



# ArcGIS CityEngine 2024.1

# Table of Contents

CityEngine PDF Help . . . . . 8

**UI Basics**

UI basics . . . . . 10

Navigator . . . . . 12

Scene Editor . . . . . 15

Viewport . . . . . 19

Inspector . . . . . 24

CGA Editor . . . . . 35

VCGA (Visual CGA) Editor . . . . . 40

Model Hierarchy . . . . . 50

Facade Wizard . . . . . 54

Status windows . . . . . 57

Layouts . . . . . 59

**Projects**

Projects . . . . . 64

Manage projects . . . . . 66

Save a file . . . . . 69

Import a file . . . . . 70

Export a project . . . . . 72

**Scenes**

Scenes . . . . . 75

3D navigation essentials . . . . . 76

Select objects . . . . . 78

Cameras . . . . . 84

Bookmarks . . . . . 85

Scene light and panorama . . . . . 87

Georeferencing . . . . . 89

**Scene Objects**

Scene objects . . . . . 96

Shapes . . . . . 97

Graphs . . . . . 98

**Shapes from graphs**

Create shapes from graph networks . . . . .	101
Block parameters . . . . .	103
Segment and sidewalk parameters . . . . .	120
Node parameters . . . . .	123
Street and intersection shapes . . . . .	130
Street and intersection shape UV . . . . .	133
Work with static models . . . . .	136
<b>Map Layer</b>	
Map layers . . . . .	139
Terrain layer . . . . .	140
Texture layer . . . . .	144
Obstacle layer . . . . .	147
Mapping layer . . . . .	149
Function layer . . . . .	150
Edit a map layer . . . . .	151
Selection with image maps . . . . .	157
<b>Drawing</b>	
Drawing . . . . .	161
<b>Draw shapes</b>	
Polygons, rectangles, and circles . . . . .	163
Push Pull tool . . . . .	167
Guides . . . . .	172
Drawing modes . . . . .	175
Snapping . . . . .	176
Intersecting shapes . . . . .	179
Cleanup shapes . . . . .	181
Align shapes to terrain . . . . .	183
Texture shapes . . . . .	184
<b>Draw graphs</b>	
Graph networks . . . . .	188
Edit street and curves . . . . .	190
Cleanup graph . . . . .	194
Align streets to terrain . . . . .	197
Generate street networks . . . . .	198

Generate a bridge . . . . .	213
<b>Editing</b>	
Editing . . . . .	218
Use transform tools . . . . .	219
<b>Edit shapes</b>	
Compute first edges and street edges . . . . .	227
Compute Edge Attributes tool . . . . .	228
Set first edge . . . . .	233
Set street edges . . . . .	234
Shape geometry tools . . . . .	236
<b>Edit graphs</b>	
Fit widths to shapes . . . . .	243
Simplify graphs . . . . .	245
Analyze graphs . . . . .	248
<b>Edit terrain</b>	
Interactive terrain editing . . . . .	251
Align terrain to shapes . . . . .	254
<b>CGA Modeling</b>	
CGA modeling . . . . .	257
Work with rules . . . . .	258
CGA essentials . . . . .	261
<b>CGA Features</b>	
CGA features . . . . .	269
Rule with parameters . . . . .	270
Conditional rule . . . . .	272
Stochastic rule . . . . .	274
CGA attributes . . . . .	275
CGA styles . . . . .	279
CGA functions . . . . .	282
Local variables . . . . .	283
Rule file import . . . . .	285
Code comments . . . . .	287
<b>Essential shape operations</b>	
Essential shape operations . . . . .	289

Extrusion . . . . .	290
Transformation . . . . .	291
Component split . . . . .	294
Subdivision split . . . . .	296
Asset insertion . . . . .	302
<b>Handles</b>	
Handles . . . . .	304
Create handles . . . . .	308
<b>Local edits</b>	
Local edits . . . . .	322
Work with local edits . . . . .	324
Rule package . . . . .	333
<b>Importing</b>	
Import data . . . . .	335
Import by drag and drop . . . . .	336
Import by dialog . . . . .	337
Get Map Data . . . . .	338
Prepare data for import . . . . .	343
DAE (Collada) . . . . .	344
DWG (Autodesk) . . . . .	345
DXF (Autocad) . . . . .	346
FBX (Autodesk) . . . . .	348
FileGDB (Esri File Geodatabase) . . . . .	349
glTF/glb (Khronos Group) . . . . .	352
IFC ( buildingSMART) . . . . .	353
KML / KMZ (Keyhole Markup) . . . . .	354
OBJ (Wavefront) . . . . .	355
OSM (OpenStreetMap) . . . . .	356
SHP (Esri Shapefile) . . . . .	360
USD (Universal Scene Description) . . . . .	363
<b>Exporting</b>	
Export data . . . . .	365
<b>Export models</b>	
Export a model . . . . .	367

ABC (Alembic)	375
DAE (Collada)	378
DATASMITH (Unreal)	382
DWG (Autodesk)	387
FBX (Autodesk)	388
FileGDB (File geodatabase)	391
glTF/glb (Khronos Group)	393
IFC ( buildingSMART)	394
KML/KMZ (Keyhole Markup Language)	395
MSPK (Mobile Scene Package)	397
OBJ (Wavefront)	398
Script-based export (Python)	400
SLPK (Scene Layer Package)	401
USD (Universal Scene Description)	403
VOB (E-on Vue)	406
Model export application notes	407
Export terrain	410
Export shapes	412
Export graphs	414
Export 3VR (360 VR Experience)	416
Share as web scene	417
<b>Content Library (ESRI.lib)</b>	
ESRI.lib	421
<b>Analysis and Measurement</b>	
Analysis	428
Dashboards	429
Scenarios	434
Measure tools	437
<b>Visibility analysis</b>	
Visibility analysis	442
Viewsheds	446
View domes	448
View corridors	450
<b>Online and Enterprise</b>	

ArcGIS Online and ArcGIS Enterprise . . . . .	453
Sign in . . . . .	454
Configure redirect URIs . . . . .	458
Get map data . . . . .	461
Sync feature layers . . . . .	462
Share data . . . . .	465
<b>ArcGIS Urban</b>	
ArcGIS Urban integration . . . . .	467
<b>360 VR Experience</b>	
Export 360 VR Experiences from CityEngine . . . . .	474
<b>Preferences and Shortcuts</b>	
Appearance . . . . .	478
General . . . . .	479
Editors . . . . .	480
Keys . . . . .	481
Miscellaneous . . . . .	484
3D mouse . . . . .	485
Mouse . . . . .	487
Touch . . . . .	488
Procedural runtime . . . . .	489
Viewport . . . . .	491
Bookmarks . . . . .	492
Cameras . . . . .	493
Light . . . . .	494
Help . . . . .	495
Network . . . . .	496
Scene . . . . .	497
Shortcuts . . . . .	498

# CityEngine PDF Help

The ArcGIS CityEngine PDF help is the main resource for learning about CityEngine. Each section begins with an overview page, and it is recommended that you read these pages first to get an idea of the content in the sections.

## Available resources online

Additional CityEngine resources are available online that are not included in the PDF document. If you are new to CityEngine, see [Introduction to CityEngine](#) and the [CityEngine tour](#) to start.

You can also connect to the [CityEngine community](#) for questions, ideas, and blogs from users and experts.

You can gain further knowledge and insight into CityEngine by reviewing the following topics in the online help:

- [Tutorials](#)
- [CGA reference](#)
- [Python reference](#)
- [Terminology](#)
- [Tips and shortcuts](#)
- [Installation and setup](#)
- [System requirements](#)

To view the CGA reference offline, click **Help > Offline CGA Reference** in the CityEngine app main menu.

Similarly, to view the offline Python reference, click **Help > Offline Python Reference**.



### Note:

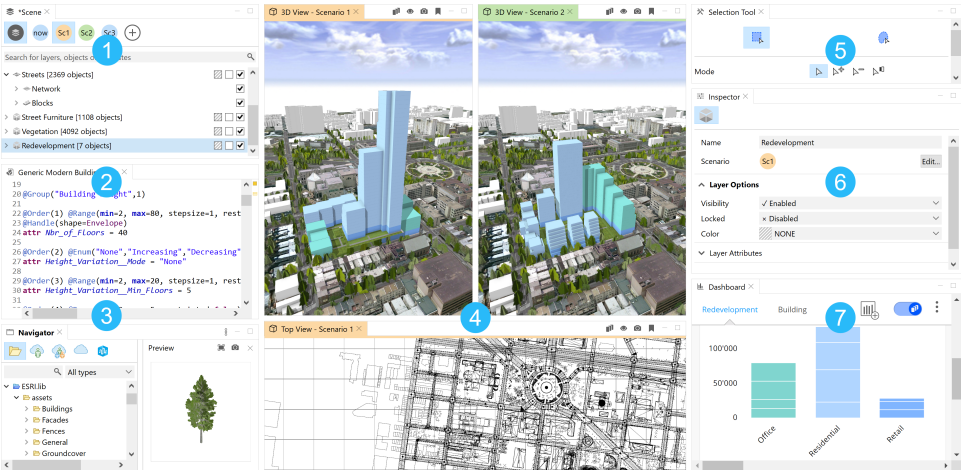
This PDF includes links to other pages in this document, as well as pages in the online help.

# UI Basics

# UI basics

## Layout

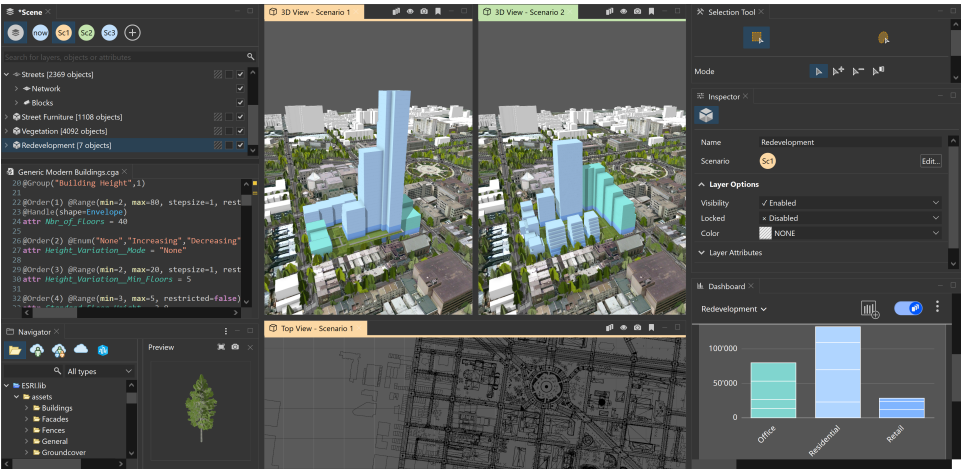
The CityEngine user interface (UI) has several windows that can be repositioned and resized to suit your needs. The following image is an example layout that compares two redevelopment scenarios:



1	<b>Scene Editor</b> —Scene, layer, object, and scenario management.
2	<b>CGA Editor</b> —Editing CGA rules.
3	<b>Navigator</b> —Manage and preview files in the workspace.
4	<b>Viewport</b> —Single-and multiple-perspective cameras and scenarios.
5	<b>Tool Options</b> —Manage options for tools such as the selection, shape creation, or transform tools.
6	<b>Inspector</b> —Detailed view and editing of selected objects and scenarios.
7	<b>Dashboards</b> —Overview of main attributes by scenario.






## Dark theme

You can also choose a **dark theme** to ease the strain on your eyes and give your scenes a more modern look.




## Additional windows

CityEngine also has the following additional windows:








- [Model Hierarchy](#) —Inspect models generated with CGA.
- [Log](#) —CityEngine messages.
- [Console](#) —CGA output.
- [Problems](#) —CGA compiler errors and warnings.
- [Progress](#) —Progress reporting of long-running CityEngine operation

# Navigator

Use the **Navigator** window  to work with your local [workspace](#), projects, and files as well as browse [ArcGIS Online](#) or on-premises [ArcGIS Enterprise](#) to exchange data with your colleagues over the cloud. You can open this window by selecting **Window > Navigator** from the main menu.

To access data on cloud services, you first need to [sign in](#) to ArcGIS Online or ArcGIS Enterprise .

The following lists the available spaces for content:


 <b>Local Workspace</b>	<p>Shows the files and folders of the current <a href="#">workspace</a> on your computer.</p> <p>If you modify files in the workspace outside of CityEngine (for example, by editing a file using the operating system's file explorer or an external program), you have to refresh the <b>Navigator</b> window. To do this, choose <b>File &gt; Refresh Workspace</b> or press <b>F5</b>.</p> <p>Using <b>Show in File Manager</b> from the <b>Navigator</b> window context menu, you can open the system file manager at the respective position.</p> <p> <b>Tip:</b></p> <p>You can copy and paste or drag files from the operating systems file explorer directly into the <b>Navigator</b> window.</p>
 <b>My Content</b>	<p>Shows all content you have added to the signed-in portal. This content is visible only to you.</p>
 <b>Groups</b>	<p>Access the list of groups to which you belong, and all the items shared with those groups.</p>
 <b>All Portal</b>	<p>Provides access to content from My Content, Groups, and all content that is shared with your organization or the public.</p> <p> <b>Tip:</b></p> <p>You need to start a search to have portal content display in the <b>Navigator</b> window.</p>
 <b>ArcGIS Urban</b>	<p>Shows the plans and projects of the currently selected urban model.</p>

## Search and filtering files

You can search for files in the active space by entering text in the search box or filter for a specific file type using the drop-down list next to the search box.

To reset the search, click the **X** icon in the search box.


## Preview files



The **Navigator** window  includes a **Preview** window to preview files such as assets, 3D objects, rasters, and CGA rules. You can double-click any file to open the **Preview** window, or right-click the file and select **File Preview**.

### **Note:**





CGA files open in CGA editor when you double-click them.

## Preview 3D models


The 3D preview can be rotated and zoomed using the same [shortcuts](#) as for the **Viewport** window . Additionally, the following set of display options are available:

- **Frame selection**  —Frames the object in the window.
- **View settings**  —Allows you to choose to view the object in Wireframe, Shaded, Textured, or Wireframe in shaded/textured mode.


CGA rules have these additional options:

- **Choose Start Rule**  —Sets the start rule applied to the shape in the preview window.
- **Select initial shape**  —Sets the shape of geometry in the preview window.
- **Scroll view or Tiles view**  —Changes how different [styles](#) are visualized.
- **X**  —Closes the preview window.

## Open files

You can open existing files to add to your workspace such as scenes, rules, or scripts. Open the files from the **Navigator** window  or from the main menu.



To open a file from the **Navigator** window , do the following:

1. Locate the file in **Navigator** window .
2. Double-click to open the file.  
Alternatively, you can right-click the file and select **Open**.



To open files from the **File** main menu, do the following:

1. Click **File > Open**.
2. Locate the file.
3. Click **Open**.

Depending on the file type, the file opens in the appropriate CityEngine window.

- CityEngine scenes open in the **Viewport** window .
- CGA rules open in the **CGA Editor** window .
- Python modules open in the python editor.
- Other files open in the system editor.

## Drag-and-drop actions

The following list describes the drag-and-drop behavior in the workspace in the **Navigator** window  to the **Viewport** window  for different file types:


- **CGA rule files and packages**—Select a set of shapes or models in the **Viewport** window, and drag a CGA file onto the selection in the viewport. The selected objects will have the CGA file assigned, and models are generated.
- **Geometry files**—The model is imported with default settings as a new static model. The model is placed at the drop spot and appears as a new layer.

- **Collada DAE files**—The Collada model is imported with default settings as a new static model. The model is placed at the drop spot and appears as a new layer.
- **CityEngine CEJ files**—Drag a CEJ file in the **Viewport** window and choose what layers to import in the pop-up window.
- **DXF files**—The DXF is imported with default settings.
- **OSM files**—The OSM file is imported with default settings.
- **Shape files**—The SHP file is imported with default settings and appears in a new layer.
- **File Geodatabase**—A dialog box appears, allowing you to decide what parts of the file geodatabase you want to import.
- **Images**—A dialog box appears, allowing you to decide how to import the image.

 **Note:**


Drag-and-drop actions work for single elements only.

# Scene Editor

The **Scene Editor** window  is the central place where you manage a scene and [scenarios](#). A CityEngine scene is organized in groups and layers. The following is a list of layer types:

- Map layer—Contains arbitrary maps (images) and can be used to globally control various parameters for scene objects. The scene [terrain](#) is also created using a map layer.
- Graph layer—Contains street networks and blocks, dynamic shapes (street shapes, blocks, and parcels), and generated models.
- Shape layer—Contains shapes, typically used as building footprints for generation of CGA models.
- Static Model layer—Contains static models, such as Collada files.
- Analysis layer—Contains analysis tools such as Viewsheds and View Corridors.
- Group layer—Contains other layers to organize the scene hierarchy.


See [Scene objects overview](#) for more information about layer types.

The **Scene Editor** window  displays the current scene as a tree with group elements, layers, and objects. You can delete, duplicate, or merge layers by selecting the corresponding menu item from the context menu or by opening **Layer** in the main menu. In addition, you can use the standard cut, copy, and paste actions to transfer objects between layers. You can rearrange layers by dragging them to the desired position.

## Create a group layer


With group layers, you can organize data by putting similar layers together based on similar geometry (for example, buildings, blocks, street networks, models) or themes (for example, elevation, imagery, terrain).

To create a layer group, do the following:

1. Right-click inside the **Scene Editor** window  and choose **New > New Layer Group**.
2. Name the group layer.

To add child layers or groups to the group you created, drag them into the new group layer.



## Layer states

When working with layers, you can turn their visibility on and off and switch their edit status and assign different colors to streamline editing in the viewport. The states are represented by the squares next to the **Scene Editor** window .



## Set the layer color

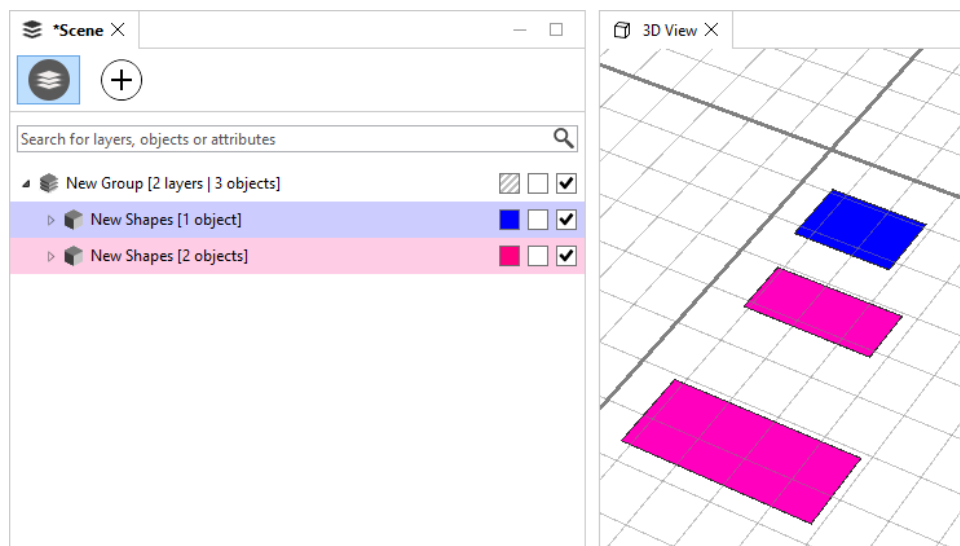
Each layer or group has a color state (colored or uncolored).

1. Click the left **Set Color** box to assign a color to the layer or group.
2. Choose a color in the palette.
3. Click **OK**.
4. Click the **Set Color** box again to remove the color.

You can access the color value in the **Scene Editor** window  and in the **Inspector** window . When a layer doesn't have a color assigned to it, the color box displays gray-striped diagonal lines.

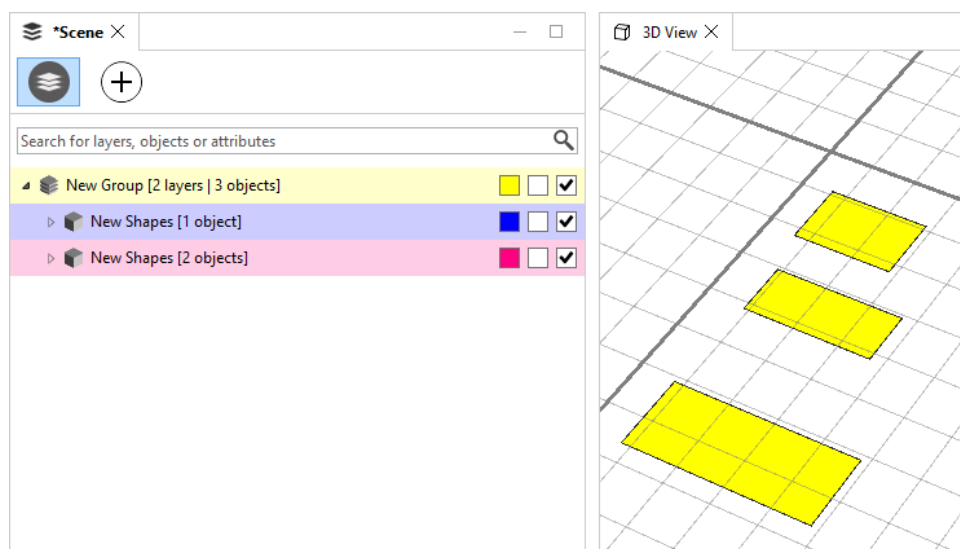
## Layer colors and hierarchies

To see how color behaves with parent and child layers, look at the examples below. The color is inherited from the group hierarchy. When you give a parent layer group a color, all child objects and models belonging to this layer or group will have that color in the **Viewport** window . In the **Scene Editor** window , the child layers will maintain their colors at all times. The example below shows two colored layers without inherited colors:




Two colored layers without inherited colors are shown.

This example, on the other hand, shows two colored layers with inherited colors.



Two colored layers with inherited colors are shown.



## Lock groups and layers

Each layer or group has a locked state (locked or unlocked). The middle check box displays the locked state of the layer. Once you lock a layer, you can't select or modify the contained objects. You can access the locked state in the **Inspector** window .


 **Note:**

When setting a group as locked, the child layers and groups will keep their original locked state but can't be selected or modified.


Turn visibility on or off

Each layer or group has a visibility state (visible or invisible). The right check box displays the visibility state of the layer or group. When disabling the visibility state, the corresponding objects will not be shown in viewports. Invisible objects can't be selected. You can access the visibility in the **Scene Editor** window  and in the **Inspector** window .

 **Note:**


- When a group is not visible, the child layers and groups will keep their original visibility state, but their objects will not be displayed in the viewports.
- The visibility of different object and layer types can also be controlled on a per-viewport basis from the view settings menu of the **Viewport** window .

Additional layer options

Renaming layers and groups	Rename a layer or a group by clicking its name twice, using the context menu (Rename entry), or by changing the name field in the <b>Inspector</b> window.
Framing elements	Double-click any element to frame the element in all viewports.
Multi selection	Select multiple layers or groups to change visibility/lock/color state for all the selected items.
Using the Search Field to select and filter objects	<p>The search field allows the user to search for objects with name match and attribute match (name and value). The result of the search is automatically selected and framed in the <b>Viewport</b> window. The result of the search action is displayed in the status bar.</p> <p>In the search field on the top of the scene window, you can type a wildcard expression (for example, "Lot*") to select matching objects.</p> <ul style="list-style-type: none"><li>• Shape*—Matches scene objects whose names start with Shape</li><li>• "Shape 12"—Matches scene objects whose names are exactly Shape 12</li><li>• "Shape* *12"—Matches scene objects whose names start with Shape AND end with 12</li></ul> <p> <b>Note:</b> Whitespace characters in a search query denote a logical OR expression. To match object names with spaces, put your search query in double quotes.</p>


Importing and exporting layers	<ul style="list-style-type: none"><li>• In cases in which building lots or footprints have been already modeled with an external program, these can be imported as shape layers. First, you have to convert your shapes into a grouped <code>.obj</code> file (each group corresponds one shape) or a <code>.dxf</code> file or similar. Then, copy it into the data folder of your project and import it by clicking <b>File &gt; Import &gt; CityEngine Layers</b> from the main menu and choose your file type, or through the <b>Navigator</b> context menu on the file. Afterward, you can apply CGA shape grammar rules in the usual way on these imported shapes.</li><li>• Any object from the scene window can be exported to a new scene by choosing <b>File &gt; Export &gt; Export Selected Objects as .cej</b> &gt; To import layers from the exported <code>.cej</code> file into a new scene, select <b>File &gt; Import &gt; CityEngine Layers</b> from the main menu.</li></ul>
--------------------------------	--

# Viewport

The 3D **Viewport** window  is mainly how you interact with the CityEngine scene besides the **Scene Editor**. You can have any number of open viewports. Select **Window > New Viewport** from the main menu to create a viewport.

To learn more on how to navigate in 3D, transform scene objects, and work with bookmarks, see [Scenes](#).


## View modes and display settings


The **Viewport** window  offers many viewing modes and display settings. These options are accessed in the top toolbar of the viewport window.

## Active scenario


You can use the **Scenario** tool  to change the [active scenario](#) for the given **Viewport** window .

## Layer type visibility




You can open the **Visibility settings**  to determine which layer types are visible for each open viewport. Selective rendering of layers can be used, for example, for rendering shapes in one viewport and geometries in another.


Item	Function	Shortcut
<b>Isolate Selection</b>	<a href="#">Isolate</a> the current selection, hide unselected objects.  <b>Note:</b> Terrains and Analyses layers will still be visible even if not selected.	Press <b>I</b>
<b>Map Layers</b>	Turns attribute map layer visibility on and off for this viewport.	Press <b>F9</b>
<b>Graph Networks</b>	Turns graph network visibility on and off for this viewport.	Press <b>F10</b>
<b>Shapes</b>	Turns shape visibility on and off for this viewport.	Press <b>F11</b>
<b>Models</b>	Turns model visibility on and off for this viewport.	Press <b>F12</b>
<b>Analyses</b>	Turns analysis visibility on and off for this viewport.	

## Camera and viewport settings

You can click the **View settings** tool  to change settings such as camera or textures. The tool has the following options:

Item	Function	Shortcut
<b>10mm fisheye lens (121° FOV)</b>	Type of lens and FOV (field of view).	
<b>18mm ultra-wide lens (90° FOV)</b>	Type of lens and FOV (field of view).	
<b>24mm wide-angle lens (73° FOV)</b>	Type of lens and FOV (field of view).	
<b>35mm standard lens (54° FOV)</b>	Type of lens and FOV (field of view).	
<b>50mm standard lens (39° FOV)</b>	Type of lens and FOV (field of view).	
<b>70mm telephoto lens (28° FOV)</b>	Type of lens and FOV (field of view).	


Item	Function	Shortcut
<b>135mm telephoto lens (15° FOV)</b>	Type of lens and FOV (field of view).	
<b>Parallel projection view</b>	Type of lens and FOV (field of view).	Press <b>D, P</b>
<b>Two point perspective correction</b>	Displays vertical edges as parallel at shallow camera tilt angles.	
<b>Wireframe</b>	<p>Renders the scene contents as the Wireframe of the face borders.</p> <p> <b>Tip:</b> To see the wireframe of terrains, select them in the <b>Scene Editor</b> and enable the <b>Wireframe</b> option in the <b>Inspector</b>.</p>	Press <b>4</b>
<b>Shaded</b>	<p>Renders the scene contents with colors but with no textures.</p> <p> <b>Note:</b> This also means that no opacity maps are applied and some otherwise transparent parts are rendered opaque.</p>	Press <b>5</b>
<b>Textured</b>	Renders the scene contents with colors and textures.	Press <b>6</b>
<b>Wireframe on Shaded/Textured</b>	Enable or disable an overlay such as the wireframe render mode but on top of the rendered elements.	Press <b>7</b>
<b>Shadows</b>	<p>Turns shadows on or off.</p> <p> <b>Note:</b> Note that enabling shadows on large models may considerably affect rendering performance.</p>	Press <b>8</b>
<b>Ambient Occlusion</b>	Enable or disable ambient occlusion rendering ( <a href="#">ambient occlusion</a> ).	Press <b>9</b>
<b>On-camera light</b>	Switches the sun position between the specified location and straight behind the camera.	Press <b>L</b>
<b>Single-Sided Lighting</b>	<ul style="list-style-type: none"> <li>• Switches between single- and double-sided lighting.</li> <li>• With single-sided lighting, faces are rendered the same whether you look from the front or the back side.</li> <li>• With double-sided lighting, faces are rendered as though they were a physical wall that looks different on both sides.</li> </ul>	
<b>Force backface culling</b>	Enforce backface culling by ignoring the CGA material.doubleSided property and only rendering the faces facing the camera.	
<b>Terrain Masking</b>	Enable or disable masking of multiple terrains in <a href="#">overlapping terrain regions</a> .	

Item	Function	Shortcut
Information Display	Turns the information display on and off. The information display provides statistics for the current scene such as number of objects and polygon count.	Press <b>D,D</b>
Axes	Switches rendering of the axis visualization in the lower left corner.  <b>Note:</b> The coordinate system can be switched in <b>View Coordinate System</b> .	Press <b>D,A</b>
Compass	Switches rendering of the Compass in the lower right corner.	Press <b>D,C</b>
Grid	Switches rendering of the Grid at elevation 0 for reference.	Press <b>D,G</b>
Bookmarks gizmos	Turns visibility on and off. Switches rendering of small cameras for the <a href="#">Bookmarks</a> locations.	
Handles	Turns <a href="#">handles</a> visibility on and off.	
View Coordinate System	Choose a Coordinate System for the Navigation Display and the axis visualization	

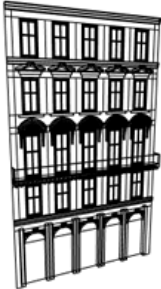
 **Note:**

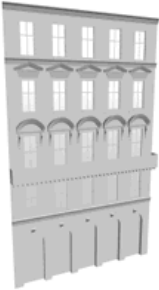


Press **D** to open the shortcut table for the display of Information, Axes, Compass, and Grid.

Render modes

The **Viewport** window  can render its contents in the following modes:

**Wireframe**—All edges of objects are drawn as lines. There are no filled polygons so you can "look-through" the objects. The wireframe mode can be activated by selecting the **Wireframe** button or pressing **4**.



<p><b>Shaded</b>—All polygons are filled with the object color. No texturing takes place. The shaded mode can be activated by selecting the <b>Shaded</b> button or pressing <b>5</b>.</p>	
<p><b>Textured</b>—All polygons are filled with textures and object colors. The textured mode can be activated by selecting the <b>Textured</b> button or pressing <b>6</b>.</p>	
<p><b>Wireframe on Shaded/Textured</b>—Overlay wireframe independent of the viewing mode by enabling <b>Wireframe on Shaded/Textured</b> in the <b>View Settings</b> menu or pressing <b>7</b>.</p>	

Viewport settings are stored per viewport together with your scene data. This means that when you reopen a scene, you get exactly the same viewport settings as when you were saving the scene.

 **Note:**


Global viewport settings can be changed by selecting **Edit > Preferences > General > Viewport** from the main menu; see [Viewport preferences](#).

### Viewport context menu

Depending on the [selection](#), the context menu contains different entries for different objectives.

<b>Frame (F)</b>	Frame the selection (or the whole scene if selection is empty).
<b>Select Objects in Same Layer</b>	All objects in the same layer or layers are selected.
<b>Select Objects with Same Rule File</b>	Selects all objects having assigned a rule file that is present in the source selection.

Select Objects with Same Start Rule	Selects all objects having a start rule that is present in the source selection.
-------------------------------------	--

Additionally, you can **Cut**, **Copy**, **Paste** or **Delete** in the context menu. Finally, you can also turn the full screen mode on and off for this **Viewport** window .

 **Tip:**

If you have a [3D mouse](#) installed, you can change the mouse options here. These options are applied for all viewports.

## Make names unique



Names of generated objects in a CityEngine scene are not automatically unique. In some cases, unique names are required. This can be achieved by clicking **Edit > Make Names Unique Tool**.


All selected scene objects are enumerated and renamed in ascending order. The delimiting character can be chosen.


 **Note:**

This is a one-time operation. Once you modify your scene (add scene objects), there might be new shapes with nonunique names again.

# Inspector


The **Inspector** window  is the main tool for viewing and modifying the attributes and parameters of [scene objects](#), such as shapes, graphs, map layers, or static models. Depending on the type of object selected, the **Inspector** window  user interface changes and provides access to the object's properties.

To open the **Inspector** window , click **Window > Inspector** in the main menu, or press **Alt+I**.

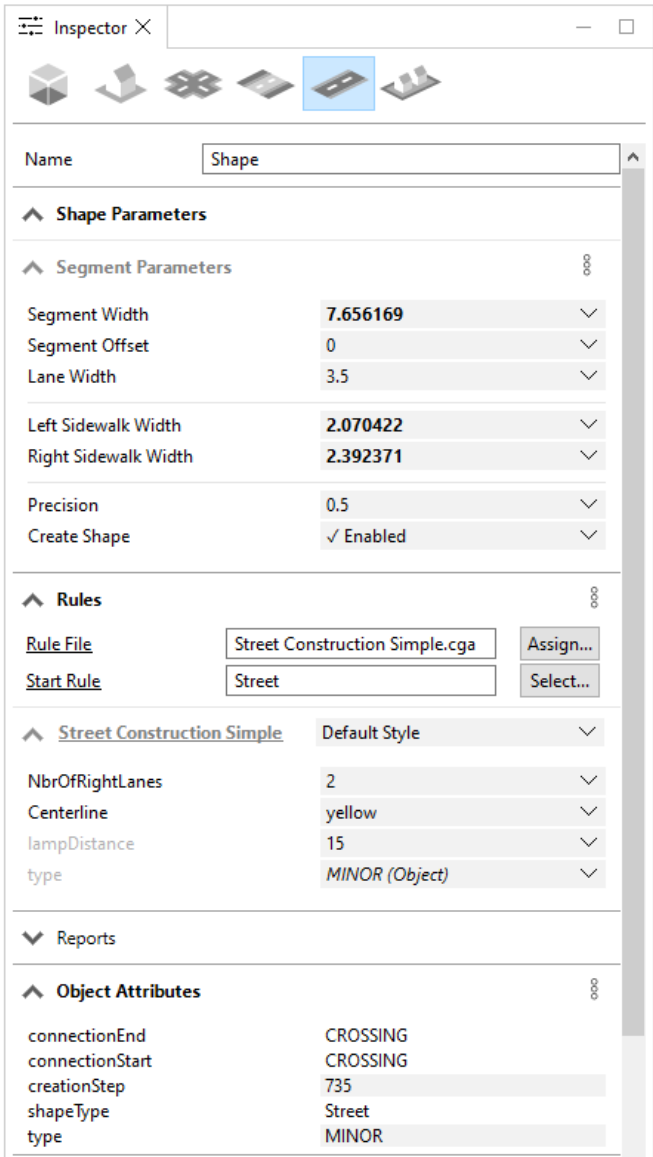
The **Inspector** window  supports both the editing of properties of a single and a collection of objects. Attributes that are unique across all objects are shown as is. If an attribute has different values in the object collection, the attribute is marked as nonunique with a question mark symbol. Attributes of the lead object are displayed when multiple objects are selected. The lead object is the most recent individually selected object.

For map layers, you can change the map files, modify the bounds, and adjust the display offset (how much the rendering of the map is displaced in relation to the actual map values). In addition, an overlay color and alpha value for the map can be specified. See [Map layer overview](#) for more information.

## Exploring properties

The properties available in the **Inspector** window , depend on the types of scene objects selected in the [Viewport](#). You can switch between the different object types by clicking their respective icon. Also, if you hover over the icon, you can see how many objects of that type are selected.

The following image is an example from a selection of different scene objects, in which the segment properties are displayed:



## Parameters

Parameters drive the creation of street shapes. You can use the **Parameters** section to adjust any of the street parameters and change their shape geometry, such as street width or node type.



The following scene objects contain parameters: segments, sidewalk, nodes, and blocks. See [Segment and sidewalk parameters](#), [Node parameters](#), and [Block parameters](#) for more information.

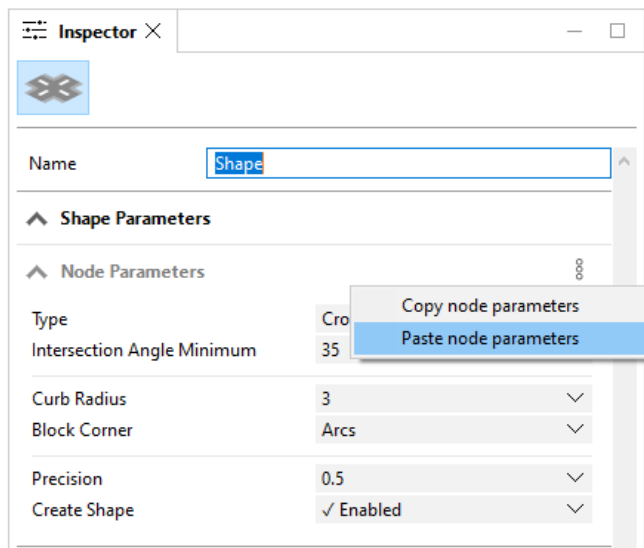
Parameters can have different sources that allow fine-tuned control over the values. See [Sources](#) for more information.

### Note:

Shapes and lots do not have parameters.

## Copy and paste parameters

For efficient editing, you can copy and paste the parameters from one selected object to a single or multiple selected objects of the same type. To do this, select an object that uses parameters, such as a node or segment parameter. Next, click **More options**  and select **Copy {parameter type} parameters**. Select the objects in which to apply the parameters. Finally, click **More options**  and select **Paste {parameter type} parameters**:




## Rules

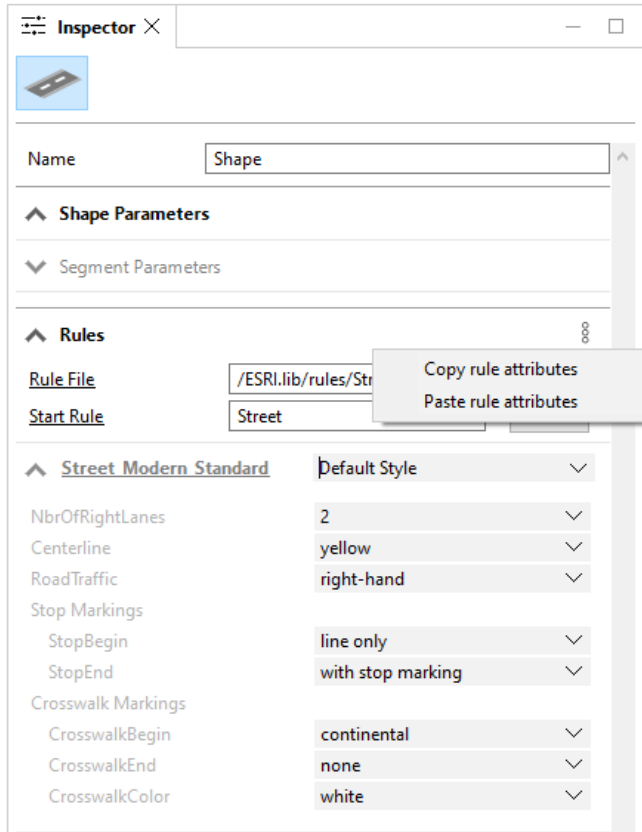
In the **Rules** section, you can control CGA rules which may contain attributes, such as color, building style, number of floors, or zoning type.

Rules can be assigned to the following scene objects: shapes, segments, sidewalks, nodes, and lots.

Rule attributes can have sources that allow for precise control of values. See [Sources](#) for more information.

## Copy and paste rule attributes

You can copy and paste rule attributes to apply to other objects of any shape type in the scene. Follow the same steps as listed in the [Parameters](#) section. This time, click **More options**  in the **Rules** section to copy and paste the rule attributes:



See [Rule-based modeling](#) and [Work with rules](#) for more information on rules.

## Object attributes


Under the **Object Attributes** section, object attributes are listed that are attached to the selected scene objects.

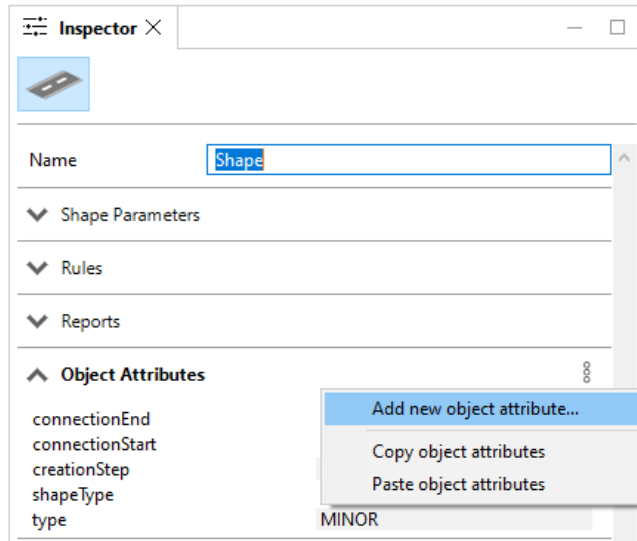
The following scene objects can have object attributes: shapes, segments, nodes, sidewalks (displays the attributes from a segment or node), blocks, lots, and static models.

Some object attributes are automatically created on import or while generating dynamic shapes. Attribute values which can be edited or deleted have a gray background, while read-only attributes have a white background.

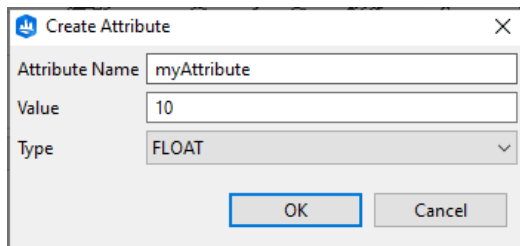
### Add an object attribute

To add an object attribute, do the following:

1. Click **More options**  and select **Add new object attribute**.



The **Create Attribute** dialog box appears:



2. Provide information for the **Attribute Name**, **Value**, and **Type**.

CityEngine has the following object attribute types:



- BOOL—Boolean
- FLOAT—Floating point value (double precision)
- STR—Character string (UTF-16 encoded)
- BOOL[ ]—Array of Boolean values
- FLOAT[ ]—Array of floating point values
- STR[ ]—Array of character string

**Note:**

- When providing the value of an attribute, you can provide a NaN value for a float (Not a Number), or a NULL value for a string.
- When providing the value for attributes arrays, separate the values with a comma, such as the following:
  - 1,2,3,4,NaN,6
  - a,b,c,d,e,f,NULL,h,i,j
  - true,false,true,false
- Array values can also be formatted with square brackets for the ease of copying from CGA code or from the console output:
  - ["a", "b", "c"]
  - [a,b,c]

3. Click **OK**.

## Copy and paste object attributes

You can also copy and paste object attributes to apply to other objects in the scene. Follow the same steps as listed in the [Parameters](#) section, but instead click **More options**  in the **Object Attributes** section of the **Inspector** window  to copy and paste the object attributes.

## Duplicate and delete object attributes


Object attributes can be duplicated or deleted by right-clicking any object attribute. Read-only attributes can only be duplicated.

## Array attributes

CityEngine supports float, string, and Boolean arrays. You can view and edit arrays for both rule and object attributes.

### Edit attributes in array table

To create or modify array attributes, do the following:

1. Click the **Expand menu** button  to expand the array attributes.

Array indices are displayed in gray.

^ myFloatArray[6]		[1, 2, 3, 4, 5, 6]
0		1
1		2
2		3
3		4
4		5
5		6
		+

2. Click the add button + to add a new row to the end of the array.  
The added element is set to the default value for each type (in other words, 0 for floats, "" for strings, false for Boolean).

3. Click an element to edit it.
4. Right-click to add, duplicate, or delete a row.

2D arrays are displayed in a table view. Row indices are displayed on the left side. Column indices are displayed on top.

^ myFloatArray[3,2] [1, 2; 3, 4; 5, 6]


	0	1
0	1	2
1	3	4
2	5	6

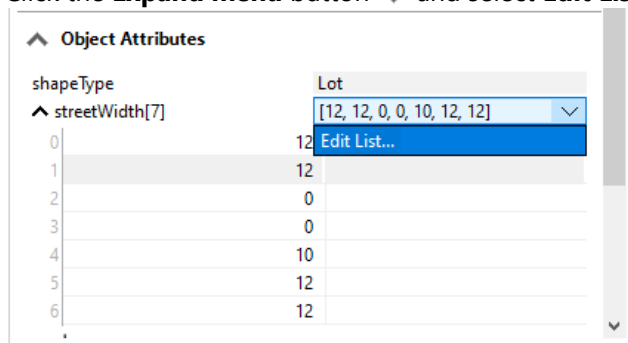
+

2D arrays are displayed in a table.

## Edit list with array menu

To edit a list with the array menu, do the following:

1. Click the **Expand menu** button  and select **Edit List** from the drop-down menu.



2. Modify the values in the **Edit List** dialog box.  
You can replicate the elements, add, duplicate, or delete elements using the toolbar.  
You can also right-click to add, duplicate, or delete a row.
3. Click **OK**.  
The new values are added to the data model and scene.

## Edit table with array menu

When an object has many attribute arrays that follow the `prefix_...` syntax (for example, `MyAttribute_A`, `MyAttribute_B`, `MyAttribute_X`), you can edit the attribute values from the drop-down menu and select **Edit Table** for each of the attributes.

To edit the multiple array attributes, do the following:

1. Click **Edit Table** from the drop-down menu of one of the attributes.  
This opens a table of array attributes for editing.
2. Modify the values in each column.
3. Click **OK**.

## Sources

Attributes and parameters can have different sources that allow fine-grained control over the values. The following is a list of possible sources:

<b>Default</b>	The default value used. For rule attributes, this is the initial attribute value. For parameters, this is the algorithm-specific default.
<b>User</b>	A value that is entered by the user. Whenever the user sets a value (also by Python), uses a slider or handle, the source is set to User defined.
<b>Object</b>	The value is taken from the corresponding object attribute. The value is displayed in italic and marked with (Object).
<b>Shape</b>	A rule attribute can use the value from the parent shape. For example, a street shape may sample the street segment's <i>streetWidth</i> . The value is displayed in italic and marked with (Shape).
<b>Layer</b>	The value is connected to a layer attribute. The value is displayed in italic and marked with the source layer in brackets. See <a href="#">Edit map layer attributes</a> for details.

 **Note:**

- The most efficient way to create connections and set sources is to use the [Connection Editor](#).
- During rule assignment, object attributes with names that match a rule attribute are connected automatically.
- Whenever you type a value to a parameter or attribute, it will automatically change to user source.

## Map attributes with the Connection Editor


Attributes and object parameters in CityEngine can be controlled from various sources. The **Connection Editor** menu helps to create these attribute connections.

To map the connection of a specific attribute, do the following:

1. Click the drop-down menu of the attribute.
2. Click **Connect Attribute** to open the **Attribute Connection Editor** menu.
3. Set the attribute connection.
4. Click **OK**.

In the **Attribute Connection Editor** menu, you have the following options:

<b>Object Attributes</b>	<p>Connect your attribute to an object attribute. This option is only available if the following occur:</p> <ul style="list-style-type: none"><li>• The scene object has an object attribute with a matching name.</li><li>• The type of the object attribute matches the required type of the attribute.</li></ul>
--------------------------	---

<b>Shape Parameters</b>	<p>Connect your attribute to the parent shape parameter. This option is only available if the following occur:</p> <ul style="list-style-type: none"> <li>• It is a rule attribute.</li> <li>• The rule file is attached to a shape of an intersection, street, or block (a lot).</li> <li>• The attribute has the same name as a shape parameter.</li> <li>• The type of the shape parameter matches the required type of the attribute.</li> </ul>
<b>Layer Attributes</b>	<p>Connect your attribute to layer attributes from arbitrary layers. Choose the source layer from the drop-down menu, and the desired layer attribute from the drop-down menu.</p> <p>Each layer will provide a list of its available layer attributes, consisting of the following:</p> <ul style="list-style-type: none"> <li>• Channel attributes for map layers—The color channels (red, blue, green, alpha, brightness, and so on.) of the layers image. They are marked as (Map Channel).</li> <li>• Object attributes—Object attributes of objects in the source layer. They are marked as (Object attribute).</li> <li>• Other attributes—Existing expressions or mappings. They are marked as (Layer attribute) or (Expression).</li> </ul> <p> <b>Note:</b> If a layer attribute with the same name as your attribute already exists, the new attribute might override the existing one.</p>
<b>Reset Attributes</b>	<p>You can reset a specific attribute through the <b>Rule default</b> option in the attribute drop-down menu. Depending on the attribute's type, this resets to the default value of the algorithm (for example, Street Shape creation) or to the default value from the rule file (rule attribute). Additional options are available from the attribute context menu (open by right-clicking):</p> <ul style="list-style-type: none"> <li>• <b>Reset user attributes</b>—Reset all user-set attributes of this rule file to its default (Rule) values.</li> <li>• <b>Reset all attributes</b>—Reset all user-set and mapped attributes of this rule file to its default (Rule) values.</li> </ul>

## Map object attributes with layer attributes

Every layer can have an arbitrary set of layer attributes defined. Whereas, map layers normally use their image data as the source for layer attributes, graph and shape data layers can query their vector objects to layer attributes.

### Object attribute mapping and sampling

Layer attributes can be used to map object attributes of their scene objects to attributes with different names or to objects on other layers.

 **Note:**

For simple cases, use the [Connection Editor](#) to perform attribute mapping.

Object attributes of nodes, segments, and shapes can be mapped or sampled with layer attributes using one of the following commands:

```
getFloatObjectAttr(name)
getFloatObjectAttr(name, sample)

getStringObjectAttr(name)
getStringObjectAttr(name, sample)

getBoolObjectAttr(name)
getBoolObjectAttr(name, sample)

getFloatArrayObjectAttr(name)
getFloatArrayObjectAttr(name, sample)

getStringArrayObjectAttr(name)
getStringArrayObjectAttr(name, sample)

getBoolArrayObjectAttr(name)
getBoolArrayObjectAttr(name, sample)
```

These commands search for object attributes within the layer for a matching name.

If the `sample` argument is false, only the shape's object attributes are examined. If `sample` is true (the default value), and the shape has no such object attribute, overlapping shapes in the attribute layer are sampled for the specified name.

If you use the object attribute `width` on street segments to control the width of created street shapes (the street parameter `streetWidth`), `getFloatObjectAttr` allows the attribute layer to obtain the value from other objects.

```
attr streetWidth = getFloatObjectAttr("width")
```

To use this layer attribute, set the source of the `streetWidth` parameter in the **Street Parameters** pane to its own layer. Below is an example with the `streetWidth` parameter dependent on the attributes:

#### Layer Attributes

```
//-----
// Example OSM Tag Mapping

streetscale = 1 // street width scale factor

width = getFloatObjectAttr("width", false)
lanes = getFloatObjectAttr("lanes", false)

attr streetWidth = // street width dependeding on available attributes
  case width > 0 : width * streetscale
  case lanes > 0 : lanes * 3.5 * streetscale
  else : streetWidthByClass * streetscale * oneway


class = getStringObjectAttr("highway", false)

streetWidthByClass =
  case class == "primary" : 8
```

The `streetWidth` parameter is dependent on other attributes.

The `streetWidth` attribute is mapped from object attribute `width`. The layer attribute `streetWidth` can now be used to control the street width of street shapes.

 **Note:**

When importing OSM, shape, or GDB data, a predefined set of layer attributes is automatically created on the imported layers. Select the new layers and show or modify the created layer attributes in the **Inspector** window .

## Reports

The **Reports** section lists all reported variables for selected models. The following table illustrates examples:

Reports									
Report	N	%	Sum	%	Avg	Min	Max	NaNs	
FAR	77	100.00	14.71	100.00	0.19	0.13	0.23	0	
FAR.Office	71	92.21	13.53	91.96	0.19	0.13	0.23	0	
FAR.Retail	6	7.79	1.18	8.04	0.20	0.18	0.23	0	
FacadeOrientation	416	100.00	26916.07	100.00	64.70	1.69	137.31	0	
FacadeOrientation.East	77	18.51	7148.92	26.56	92.84	60.79	137.31	0	
FacadeOrientation.North	103	24.76	6286.00	23.35	61.03	1.69	98.98	0	
FacadeOrientation.South	118	28.37	6329.14	23.51	53.64	20.61	99.08	0	
FacadeOrientation.West	118	28.37	7152.02	26.57	60.61	24.13	108.09	0	
Footprint Area (m2)	3	0.00	1547.39	0.00	515.80	472.52	601.00	0	
GFA	77	100.00	38491.82	100.00	499.89	343.55	601.00	77	
GFA.Office	71	92.21	35397.04	91.96	498.55	343.55	601.00	0	
GFA.Office#color	71	92.21	#80d5cf;#80d5cf;#80d5...	0.00	#80d5cf	#80d5cf	#80d5cf	71	

The Reports table in the Inspector window is shown.

Report	The name of the report variable calculated.
N	The number of occurrences of the report variable.
% (N)	The percentage the variable occurs in a group.
Sum	The sum of variable values.
% (Sum)	The percentage of the sum of variable values in a group.
Avg	The average of the variable values.
Min	The minimum of the variable values.
Max	The maximum of the variable values.
NaNs	The number of occurrences with values that are not a number. For example, string variables, such as <code>Retail</code> or <code>Office</code> , don't have numeric values.

 **Note:**


Report variables may contain a dot that separates a common group name and a variable name, such as `FAR.Office` or `FAR.Retail`. The variables are combined in the `FAR` group.



See [Tutorial 11: Reporting](#) for more information. Also, the [report operation](#) has further details about reporting with CGA rules.

# CGA Editor



CGA (Computer Graphic Architecture) is the unique programming language for ArcGIS CityEngine that you can specify to generate 3D content for urban planning. Use the **CGA Editor** windows to create and modify **CGA** rules that are applied to shapes to create 3D geometry and architecture.

## CGA Editor

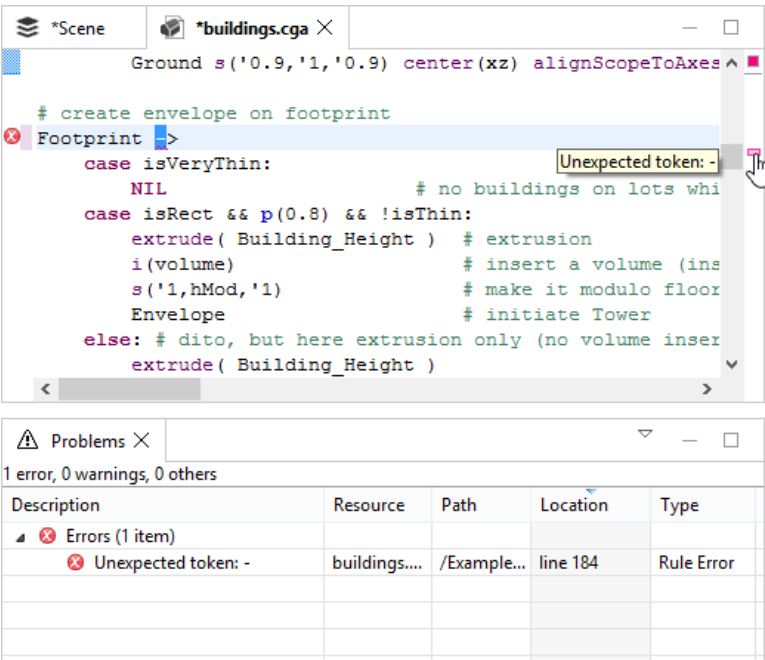
In the **CGA Editor** window , you can write, modify, and save **CGA** rules. To edit a CGA rule, do the following:

1. Open the CGA rule file.
  - Double-click the rule file in the **Navigator** window .
  - Click **Rule File** in the **Inspector** window  if the selected object has a rule file assigned to it.
2. Make necessary edits.
3. Press **Ctrl+S** to save the CGA rule file or click **File > Save** in the main menu.

### Note:

- The **CGA Editor** window  has syntax highlighting for better readability of the code.
- The **CGA Editor** window  detects syntax errors and highlights them with a white X on a red background. Click **Window > Show Problems** in the main menu for a list of syntax errors.
- Press **Ctrl+Spacebar** when typing to complete a command.



## Manage rule errors and warnings



A rule file with a syntax error is shown.

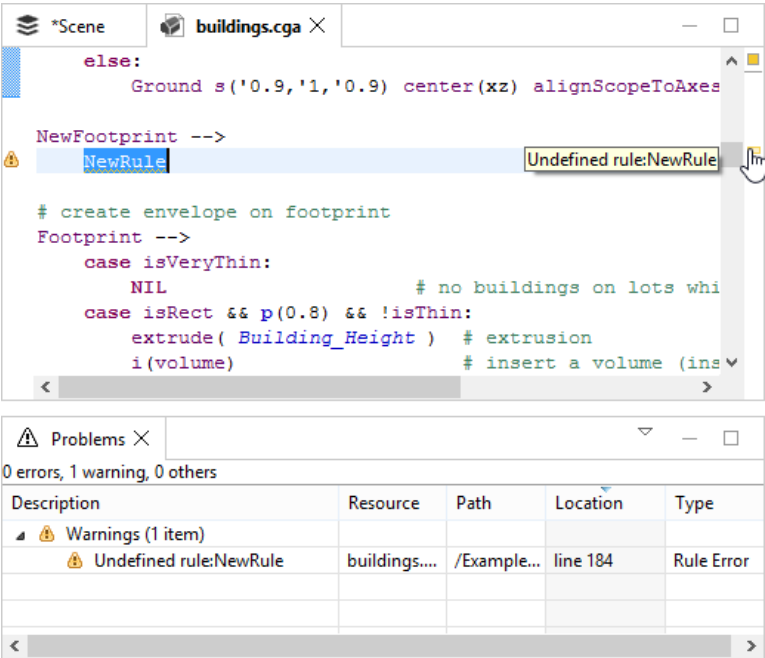
The syntax error in the CGA code is detected automatically and marked red.

 **Note:**

The position of an error is indicated by a small red box next to the scrollbar on the right. More detailed information about the error can be found in the **Problems** window  or by hovering over the red indicators or error markers in the **CGA Editor** window .

Errors must be resolved before applying the rules. It is not possible to generate models if the assigned rule file contains errors.



The **CGA Editor** window  also issues warnings:

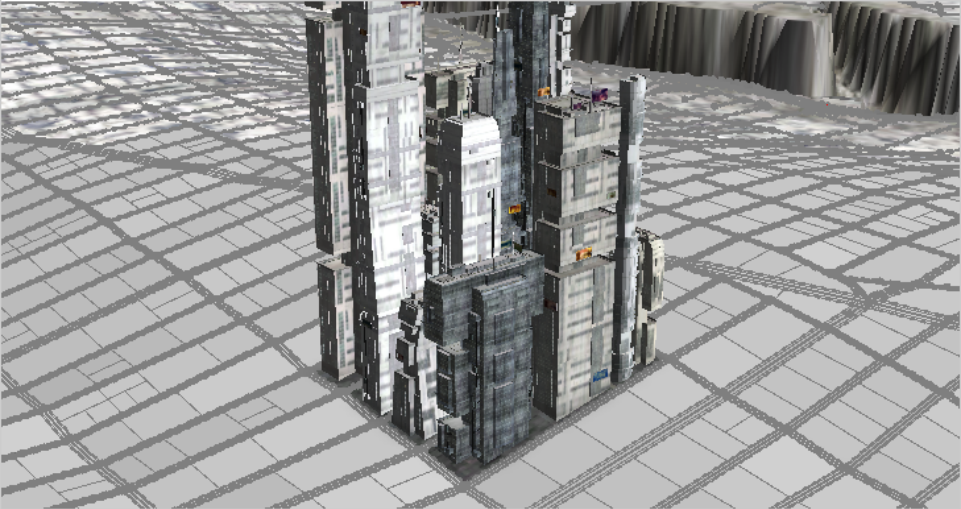


An undefined rule file warning is shown.

In this case, the rule `NewRule` is not defined. This is not necessarily a problem. Warnings indicate potential problems but still allow generation.

## Problems view

You can open the **Problems** window  using **Window > Show Problems**. On top of Rule Errors (static compile errors), the **Problems** window  also shows Model Errors (dynamic runtime errors), in other words, problems encountered during generation of a model. Such errors and warnings depend on the rule as well as on the initial shape (in other words, its geometry and attributes such as the seed, and so on). The **Problems** view allows you to find and resolve such problems. In the example below, a number of buildings were generated and an "asset not found" error was returned.



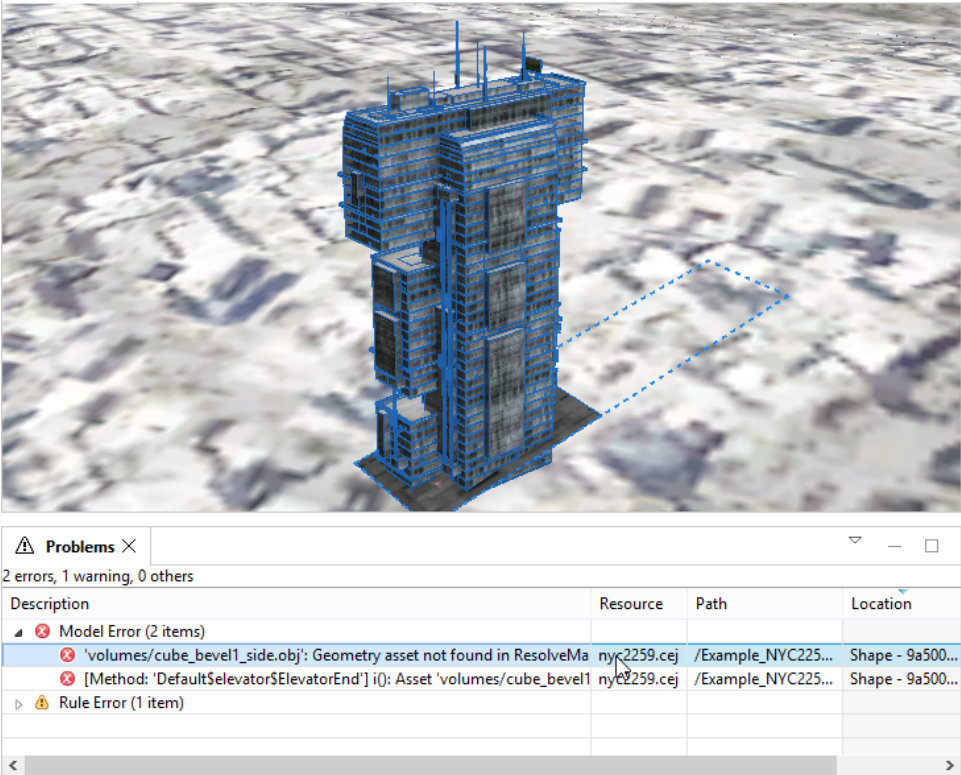
Problems X

2 errors, 1 warning, 0 others

Description	Resource	Path	Location
Model Error (2 items)			
'volumes/cube_bevel1_side.obj': Geometry asset not found in ResolveMa	nyc2259.cej	/Example_NYC225...	Shape - 9a500...
[Method: 'Default\$elevator\$ElevatorEnd'] i(): Asset 'volumes/cube_bevel1	nyc2259.cej	/Example_NYC225...	Shape - 9a500...
Rule Error (1 item)			

A model error is shown.

To find the according model or shape, double-click the error, and the model plus shape will be selected and framed. The picture below shows the initial shape and the generated model in which the generation resulted in the "asset not found" error.



An asset not found error is shown.

### Configuration

The **Problems** window can be configured according to your taste and needs. Here are the recommended settings:

- Right-click in the **Problems** window and select **Group By > Type** to group the errors by their type (in other words, separate Rule Errors and Model Errors):
- Click **Location** to sort the errors by Location (in other words, by their initial shape).
- Right-click and select **Configure Contents**.
  - Deselect **Use item limits** to disable the default limit of 100 markers.

### Code completion

The **CGA Editor** window features automatic code completion. At any position in the CGA code, you can press **Ctrl+Spacebar** and a pop-up appears with a suggestions that match the current context. Use the cursor keys or the mouse to choose one.


### Keyboard shortcuts

Keyboard shortcuts for working with the **CGA Editor** window include the following:

- **Ctrl+S** —Save the file (changes must be saved before generation; files with changes are marked with an asterisk (\*) on the tab).
- **Ctrl+G** —Generate (the selected objects, in other words, shapes or models).
- **Ctrl+F5** —Re-generate all models.

- **Ctrl+F** —Opens the Find / search-replace dialog box.
- **Ctrl+L** —Opens the Go to line dialog box.
- **Ctrl+Shift+L** —Shows all shortcuts.

 **Note:**

These shortcuts only work if the **CGA Editor** window  is the current view (in other words, its tab is highlighted).

# VCGA (Visual CGA) Editor

The **Visual CGA Editor** is a node-based programming interface that simplifies building procedural designs. It is based on a high-level set of [architectural components](#) and [materials](#) released with the ESRI.lib, designed to support a wide range of massing typologies usable across VCGA, CGA, and Urban environments.

Download the VCGA Playground example to have access to the following:

- Multiple examples for VCGA design.
- A rich CGA component library. CGA components are nodes in a VCGA design; you can use the library to create your own designs.

Open CityEngine and click **Help > Download Tutorials and Examples > Example VCGA Playground** in the main menu. The **Example VCGA Playground** project is automatically downloaded and added to your CityEngine workspace.

You can also access the [VCGA Playground example](#) on ArcGIS Online. There are accompanying [video tutorials](#) to help get you started.

## Visual CGA Editor

In the **Visual CGA Editor** window, you can write and modify VCGA designs which are identified by the `.vcga` extension. To edit a VCGA design, do the following:

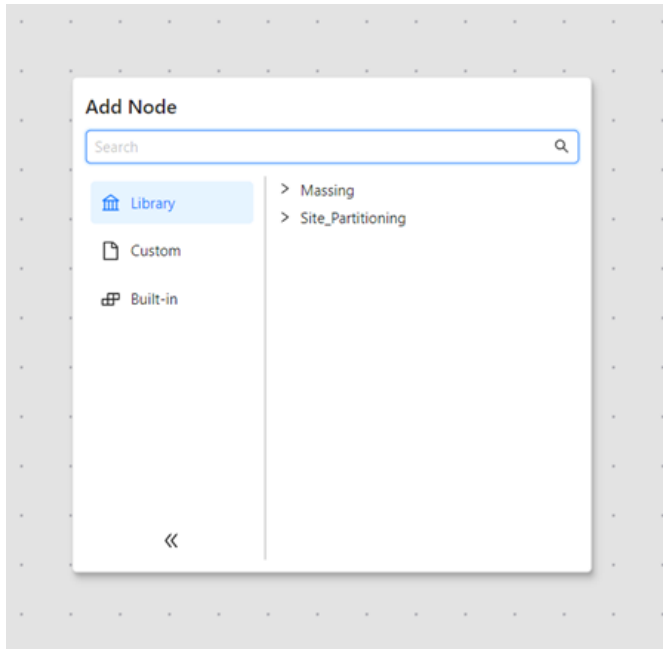
1. Open the VCGA design file:
  - Double-click the design file in the **Navigator** window.
  - If a selected object has a VCGA design file assigned, click **Rule File** in the **Inspector** window to open it.
2. Edit the design.

The **Visual CGA Editor** automatically saves the VCGA design with every change and generates the 3D model if it is assigned to a selected shape in the **Viewport** window. This means there is no need to manually save and generate the VCGA design.

## Visual CGA Editor nodes

The **Visual CGA Editor** utilizes a node-based system in which nodes are connected to implement procedural designs. Each node has input and output slots, in which circular slots represent values and square slots represent shapes.

To add nodes, double click anywhere in the **Visual CGA Editor** window, or right-click in the window and select **Add Node**. This opens the **Add Node** dialog box:



The **Add Node** dialog box has the following types of nodes:

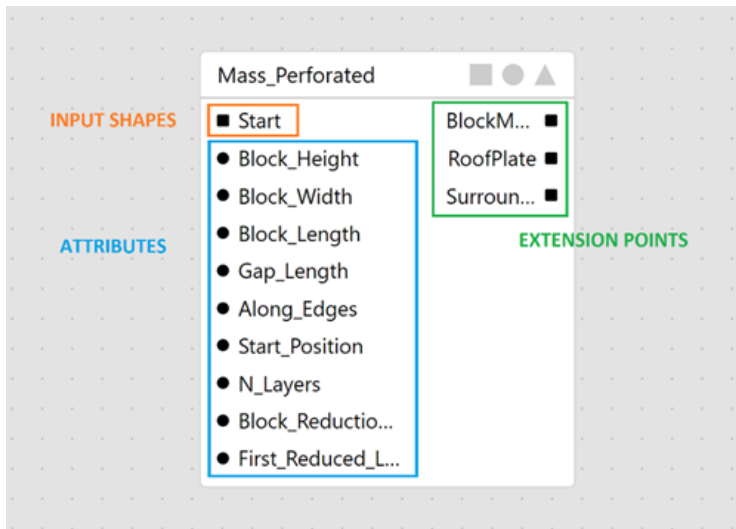
- **Library**—Displays the [architectural components](#) provided by the ESRI.lib.
- **Custom**—Displays the custom components located in the `rules` folder of the current project.
- **Built-in**—Displays the built-in nodes provided by the editor, such as input, attribute, extension, and flow control nodes.

This layout helps you to quickly find and use different types of nodes to build your procedural designs efficiently.


## Component nodes

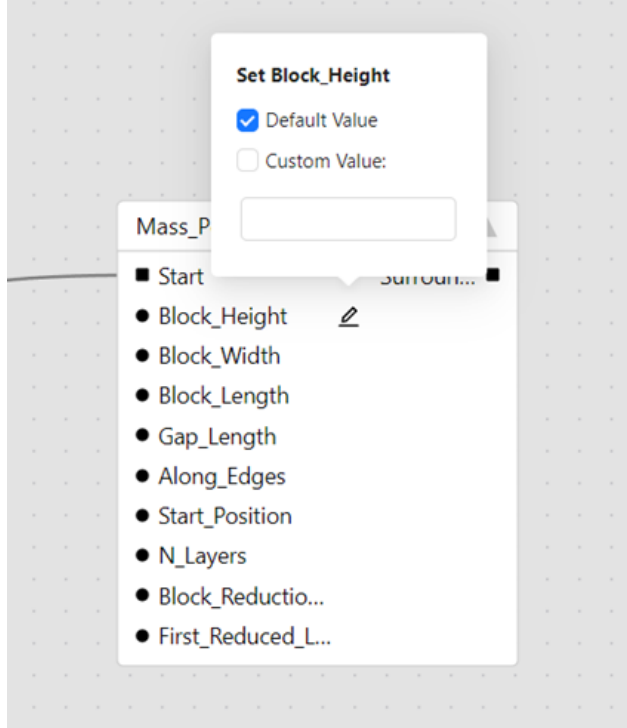
Each component node in the **Visual CGA Editor** window corresponds to either a CGA or a VCGA file. You can directly open this file using the **Open Component File** option in the node context menu that displays when you right-click the node.

## Body



The body of each component node includes the following three distinct slot sections:

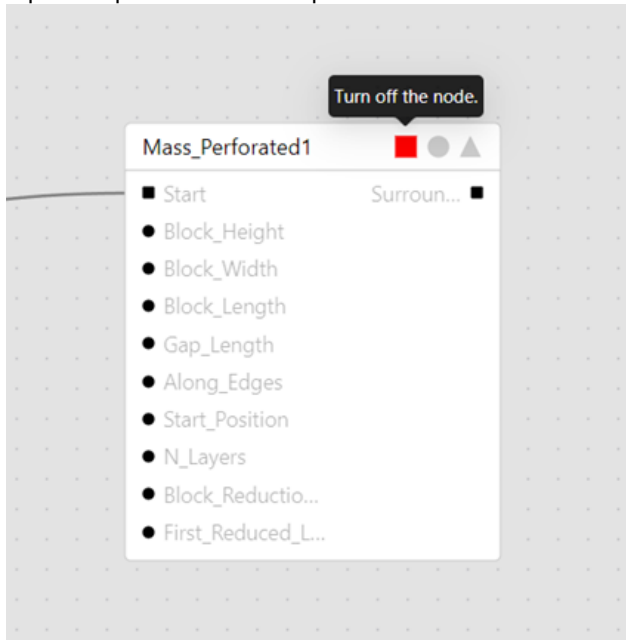
- Input shape slot—This slot represents the component’s input shape.
- Extension slots—These slots represent the component’s extension point shapes.
- Attribute slots—These slots represent the component’s input parameters. Each slot allows you to set a value in two ways:
  - You can create a value or attribute node and connect it to the slot.
  - You can use the node mini-inspector pop-up that opens when you click **Edit**  when hovering over the slot. Initially, every attribute slot uses the component’s default value, but you can choose to set new values.



### Title bar

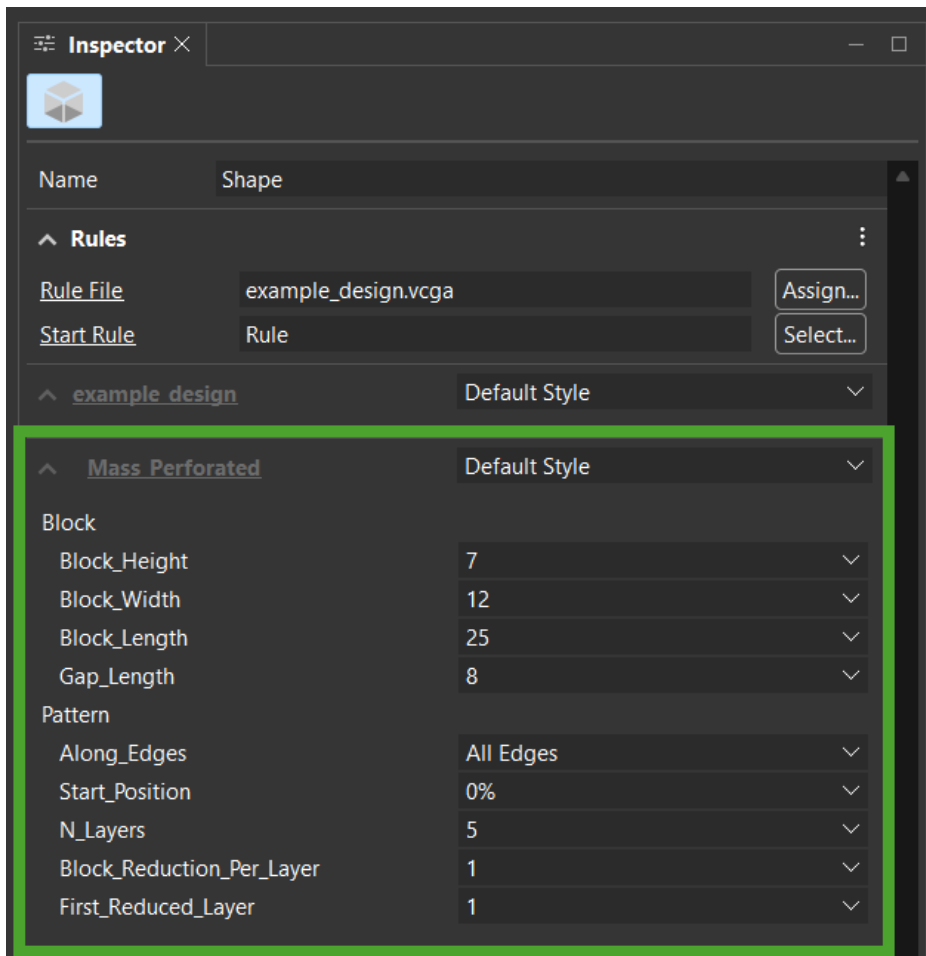
In the component node's title bar, there are buttons that have the following functionalities:

- The square button toggles a node on or off. Turning a node off activates the NIL operation, which removes the input shapes from the shape tree.

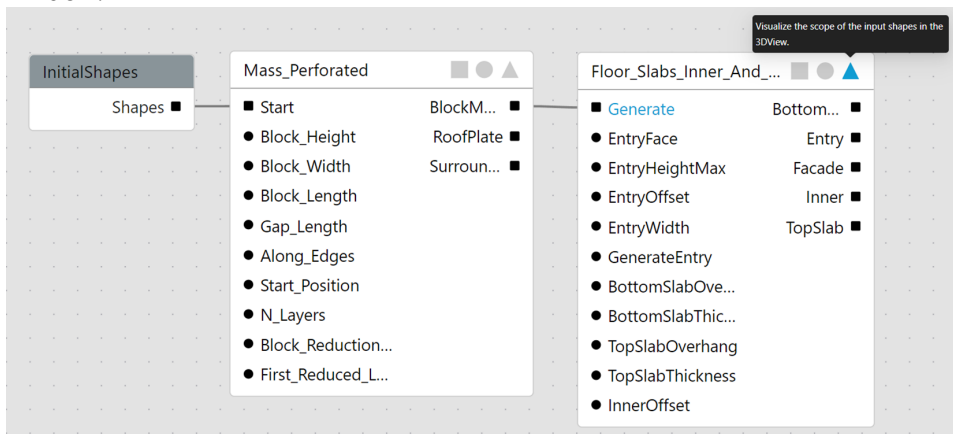


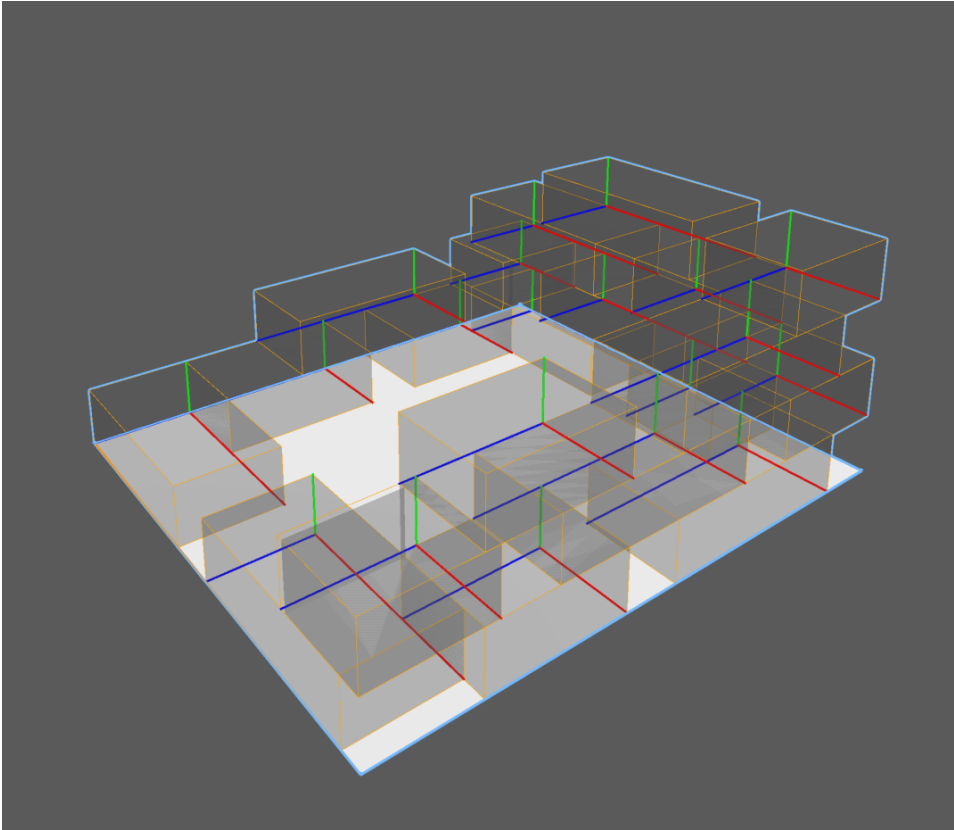
- The circular button toggles the visibility of the component's attributes in the **Inspector** window where they are hidden by default.





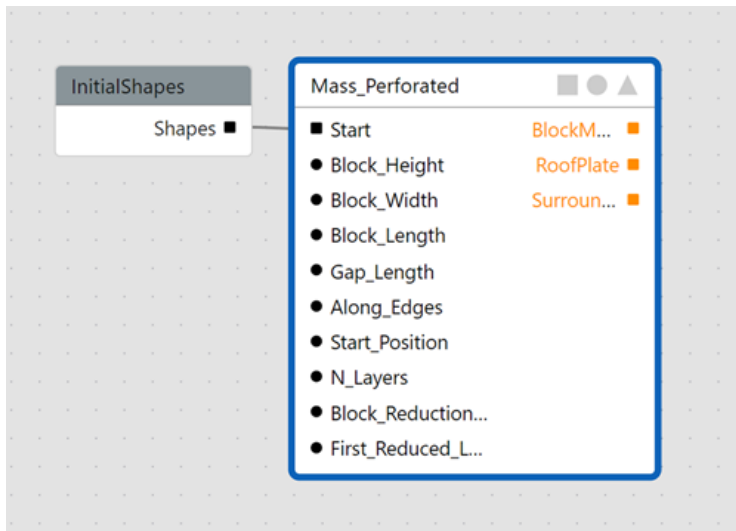
- The triangular button enables visualization of the input shapes' scopes on the lead selection in the **Viewport** window.

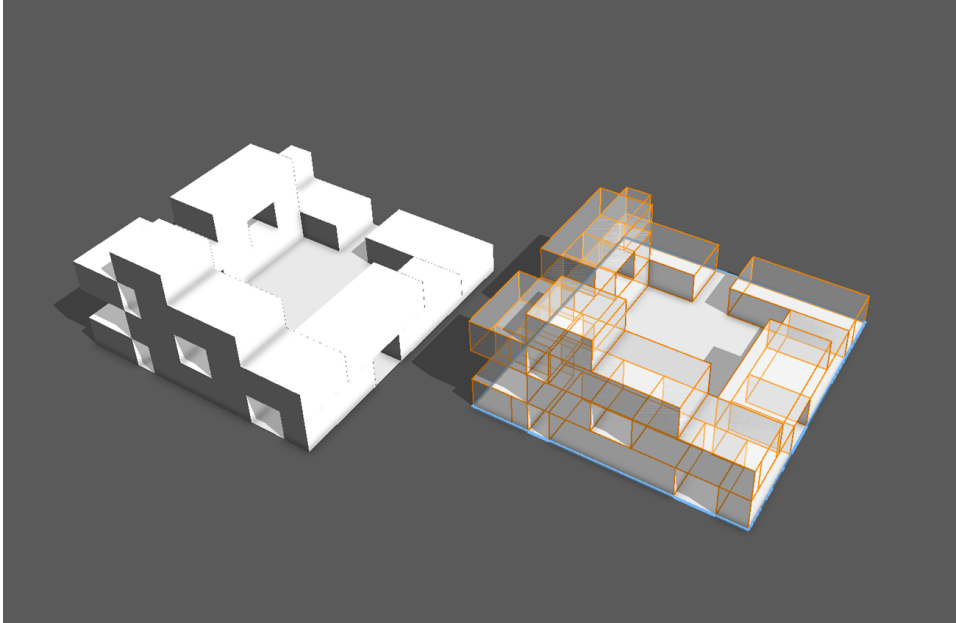




### Visualization

You can select a component node to highlight its extension points on the lead selection in the **Viewport** window:





It is also possible to highlight a single extension point by hovering over the corresponding output slot. This feature is useful for visually identifying specific extension points in the **Viewport** window.

## Built-in nodes

In the **Visual CGA Editor** window, you can use built-in nodes such as shape, attribute, or value nodes to create VCGA designs.

### *Shape nodes*

The shape node is the root of the visual graph and it cannot be deleted or cloned. It represents the shapes that the rule is assigned to (in the .cej scene).

### *Input nodes*

You can use the following distinct types of input nodes:

- Value nodes:
  - These nodes exclusively provide a value.
- Attribute nodes:
  - These nodes define the attributes of the VCGA design and are displayed in the **Inspector** window.

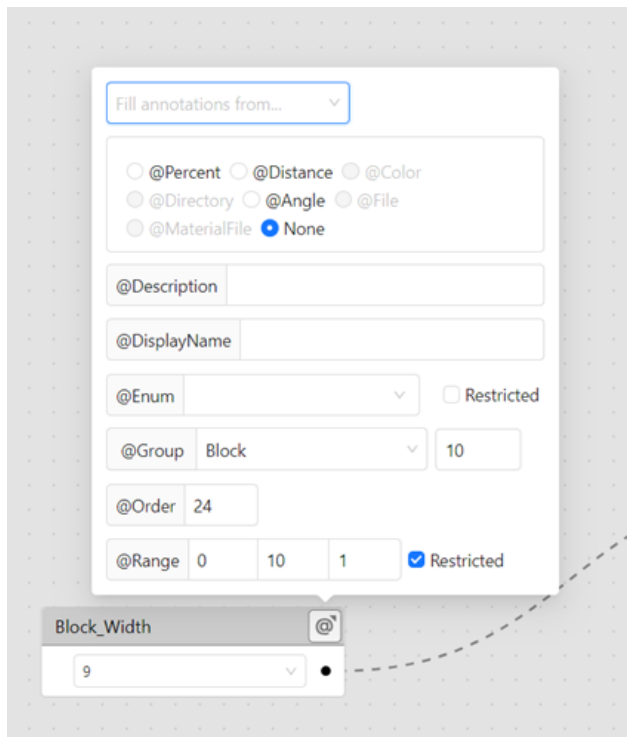
### **Note:**

Attribute nodes can be conveniently created automatically by dragging a connection from an attribute slot and dropping it into the canvas. This action opens up the on-drop context menu, offering the **Link attribute in the Inspector** option.

- They can either adopt the default attribute value or specify a new value.

**Note:**

- The default values depend on the connected initial shapes and are evaluated at runtime. Therefore, they cannot be displayed in the attribute nodes. However, you can see the default value for a specific shape in the **Rules** section of the **Inspector** window.
- When attribute nodes adopt a default attribute value, they also inherit its annotations. Conversely, when attributes nodes specify a new value, they allow for the configuration of custom annotations through the annotation editor.








*Additional built-in nodes*

You can also use the following additional built-in nodes:

- Extension nodes:
  - These nodes specify the extension points of the VCGA design.
  - They can either follow a default behavior implementation or define a new leaf shape.
- Switch nodes:
  - Switch nodes enable the activation and deactivation of different branches within the visual graph through custom properties.
- Conditional flow control nodes:
  - These nodes activate or deactivate different branches of the visual graph through conditional statements on attributes.

**Visual CGA Editor toolbar**

The **Visual CGA Editor** toolbar includes the following tools:

<b>Select and frame</b> 	Select and frame objects in the <b>Viewport</b> window with the current VCGA design assigned.
<b>Assign</b> 	Assign the current VCGA design to the selected objects in the <b>Viewport</b> window.
<b>Export</b> 	Export the current Visual CGA design as CGA rule file.
<b>Frame</b> 	Frame the selected nodes (F), or frame all if nothing is selected.
<b>Zoom in</b> 	Zoom / Dolly in.
<b>Zoom out</b> 	Zoom / Dolly out.
<b>View mini-map</b> 	Toggle the mini-map in the editor.

### Manage rule errors

Errors in the VCGA file are detected automatically and are marked differently based on the type of error.

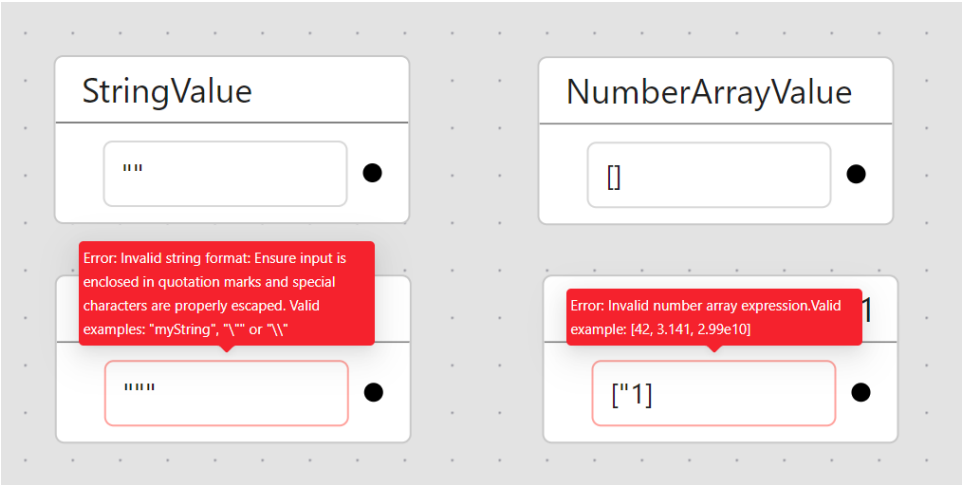
### Component node errors

Component nodes can show the following possible error states:

- **Error: the file \*\*\* has changed**—Reload the node to resolve: This error indicates that the CGA/VCGA file linked to the component node has been modified, altering its signature (e.g., attributes or extensions have been introduced, modified, or removed). The error can be fixed using the **Reload Node** option in the context menu.
- **Error: the file \*\*\* contains errors**—This error indicates that the CGA/VCGA file linked to the component node contains errors. This can be fixed by manually fixing the errors inside the respective CGA or VCGA file.
- **Error: the file \*\*\* could not be found**—This error occurs when the CGA/VCGA file linked to the component node has been deleted or renamed. It can be fixed by selecting **Change Component File** from the context menu.

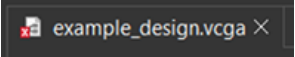
### Input node errors

Input nodes validate the entered values. If validation fails, an error is displayed based on the node's type. An explanatory pop-up provides details about why the validation failed.



## VCGA file errors

When there are compilation errors within the VCGA file, a small red cross appears in the window tab:



More detailed information about the error can be found in the **Problems** window (**Window > Problems**). Errors must be resolved before applying the rules. It is not possible to generate models if the assigned rule file contains errors.

## Shortcuts


In the **Visual CGA Editor** window, you can use navigation and keyboard shortcuts to work more efficiently.

You can configure the global navigation mouse scheme in the [CityEngine preferences](#). This allows customizing nodes selection, panning (scrolling), and zooming shortcuts.

You can use the following keyboard shortcuts when working in the **Visual CGA Editor** window:

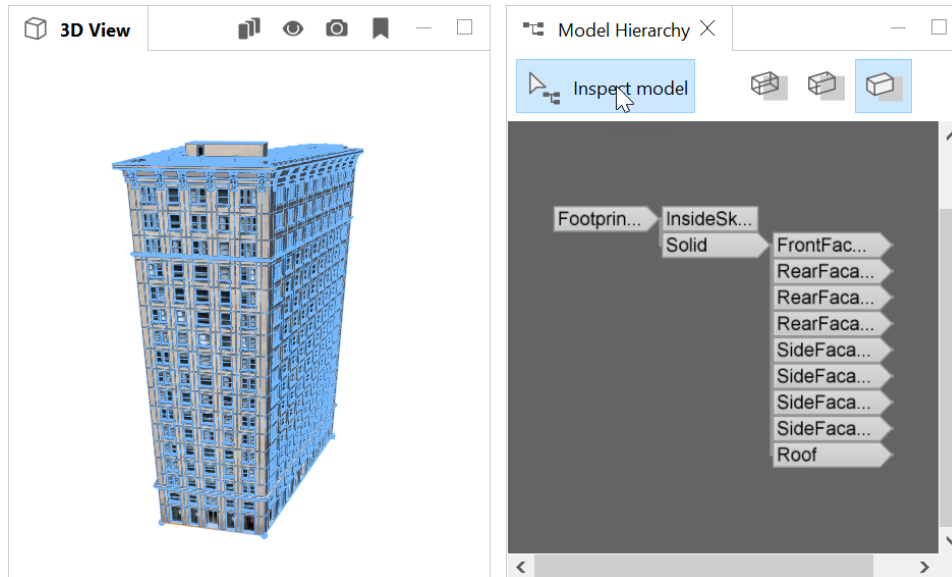
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+A	Select all nodes
Ctrl+C	Copy selection to clipboard
Ctrl+V	Paste clipboard to canvas
F	Frame the selected nodes, or frame all if nothing is selected
Space	Maximize or minimize the <b>Visual CGA Editor</b> window

# Model Hierarchy

The shape tree of a generated model can be interactively explored. To open the **Model Hierarchy** window , select **Window > Model Hierarchy** from the menu.

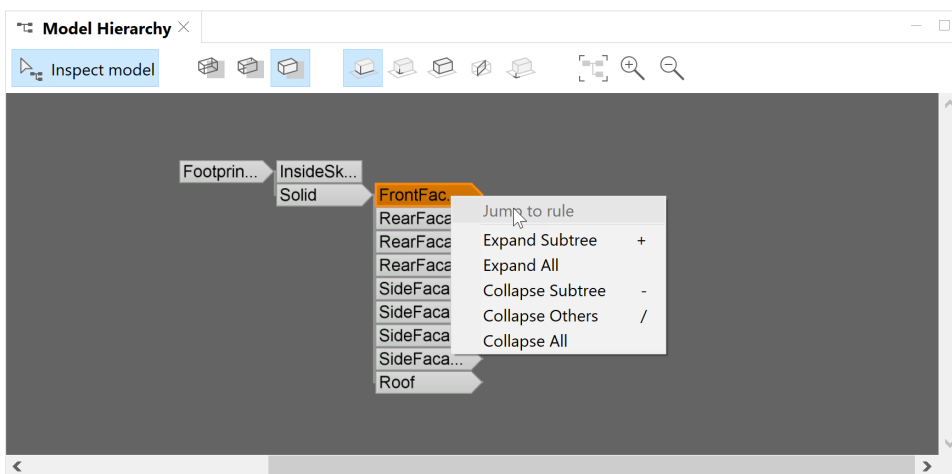
## Inspect model

To generate a model, select it and click the **Inspect Model** tool on the toolbar of the **Model Hierarchy** window :



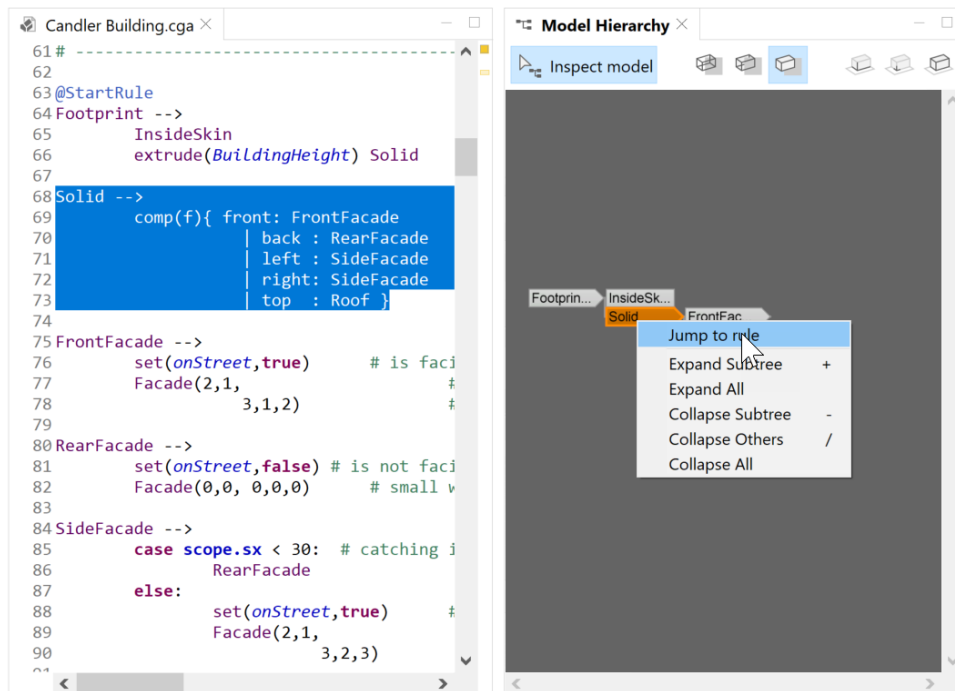
A generated model switched to the edit model; the hierarchy appears in the Model Hierarchy Explorer.

The model hierarchy (or shape tree) of a model can now be expanded and traversed by clicking nodes or using the context menu (right-click) on a tree node:



The right-click menu expands and collapses model hierarchy nodes.

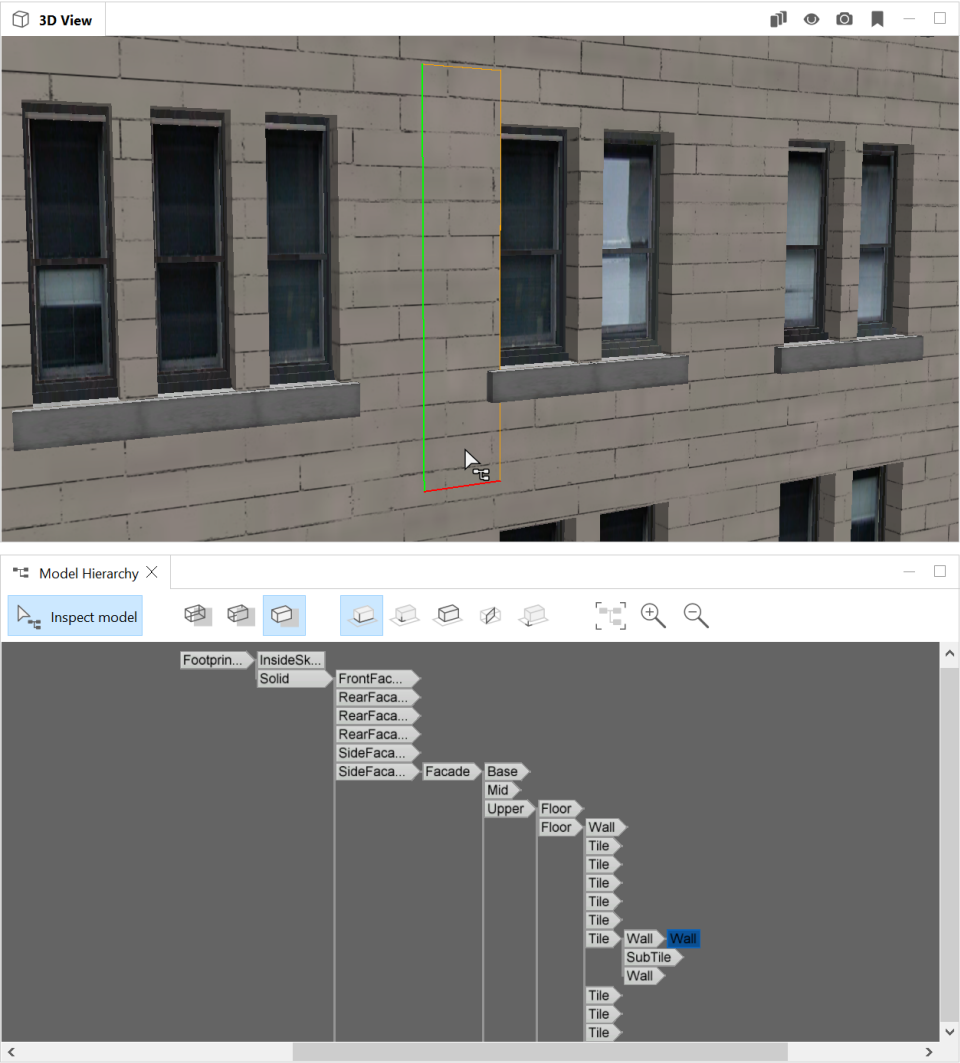
The shape tree of a specific building is defined by the associated rule file and the initial shape. Double-click a node or choose **Jump to rule** in the context menu to jump to the respective rule in the rule file.



Part of a CGA rule file and the hierarchy of the generated model are shown.


The pictures above show snippets from the Candler rule file. The rule file and the model hierarchy structures match (for example, Footprint → Solid → FrontFacade/RearFacade/SideFacade/Roof).

Each node of the model hierarchy can be expanded and collapsed using the right-click menu. Moreover, each shape node can be selected or deselected with a left-click (use the **Ctrl** modifier to select more than one node). A number of special rendering modes are available to visualize the attributes of the selected shapes. It is also possible to directly click into the model in the 3D Viewport:



Model selected in Viewport and shown in Model Hierarchy.

## Toolbar


The following is a list of options in the **Model Hierarchy** window .

<b>Inspect model</b>	Activate inspect model mode.
<b>Transparent model</b>	Set the drawing mode of the selected model to transparent.
<b>Translucent model</b>	Set the drawing mode of the selected model to translucent.
<b>Opaque model</b>	Set the drawing mode of the selected model to opaque.
<b>Show scope</b>	The selected shape nodes' scopes are drawn (x-axis red, y-axis green, z-axis blue, other scope edges orange).
<b>Show pivot</b>	The selected shape nodes' pivots are drawn (x-axis red, y-axis green, z-axis blue).
<b>Show geometry</b>	The selected shape nodes' geometry is drawn.
<b>Show trim planes</b>	The selected shape nodes' trim planes are drawn.

Show origin of model	The model's origin is drawn (x-axis red, y-axis green, z-axis blue).
Frame selection	The selected nodes in the model hierarchy view are framed.
Zoom in	Zoom in to the shape tree structure in the model hierarchy view.
Zoom out	Zoom out of the shape tree structure in the model hierarchy view.

The size and line width of scopes, pivots, and trim planes can be set in **Edit > Preferences > General > Procedural Runtime > Display Options**.




# Facade Wizard

CityEngine features the **Facade Wizard**, a streamlined interactive tool for the rapid creation of textured 3D facades. The tool outputs CGA code that can be used within CityEngine as any other piece of CGA code. This document provides an overview of the typical facade creation workflow and explains the individual steps. You can open the **Facade Wizard** window  by selecting **Window > Facade Wizard** in the main menu.

See [Tutorial 13: Facade wizard](#) for more in-depth information.

## Basic workflow

The basic workflow of the facade wizard is as follows:

1. Open the **Facade Wizard** window .
2. Select a shape or a single face of a multiple-face shape in your scene. Alternatively, select a facade texture image in your project using the **Navigator** window .
3. Load the selected shape or image into the **Facade Wizard** window  by clicking the **New Facade** button in the **Facade Wizard** toolbar.
4. Click the **New Facade from Image** button to load an image from the file browser.
5. Use the standard CityEngine 3D navigation controls and shortcuts to navigate when editing the facade (in particular, press **A** to frame the whole facade, and press **Z** to position the camera in front).
6. Add vertical and horizontal splits by clicking the **Y Split** and **X Split** options, respectively.
7. Add vertical and horizontal repeats by clicking the **Y Repeat** and **X Repeat** options, respectively.
8. Move the mouse over the facade or individual regions. Lines indicate where the split or repeats will be added.





### Tip:

Use the left and right arrow keys to switch between the different tools.


9. Move existing splits or modify repeats by moving the mouse near the split or repeat you want to edit.
10. Drag the mouse to move or adjust the splits or repeats.
11. Adjust the depth of the final regions by using the **Z-adjust** tool.
12. Save the facade by clicking **Save**.  
A file browser will ask you to specify a CGA rule file where the rules will be written. If your facade was loaded from a shape, the shape's or face's rule file and the start rule will be set, and you can select and generate the model.




### Note:

Currently, the **Facade Wizard** window  creates CGA code one way only, in other words, you cannot load a facade with associated rules. However, the created CGA code can be edited using the **CGA Editor** window .

## Additional considerations

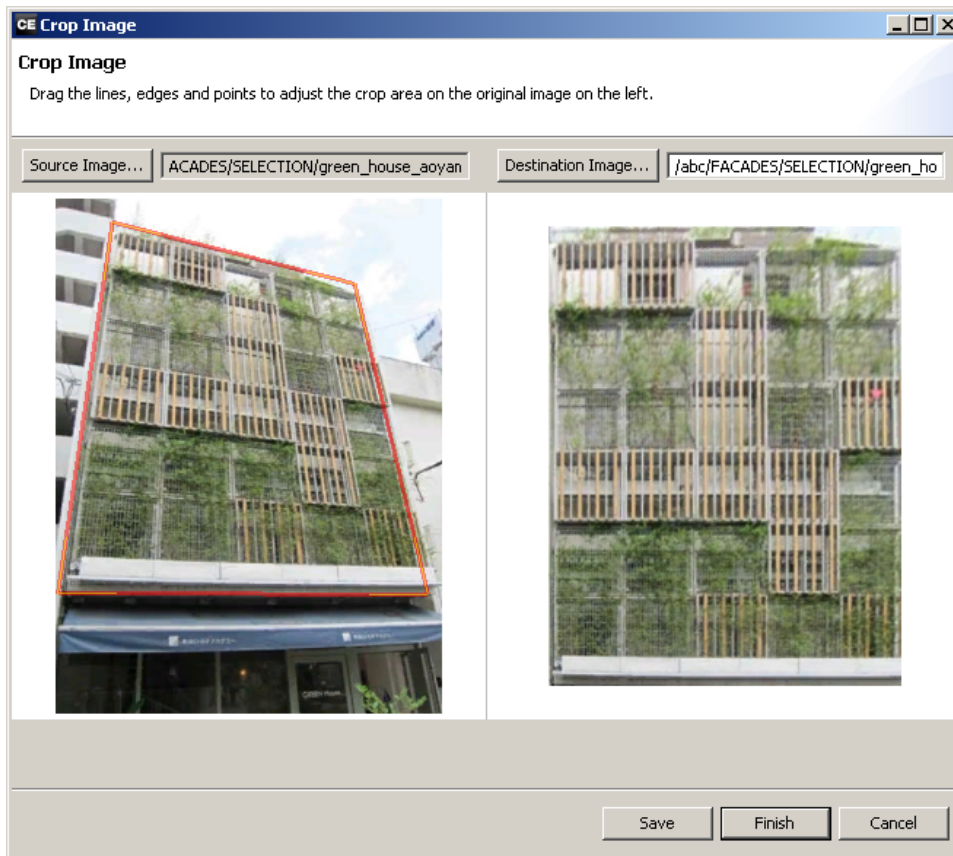
Flexible vs. rigid splits	Normally, if you have multiple splits along an axis (X or Y), the Facade Wizard will assign flexible splits automatically so a created facade can adapt to different shapes. However, in some cases, it is necessary to manually define split types. Move with the mouse over a region and press the up or down arrow keys to change the split model: A bold yellow line indicates a rigid split, a dotted yellow line indicates a flexible split. No line indicates automatic mode.
Facade dimensions	When loading images, a dialog box will ask for an approximate initial width. When loading a shape, the dimensions will be taken from the actual shape size.  <b>Note:</b> A more precise width or height can be specified at any time using the context menu's <b>Set Region Width / Height</b> entry.
Select region	Using the context menu's <b>Select Region</b> option, you can specify which texture area will be used for repeats. The selected region appears differently colored than the other repeating regions.
Snapping	The Facade Wizard maintains a history of all previous X and Y splits. Press <b>Shift</b> when adding splits, and the split will snap to previous locations. The snapping history is kept when loading new facades, making snapping useful when, for instance, creating multiple facades of the same mass volume. Use the context menu to clear the snap history.

## Crop Image tool

The **Crop Image** tool provides an intuitive and effective means for the preparation of facade textures; perspective correction and region selection are done in one step. Starting with automatically detected facade bounds, select the facade or an element of interest. You adjusted visually while watching the result in real time. To open the tool, select the file and click **Shapes > Crop Image** from the main menu or **Crop Image** from the context menu of an image file in the **Navigator** window .

### Interface

The **Crop Image** tool is divided into left and right:



*Crop Image wizard*

- On the left, the source image is loaded with the **Source Image** option. The viewport displays the original image with the perspective frame on top. That frame may be adjusted to crop the facade or element of choice.
- The right viewport shows the corrected frame selection from the left. Adjust and click **Finish** to save the cropped file to the **Destination Image** path.

### *Perspective frame manipulation*

You can manipulate the perspective frame the following ways:

- Corners—Drag the corners directly into position.
- Corner handles: —Adjust the horizontal or the vertical side line of a corner while keeping the other side line as is.
- Side line—Move the side line perspectively in parallel.

### *Zoom and pan*




In both viewports, you can do the following:

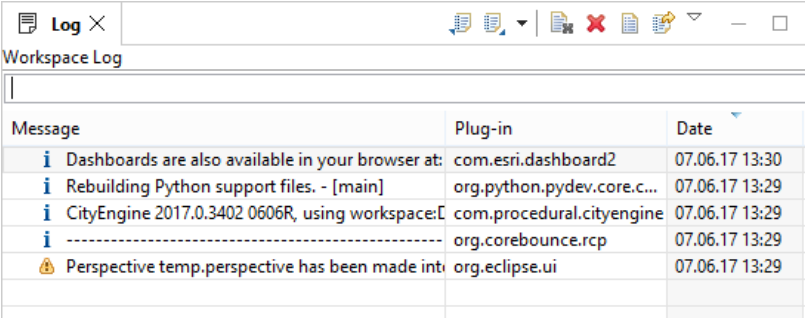
- Zoom—Use the mouse wheel.
- Pan—Move the mouse while pressing **Alt** and the middle mouse button.






## Status windows

CityEngine includes additional windows to provide information about a scene, such as progress status of operations, log records, and errors.

### Log

The **Log** window  shows the log records of CityEngine. You can open the **Log** window  by selecting **Window** > **Log** from the main menu. Log records are created by various parts of CityEngine and range from informational messages to severe internal error conditions (such as out of memory). The properties and values of each log record are shown in this view. The **Log** window  view is especially useful in tracking down strange or erroneous behavior.




Message	Plug-in	Date
 Dashboards are also available in your browser at:	com.esri.dashboard2	07.06.17 13:30
 Rebuilding Python support files. - [main]	org.python.pydev.core.c...	07.06.17 13:29
 CityEngine 2017.0.3402 0606R, using workspace:	com.procedural.cityengine	07.06.17 13:29
 -----	org.corebounce.rcp	07.06.17 13:29
 Perspective temp.perspective has been made int	org.eclipse.ui	07.06.17 13:29

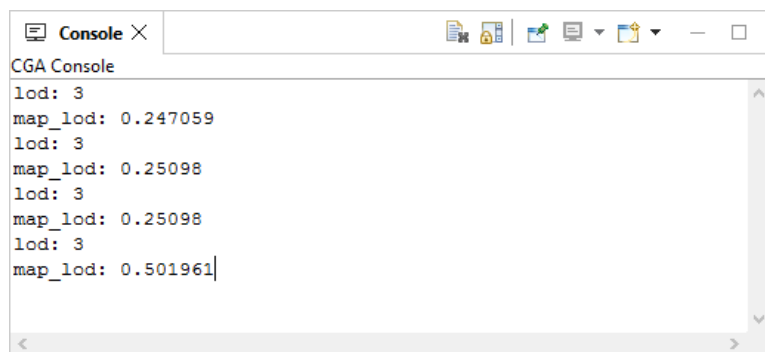
Log window

The meaning of the severity color is as follows:

- None: Information
- Yellow: Warning
- Red: Error

### Console

You can open the **Console** window  by selecting **Window** > **Console** from the main menu. The console window contains various consoles used when working with CityEngine. The top toolbar button switches between consoles (if available).



Console window

### CGA console

If a CGA command produces textual output (such as the CGA print command), this output will be shown in the CGA console. This console is available once a CGA print output is produced.




## Python output console

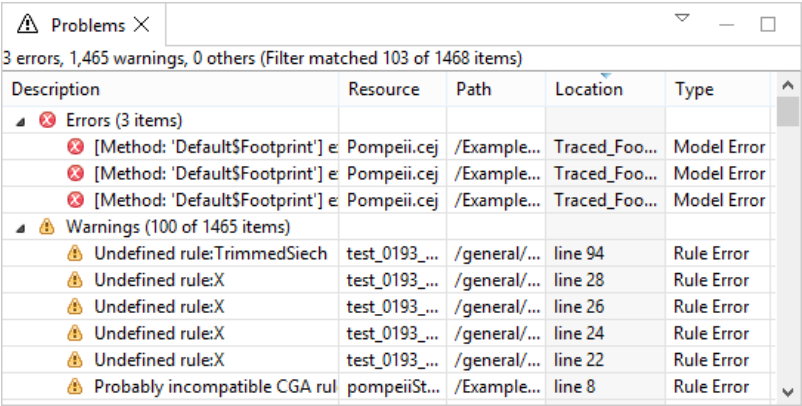
The Python output console is the default output console for Python statements that use a `print()` command. This console is available once a Python print output is produced.

## Python command console

The Python command console to enter Python commands. This console needs to be opened in the top toolbar of the console window.


## Problems

The **Problems** window  displays errors during CGA shape grammar editing. Errors and warnings are passed up from the CGA compiler. The **Problems** view  lists the error, file name, and folder. If you select an error, the associated file will open in the **CGA Editor** and the pointer will display the line where the error was encountered. You can open the CGA **Problems** window  by selecting **Window > Problems** from the main menu.





Description	Resource	Path	Location	Type
3 errors, 1,465 warnings, 0 others (Filter matched 103 of 1468 items)				
Errors (3 items)				
[Method: 'Default\$Footprint'] e	Pompeii.cej	/Example...	Traced_Foo...	Model Error
[Method: 'Default\$Footprint'] e	Pompeii.cej	/Example...	Traced_Foo...	Model Error
[Method: 'Default\$Footprint'] e	Pompeii.cej	/Example...	Traced_Foo...	Model Error
Warnings (100 of 1465 items)				
Undefined rule:TrimmedSiech	test_0193_...	/general/...	line 94	Rule Error
Undefined rule:X	test_0193_...	/general/...	line 28	Rule Error
Undefined rule:X	test_0193_...	/general/...	line 26	Rule Error
Undefined rule:X	test_0193_...	/general/...	line 24	Rule Error
Undefined rule:X	test_0193_...	/general/...	line 22	Rule Error
Probably incompatible CGA rul	pompeiiSt...	/Example...	line 8	Rule Error

Problems window

In the views settings of the **Problems** window , change the grouping to **Type** to have problems sorted by **Model Errors** and **Rule Errors**.

## Progress

The **Progress** window  shows the progress status of long-running CityEngine operations. You can monitor the progress in the progress view as well as cancel an operation by clicking the red **Stop** button on the right side of the operation. You can open the progress view by selecting **Window > Progress** from the main menu.



Processing 4,612 shapes
Generating occluders...

Progress window

### Note:

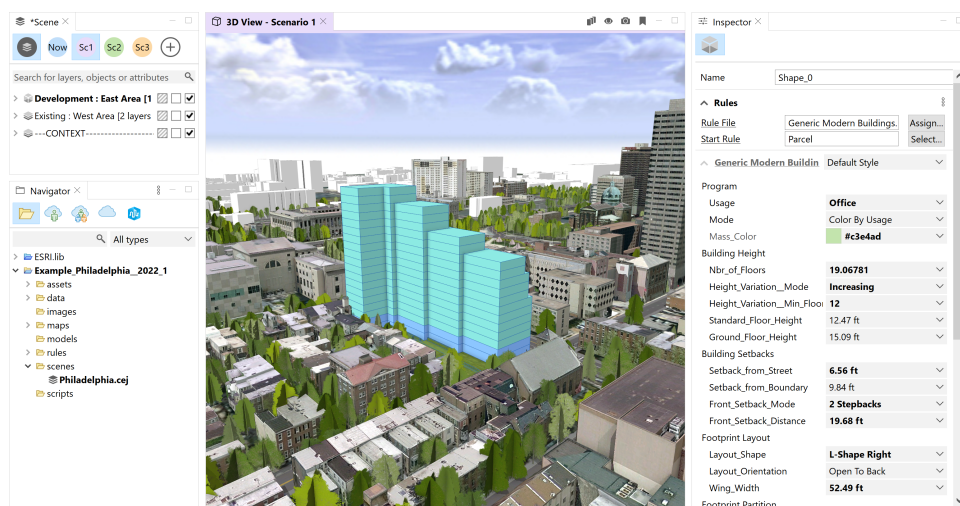
You can cancel all pending model generation by pressing the `Esc` key or choosing **Cancel** from the main toolbar.

# Layouts

You can switch between different window layouts to maximize the screen space available. Click **Window > Layout** in the main menu to see the available layouts.

## Default

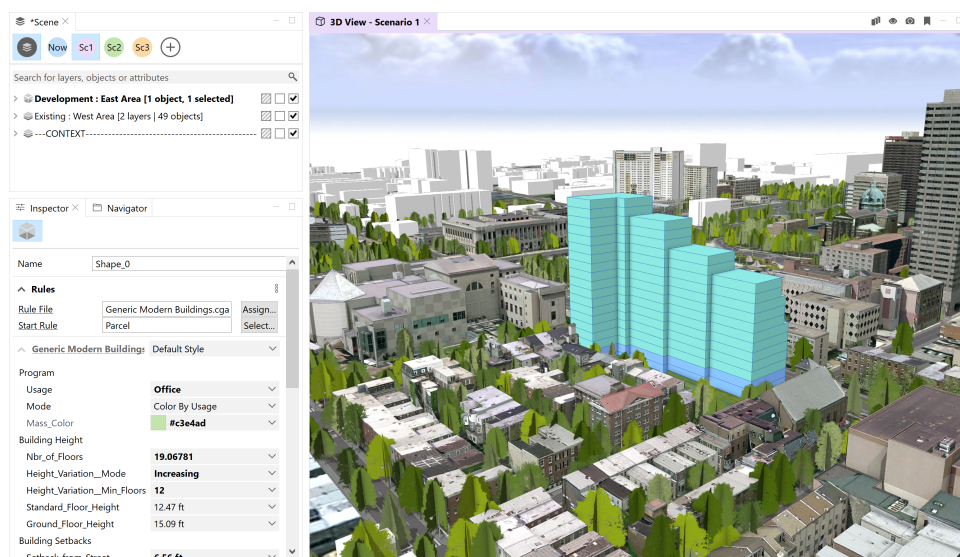
The **Default** layout has one big perspective view with editing options open to allow for scene overview and management tasks such as import and export.



The Default layout is shown.

## Compact

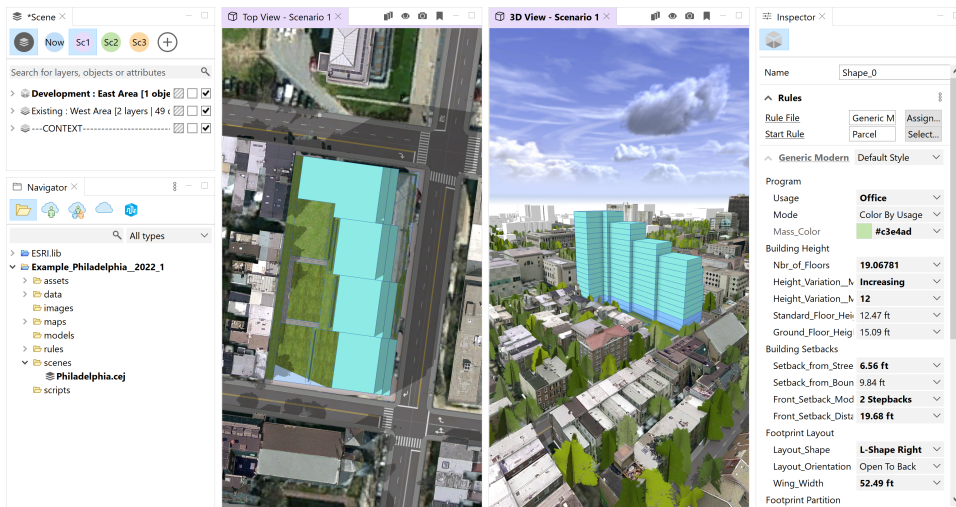
The **Compact** layout has one big perspective view and some editing options open. This makes it suitable for presentations.



Compact layout

## Top & 3D View

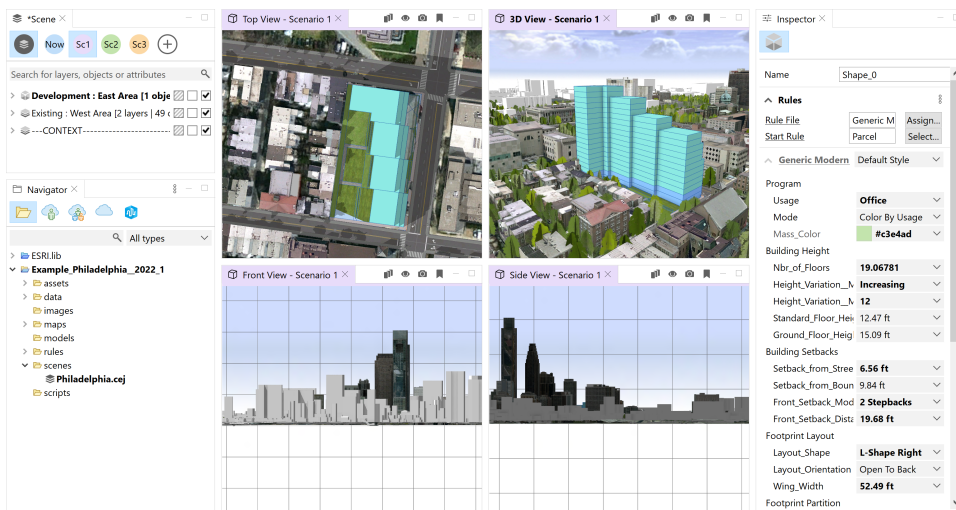
The **Top & 3D View** layout has a perspective view and a top view with space for editing options on both sides.



Top &amp; 3D View layout

## Top, Front, Side & 3D View

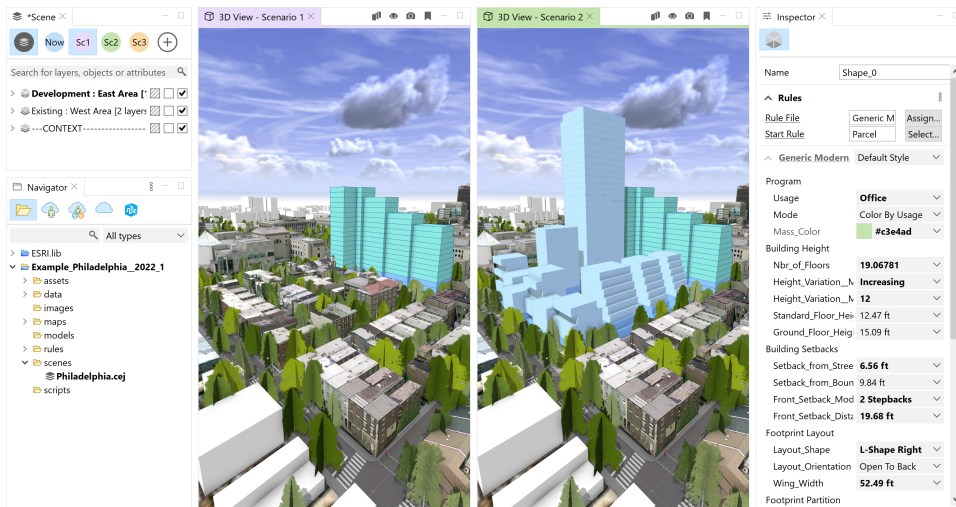
The **Top, Front, Side & 3D View** layout is the classical CAD layout with a perspective, top, front, and side view.



Top, Front, Side &amp; 3D View

## 2 Scenarios Side-by-side

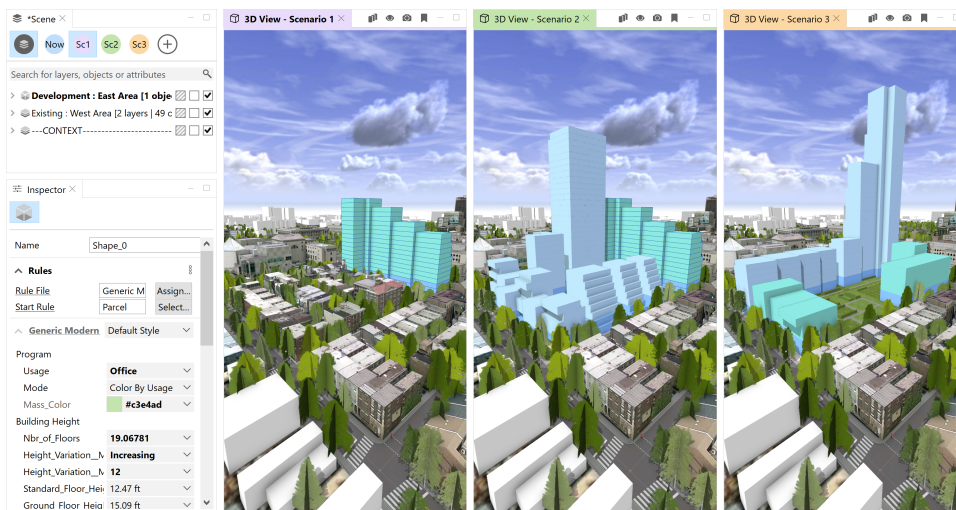
The **2 Scenarios Side-by-side** layout has two perspective views with different scenarios assigned to it. This layout is especially suitable for comparing scenarios.



2 Scenarios Side-by-side

## 3 Scenarios Side-by-side

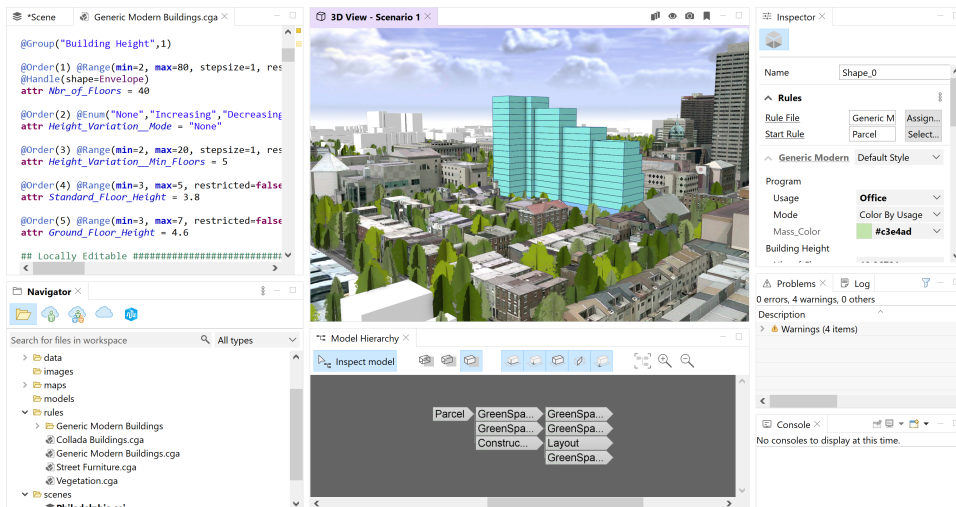
The **3 Scenarios Side-by-side** layout provides three perspective views for different scenarios.



3 Scenarios Side-by-side

## Rule Programming

The **Rule Programming** layout has one perspective view and leaves more area for the **CGA Editor**. This layout is especially suited for editing rules and monitoring the outcome.



Rule Programming layout

## Multiple viewport windows

You can open as many viewports as you require for your modeling task. You can create multiple viewports to examine your 3D data from different angles, or to view multiple scenarios side by side. A viewport is created by selecting **Window > New Viewport** from the main menu.

The following lists the default viewports available:

- 3D View
- Top View
- Front View
- Side View

Additional viewport types can be created and managed using the **Window > New Camera** and **Manage Cameras** options. [Camera](#) changes and render settings affect all viewports of the same type. To create two 3D viewports using different rendering modes, you need to create a new viewport type.

# Projects

# Projects

A typical CityEngine project consists of a variety of files: scenes, rules, and any other data related to the project. Keeping assets, rules, and scenes in predefined locations, or workspaces, helps you and your collaborators.

## Workspace

When you start CityEngine for the first time, a CityEngine default workspace is created for you automatically in your home directory. The workspace is basically a folder in your file system where all your projects are stored. This allows you to use resources such as 3D models or rule files across multiple projects.

- You can create as many workspaces as you like, but you can open only one workspace at a time.
- Each newly created workspace gets its own copy of [ESRI.lib](#). It contains assets such as vegetation, street furniture, or building rules that you can use in your projects.
- You have access to the current workspace and its projects through the [Navigator](#).
- It is a good practice to add and delete assets, and import and export projects in your workspace using the [Navigator](#). Additionally, you can update assets with other tools.
- If a file added with the file system doesn't show up in the [Navigator](#), you can refresh the workspace manually by pressing **F5** or by clicking **File > Refresh Workspace**.
- Some users work with one workspace only; others have separate workspaces for each of their clients. A good rule of thumb is to keep projects in the same workspace in case you plan to share assets between them.



### Tip:

To get a step-by-step introduction on how to manage projects in CityEngine, see [Tutorial 1: Essential skills](#).

## Switch to an existing workspace

To switch to an existing workspace, do the following:

1. Click **File > Switch Workspace**.
2. If the workspace is not listed, go to **Other** and browse to the workspace in your file system.



### Note:

Switching to another workspace automatically restarts CityEngine.

## Create a workspace

To create a workspace, do the following:

1. Click **File > Switch Workspace > Other** in the main menu.
2. Set the path and the folder name for the workspace.

Under **Copy Settings**, you can select the following settings:

- **Workspace Layout**—Inherit opened views, their sizes, and selected perspectives from the current workspace.
- **Working Sets**—Inherit user-defined working sets from the current workspace.

## Specify workspace on command line at startup

The `-data` argument can be added on the command line to specify the startup workspace directory for CityEngine. Its value can point to a new or existing workspace. Here is an example, assuming CityEngine.exe is on the PATH:

```
CityEngine -data C:\Users\MyUser\AppData\Local\Temp\MyCeWorkspace
```

This creates a new MyCeWorkspace directory in the Temp directory. The new workspace is automatically populated with the [ESRI.lib](#) project. Set the CITYENGINE\_LIB\_MINIMAL environment variable to 1 to reduce the disk space consumed by ESRI.lib.

## Work with assets

To import assets into a project, drag them from your system file browser to the desired folder in the [Navigator](#). You will be prompted to link (create a reference the source only) or copy it into the workspace.

To edit an asset, right-click the asset in the [Navigator](#) and select **Default Editor** or **Other**. Alternatively, you can select **Show in File Manager** to access the file over the file browser of your operating system.



### Note:

Alternatively, you can assign your preferred application to a file type by selecting **Open With > Other** from the [Navigator](#) context menu. Once the assignment has been made, you can open files of that type in the preferred application by right-clicking a file and selecting **Open** in the [Navigator](#) context menu. Choose **External programs** and select the preferred program from the list.

# Manage projects

In CityEngine, all the files associated with a given scene or group of scenes are organized in a project. A project is a collection of folders for the following file types:

- **assets**—This is the default location for 3D models and textures that are used by [CGA](#).
- **data**—Use this folder to store mass models (as grouped `.obj` or `.dxf` files), for example. Dashboard configuration files for each scene and elevation delta files for each layer are also stored in this folder. If you have other project-related resources such as artwork and sketches, place them into the data folder as well.
- **images**—Additional imagery such as **Viewport** snapshots are stored here.
- **maps**—This folder contains the image maps used by the map layers. For example, a height map or obstacle map is stored here. Downloaded data from [Get Map Data](#) is stored in this folder.
- **models**—This folder is the default location for exported 3D models.
- **rules**—This folder contains CGA shape grammar rule files (`.cga`). Double-click a `.cga` file to open the file in the **CGA Editor**.
- **scenes**—CityEngine scene files (`.cej`) are stored here. Double-click a scene file to close the current scene (if any) and open the newly selected scene.
- **scripts**—This folder contains Python scripts (`.py`).

## Caution:

It is not recommended that you delete or rename the default folders mentioned above. You can add folders on the root level or put your additional material in the data folder.

## Create a project

You can create projects, scenes, folders, and files using the following options in CityEngine:

- Click **File > New**.
- Right-click the **Navigator** and select **New**.
- Press `Ctrl+N`.

After the wizard appears, do the following:

1. Click **CityEngine project** inside the CityEngine folder.
2. Click **Next**.
3. Enter the project name.
4. Click **Use default location** or click **Browse** to select a different location.
5. Click **Finish**.  
The **Navigator** displays the new project.

## Import an existing project into a workspace

For unzipped or zipped projects, do the following:

1. Click **File** > **Import/Link Project Folder into Workspace** or **Import Zipped Project Workspace**.
2. Select root folder of project to import.
3. Click **Finish**.

## Import options

You can choose from the following options for importing:

- **Select root directory**—Select the root directory in the file system to start scanning for projects to import. Type the full path or browse to select the path in the file system.
- **Select archive file**—Select an archive (.zip) file to scan for projects to import. Type the full path or browse to select the archive in the file system.
- **Copy projects into workspace**—When selected, this will cause the imported project to be copied into the current workspace. If this option is not selected, the project content will be linked into the workspace. This option is not available for zipped projects.

### **Note:**

It is recommended that you always turn on the **Copy projects into workspace** option. Exceptions are when you have limited storage space or for collaborative work environments.

- **Add project to working sets**—Add an imported project to the user-defined working sets.

## Create a scene

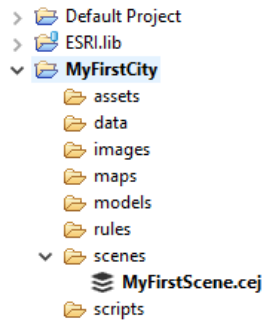
To create a scene in CityEngine, do one of the following:

- Click **File** > **New** in the main menu.
- Right-click the **Navigator** and select **New**.
- Press **Ctrl+N**.

After the wizard appears, do the following:

1. Click **CityEngine scene** inside the CityEngine folder.
2. Click **Next**.
3. Select the **project folder**.
4. Enter a file name.
5. Use the default coordinate system or choose one by clicking **Choose**.
6. Click **Finish**.

The scene is added to the project in **Navigator**.



*The scene is added to the project in Navigator.*



**Tip:**

Alternatively, right-click the project folder in **Navigator** and select **New > CityEngine scene**.

# Save a file

After making edits to your scene or CGA rule file, you can save the changes. When you save in CityEngine, the window that is active saves the content. For example, to save a scene, the [Viewport](#) or the [Scene Editor](#) window must be active.

## Save a scene

1. Ensure that the **Viewport** window is active.
2. Click **File** > **Save**.

## Save a CGA rule file

1. Ensure that the [CGA Editor](#) window is active.
2. To save the CGA rule file, do one of the following:
  - Click **File** > **Save** in the main menu.
  - Right-click in the **CGA Editor** and select **Save**.
  - Press **Ctrl+S**.



### Tip:

To make changes to a library file, it is recommended that you make a copy of the `ESRI.lib` CGA rule file.

# Import a file

CityEngine includes a variety of import wizards to bring files into a project.

## Import local files into a project

To import files into a project, do the following:

1. Click **File > Import**.
2. Click **Files into Existing Project > File System**.
3. Click **Next**.
  - **From directory**—Type or browse to select the directory containing the files you want to import. Recent directories from which files have been imported are shown on the **From directory** field's combo box.
  - In the pane on the right, check the files you want to import. Check a folder in the left pane to import its entire contents into the workspace. A black square in the middle of the check box (instead of a check mark) indicates that only some of the files in the folder will be imported into the workspace.
  - **Filter Types**—Used to filter the types of files that will be imported.
  - **Into folder**—Should already be filled in with the name of the project you are working with, but it can be changed with **Browse**.
  - **Options**—The following choices are available:
    - **Overwrite existing resources without warning**—Determines whether importing a resource should silently overwrite a resource that already exists in the workspace. If this option is off, you will be prompted before a given resource is overwritten, in which case you can overwrite the resource, skip it, or cancel the import.
    - **Create top-level folder**—Creates a folder inside the **Into folder** with the top folder name in the left pane. Otherwise, only the selected file folders are created in the **Into folder**.
4. Click **Finish** when done.  
The selected files and folders appear in the **Navigator**.

## Import an archive file into a project

1. Click **File > Import**.
2. Click **Files into Existing Project > Archive File**.  
In the pane on the right, check the files you want to import. Check a folder in the left pane to import its entire contents into the workspace. A black square in the middle of the check box (instead of a check mark) indicates that only some of the files in the folder will be imported into the workspace.
3. Click **Next**.

The following options are available for an archive file import:

- **Archive File**—The file from which to import. Type the full path or click **Browse** to select the path in the file system.
- **Filter Types**—Select which file types to import. Use this to restrict the import to only certain file types.
- **Into Folder**—The folder into which the resources are imported. Type the path or click **Browse** to select a path in the workspace.

- **Overwrite existing resources without warning**—Determines whether importing a resource should silently overwrite a resource that already exists in the workspace. If this option is off, you will be prompted before a given resource is overwritten, in which case you can overwrite the resource, skip it, or cancel the import.

# Export a project

CityEngine provides means for exchanging projects in collaborative environments. The most efficient way to exchange project data is to create archived projects. An archived project contains all project-specific settings, scenes, rules, and assets.

## Export a project to an archive file

To export a project to an archive file, do the following:

1. Click **File > Export**.
2. Click **General > Archive File**.
3. Click **Next**.

You have the following options for exporting a project as an archive file:

- **Select resources to export**—The project (and resources within that project) to export to the archive.
- **Filter Types**—Select which file types to export. Use this to restrict the export to only certain file types.
- **To Archive File**—The path and name of the archive file into which the resources will be exported. Type the path, select a previous path from the drop-down list, or click **Browse** to select a path and file name in the file system.
- **Save in zip format**—Exports the file in zip format or **Save in tar format** exports the file in tar format.
- **Compress the contents of the file**—Compresses the contents (resources selected to be exported) in the archive that is created.
- **Create directory structure for files**—Creates a hierarchical folder structure in the file system as it exists in the workspace.
- **Create only selected directories**—Creates a hierarchical structure in the file system only for selected folders.

## Export a project to the local file system

1. Click **File > Export**.
2. Click **General > File System**.
3. Click **Next**.

You have the following options for exporting a project as a local file:

- **Select resources to export**—The project (and resources within that project) to export to the file system.
- **Filter Types**—Select which file types to export. Use this to restrict the export to only certain file types.
- **To Directory**—The directory on the file system into which the resources will be exported. Type the path, select a previous export path from the drop-down list, or click **Browse** to select a path.
- **Overwrite existing files without warning**—Determines whether exporting a resource should silently overwrite a resource that already exists in the file system. If this option is off, you will be prompted before a given file is overwritten, in which case you can overwrite the file, skip it, or cancel the export.
- **Create directory structure for files**—Creates a hierarchy (folder) structure in the file system as it exists in the workspace.



- **Create only selected directories**—Creates a hierarchy (folder) structure in the file system only for selected folders.






# Scenes

# Scenes

In this section, you learn how to [navigate](#) in a 3D scene, and [select](#) and [transform](#) objects. Additionally, you learn about [cameras](#), [bookmarks](#), [georeferencing](#), and [scene light and panorama](#).

## 3D navigation essentials

To navigate in the 3D **Viewport** window , you can use the navigation tools to track, tumble, dolly, or look around. You can select the appropriate tool on the toolbar and click and drag the mouse in the **Viewport** window . Alternatively, these actions can be applied any time using a modifier. The following table describes navigation actions:


Frame tool  + click	Frame the selection (or the whole scene if the selection is empty) in all 3D viewports.
Pan tool  + click / Press <b>Alt+scroll wheel</b>	Pan/track the camera horizontally or parallel to the view plane.
Dolly tool  + click / Press <b>Alt+right-click</b>	Dolly/zoom the camera toward or away from the point of interest.
Tumble tool  + click / Press <b>Alt+left-click</b>	Tumble/rotate the camera around the point of interest.
Look around tool  + click / Press <b>B+click</b>	Rotate the camera around its eye. This can help to create bookmarks for <a href="#">360 VR exports</a> .
Press <b>F</b>	Frame the selection (or the whole scene if the selection is empty).
Press arrow keys	Move the camera left, right, forward, or backward.
Press <b>A</b>	Frame the whole scene.
Press <b>N</b>	Turn the camera's view direction toward north while keeping the view angle to the ground.
Press <b>Shift+N</b>	Turn the camera's view direction vertically down and rotate the scene so north is up.
Press <b>X</b> , <b>Y</b> , or <b>Z</b>	Turn the camera's view to respectively align to the x-, y-, or z-axis. This allows you to quickly see front, back, top, bottom, and side views.
Press <b>H</b>	Reset the camera to the default position.
Press <b>Spacebar</b>	Maximize 3D <b>Viewport</b> . The active 3D <b>Viewport</b> is maximized and fills the entire CityEngine window. Press the <b>Spacebar</b> again to restore the previous window layout.
Right-click <b>Viewport</b> then <b>Toggle Fullscreen</b>	Make 3D <b>Viewport</b> fullscreen. To return to the normal mode, press <b>Esc</b> or click the same context menu entry.


You can click an object to select it or drag a selection rectangle and press **F** to frame the selection. Then you can rotate, pan, or zoom in and out to explore the object and CityEngine scene.

 **Tip:**




- The CityEngine preferences (**Edit > Preferences > General > Navigation Devices > Mouse**) allow you to change the mouse navigation schemes according to the schemes of other 3D applications.  
If you have a 3Dconnexion SpaceMouse device (<https://www.3dconnexion.com/>), click **Edit > Preferences > General > Navigation Devices > 3D Mouse** to set the scheme.
- Linux users may want to change the modifier key mapping for navigation to the **Ctrl** key since some window managers catch the **Alt** key.

# Select objects

When you select objects with the **Select** tool  ( **Q** ), you identify the objects to which you want to apply an action or operation.

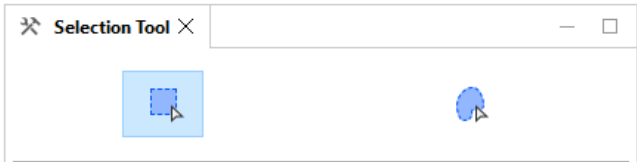
- You can select objects in the [Viewport](#), [Scene Editor](#), or **Select** menu. When selecting objects in the viewport, they get highlighted in the [Scene Editor](#) as well and vice versa.
- Selected objects are highlighted light blue/light gray.
- Objects that are part of a layer that is locked or set as invisible are not selectable.
- Activating [Isolation mode](#) ( **I** ) and altering scene object type visibility (viewport menu) do not affect the current selection.
- Map layers can be selected in the **Scene Editor** window  only.


## Select






To create a rectangular selection, click the **Select** button  ( **Q** ) to open the **Selection** tool  in the **Tool Options** window . You can also click **Select > Selection Tool** in the main menu.

Drag the selection box over the objects to select. Release the mouse button to finalize the selection.




## Selection tool options



The **Selection** tool options  include the following:

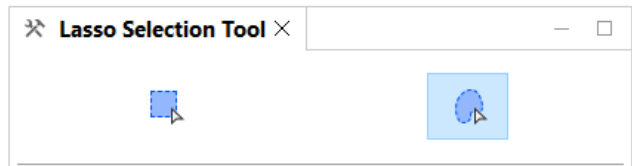
 <b>Selection</b>	
<b>Mode</b>	<ul style="list-style-type: none"><li>•  Select</li><li>•  Add to selection ( <b>Shift</b> )</li><li>•  Subtract from selection ( <b>Shift+Ctrl</b> )</li><li>•  Invert selection ( <b>Ctrl</b> )</li></ul> <p>Press and hold the modifiers in the parentheses to change the mode.</p>
<b>Disable Highlight</b>	Hides the selection highlighting while the object remains selected. This is useful when modifying CGA attributes in the Inspector.
<b>Selection Sets</b>	See the <a href="#">Selection sets</a> section below.


## Lasso select






To create a lasso selection, click the **Select** button  ( **Q** ) and click the **Lasso Selection** tool  in the **Tool Options** window . You can also click **Select > Lasso Selection Tool** in the main menu.

Click and hold the mouse button and drag the mouse following the boundary of the area you want to select to create a precise custom selection. Release the mouse button to finalize the selection.

### Lasso Selection tool options





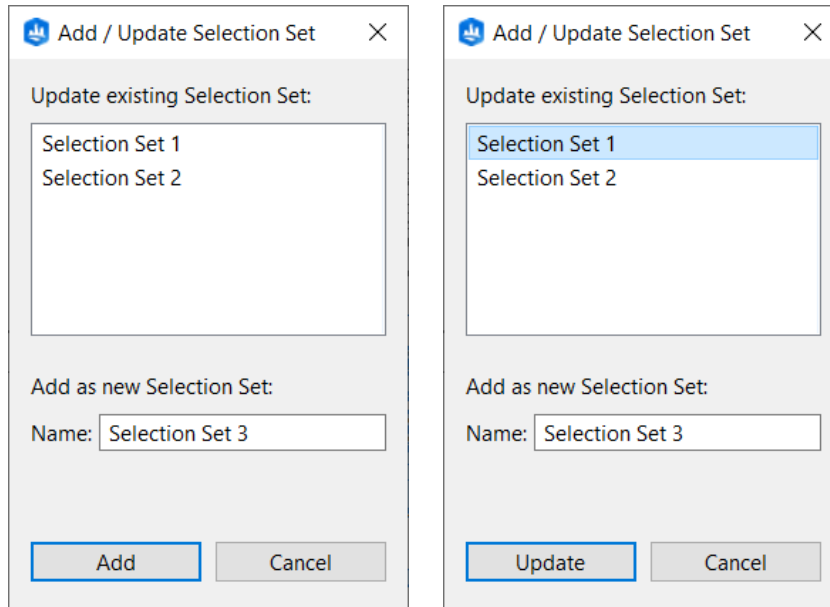
The **Lasso Selection** tool options  include the following:

 <b>Lasso Selection</b>	
<b>Mode</b>	<ul style="list-style-type: none"><li>•  Select</li><li>•  Add to selection ( <b>Shift</b> )</li><li>•  Subtract from selection ( <b>Shift+Ctrl</b> )</li><li>•  Invert selection ( <b>Ctrl</b> )</li></ul> <p>Press and hold the modifiers in the parentheses to change the mode.</p>
<b>Disable Highlight</b>	Hides the selection highlighting while the object remains selected. This is useful when modifying CGA attributes in the Inspector.
<b>Selection Sets</b>	See the <a href="#">Selection sets</a> section below.

### Selection sets

You can add, update, or edit selection sets for later use. To add or update a selection set, do the following:

1. Select the shapes to add or update a selection set.
2. Click the **Add / Update Selection Set** button  under **Selection Sets** in the  **Selection Tool** options. You can also click **Select > Add / Update Selection Set** in the main menu.
3. To add or update an existing selection set, do one of the following:
  - Click **Add** to add and save the selection set.
  - Select an existing selection set in the list and click **Update** to update the selection.



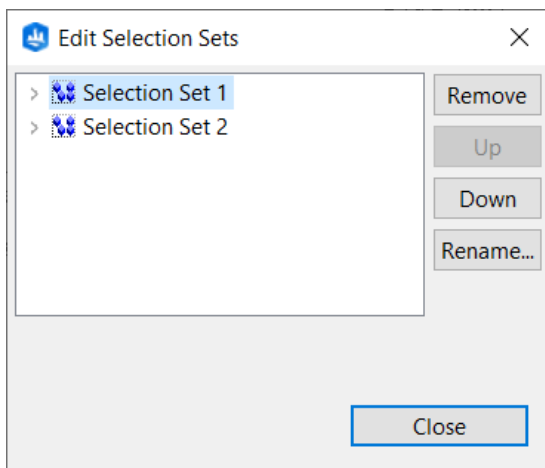
4. Click **Cancel** to close the window without adding or updating selections sets.

To apply a saved selection set to shapes, do the following:

1. Click the **Choose...** drop-down menu under **Selection Sets** in the **Selection Tool** options. You can also click **Select > Apply Selection Set** in the main menu.
2. Click a saved selection set in the list.  
The selection is applied to the shapes.

You can rename, reorder, or remove an existing selection set by doing one of the following:

- Click **Edit Selection Sets...** under **Selection Sets** in the **Selection Tool** options.
- Click **Select > Edit Selection Sets...** in the main menu.



 **Note:**

- Selection sets use an internal identifier to reference the selected objects. Therefore, previously created selection sets do not break when you rename objects.
- When deleting a selection set, the objects within don't get deleted.
- When loading a selection set, objects that have been deleted in the meantime are ignored.

## Select in the Viewport window


The **Select** tool  (Q) offers different methods to select objects:

Click	Selects an individual object. On an already selected object, components are selected (faces, edges, or vertices).
Double-click	Selects and frames an individual object.
Left-to-right selection rectangle	Selects objects or components that are fully inside the selection rectangle. Map layers are not added to the selection.
Right-to-left selection rectangle	Selects objects that intersect with the selection rectangle. Map layers are not added to the selection. *


\* This option isn't available with the **Lasso Selection** tool.

The selection methods can be combined with modifiers:

Shift+click	Add to existing selection.
Ctrl+click	Deselects a selected object or adds an unselected object to a selection.
Ctrl+Shift+click	Remove objects from an existing selection. Unselected objects are not added to the selection. This feature is especially handy when selecting using a selection box.



Right-click in the **Viewport** window  offers a subset of selection methods that are described in the [Select main menu](#).

## Select in the Scene Editor window

The following options are available to perform selections in the **Scene Editor** window :

Click	<ul style="list-style-type: none"><li>• On a layer/group layer: selects the layer, but does not select the scene objects inside.</li><li>• On a scene object: selects individual object, deselects already selected objects.</li></ul>
Double-click	<ul style="list-style-type: none"><li>• On a layer/group layer: has no effect.</li><li>• On a scene object: select and frame individual object.</li></ul>
Shift+click	Add to selection, in lists: add adjacent objects.
Ctrl+click	Deselects a selected object or adds an unselected object to the selection.


## Select main menu

The **Select** menu provides methods to grow a current selection you made in the **Scene Editor** window  or in the **Viewport** window .



<b>Select All, Deselect All, Invert Selection</b>	<ul style="list-style-type: none"><li>• In this case, "All" means scene objects that are not locked and with visibility = true. Map layers are not included.</li><li>• Select all and deselect all can also be triggered using the shortcuts <b>Ctrl+A</b> and <b>Ctrl+Shift+A</b>.</li><li>• Invert the selection of all selected objects.</li></ul>
<b>Select Objects in Same Layer</b>	All objects in the same layer or layers are selected.
<b>Select Objects by Map Layer</b>	See <a href="#">Selection with image maps</a> for details.
<b>Select Objects in Same Layer Group</b>	All objects in the same layer group or groups are selected.
<b>Select Objects of Same Type</b>	Selects all objects with the same type or types as the current selection.
<b>Select Objects of Same Group</b>	Selects all objects belonging to the same group as the objects in the source selection. For example, if the source selection contains a block, the resulting selection will contain the block and the block's shapes.
<b>Select Objects with Same Rule File</b>	Selects all objects having assigned a rule file that is present in the source selection.
<b>Select Objects with Same Start Rule</b>	Selects all objects having a start rule that is present in the source selection.
<b>Select Continuous Graph Objects</b>	This method can be used to select graph segments that are continuous, for example, if they together define one street. The search for continued segments starts at the source selection.

## Multi selections and lead selection


When you create multi selections, the highlighted color of the "lead selection" object varies from the color of the other selected objects. This is because the lead selection is used for operations where additional information is needed. For example, the subtract tool uses the lead selection to determine whether object A shall be subtracted from B or B from A.

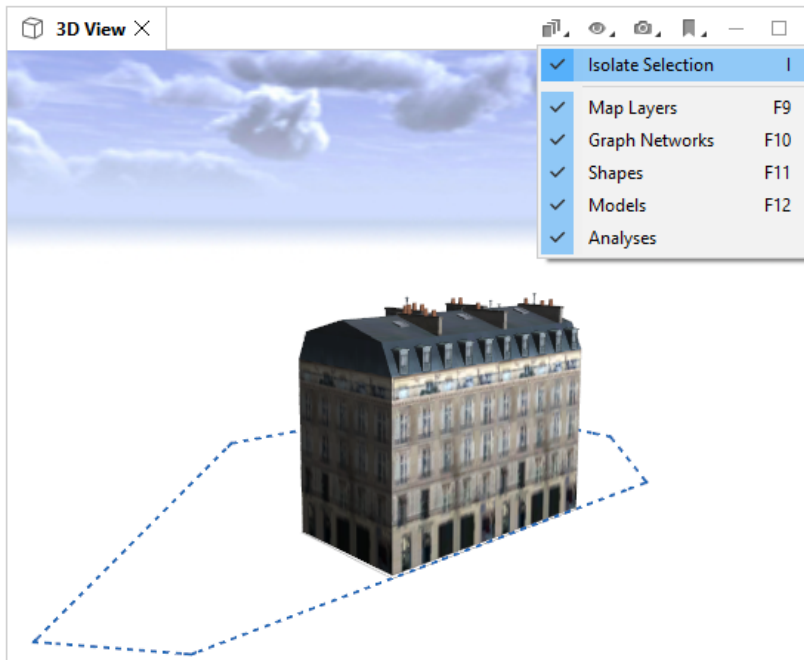
- Typically, the last object selected becomes the lead selection.
- The lead selection is marked with a slightly brighter selection color.
- To change the lead selection in an existing selection, use **Shift+click** in the **Viewport** window .

## Isolate selection

When working with large scenes, it is often useful to hide objects temporarily to declutter the scene. Open the **Visibility settings**  and click **Isolate Selection** or press **I** in the **Viewport** window .


1. Select the object to isolate.
2. Click **Isolate Selection** (I) to remove all other models in the scene.  
Modify the isolated objects. You can also change the selection.

3. Click **Isolate Selection** (I) again to reveal all (according to visibility settings in the **Scene Editor** window ) models in the scene.




*Isolate selection in the Viewport*

# Cameras

CityEngine supports an arbitrary number of cameras. Multiple cameras are especially useful if you are working with more than one 3D **Viewport** window . Here are the predefined cameras:

- **Perspective**—Perspective view of the scene.
- **Front**—Front view of the scene (you look along the z-axis)
- **Top**—Top view of the scene (you look along the y-axis)
- **Side**—Side view of the scene (you look along the x-axis)

In addition to these predefined cameras, you can align the camera along a specific axis by pressing **X**, **Y**, or **Z**. While **Y** only orients the y-axis in the camera direction, **X** and **Z** also reorient the y-axis upward. This, in combination with switching between orthographic and perspective view ( **D,P** ), allows you to quickly walk through multiple views of your scene.

See [cameras](#) in the **View settings** tool  for more information about the default cameras.

## Access cameras

Cameras are accessible from the **Window > New Viewport** menu.

To manage cameras, do the following:


1. Click **Window > New Viewport > New Camera**.  
This creates a camera from the current view.
2. Click **Window > New Viewport > Manage Cameras** or **Edit > Preferences > General > Viewport > Cameras** to manage and edit cameras.  
The [Camera preferences](#) menu opens. You can add new cameras or edit current ones. Cameras can be shared across 3D viewports, meaning that if multiple 3D viewports use the same camera, changing the camera in one viewport (for example, by rotation) affects the second viewport with the same camera as well.

## Configure cameras

Camera configurations have the following options:

- **Perspective Projection View**—Allows you to turn off perspective rendering
- **Angle of view**—The width of the field of view
- **Position X, Y, Z**—The position of the camera
- **Tilt Angle**—The rotation of the camera around the view direction
- **Heading Angle**—The rotation of the camera around up- axis


# Bookmarks

A specific camera configuration can be saved as a bookmark. Bookmarks are accessible from the **Bookmarks** menu . The home bookmark can also be activated by pressing **H**.

## Create a bookmark





Bookmarks belong to the camera from which they were created and can only be applied to that camera.

To create a bookmark, do the following:

1. Adjust the camera as desired for the bookmark.
2. Click the **Bookmarks** menu .
3. Click **New Bookmark**.  
The first 10 bookmarks are mapped to the numeric keypad and can be activated by pressing the corresponding key on the numeric keypad.
4. Choose a name and confirm.

## Edit a bookmark


You can edit a bookmark in the following ways:

- Click the **Bookmarks** menu  and click **Edit Bookmarks**.
- Click a **Bookmark gizmo** in the **Viewport** window  to open **Bookmarks**  in the **Inspector** window .
- Click **Edit** > **Preferences** > **General** > **Viewport** > **Bookmarks**.


In the **Bookmarks** menu, you can create bookmarks, change the order of the bookmarks in the list, rename bookmarks, and update the camera properties such as [two-point perspective correction](#).

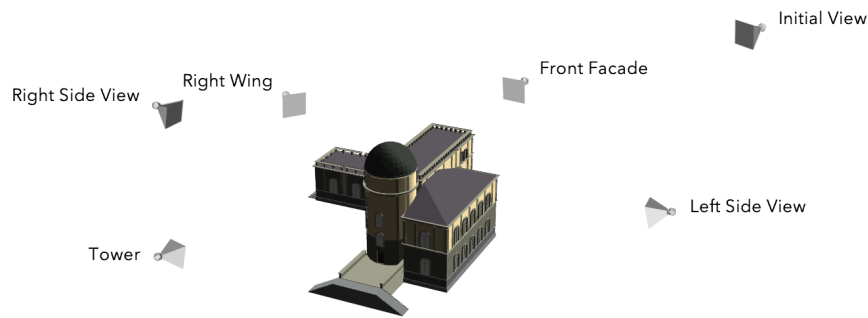
A bookmark has the same settings as a [camera](#).

### Tip:

- Bookmarks can be selected and transformed directly in the **Viewport** window  using the **Move**, **Scale**, and **Rotate** tools.
- Press **Ctrl+[Numpad]** to create a bookmark or update the current bookmark to the corresponding number entered.

## Bookmarks visibility

[Turn on or off visibility](#) of **Bookmark gizmos** in the **View Settings** menu . With **Bookmark gizmos** visible, you can see the bookmark position and camera orientation in the **Viewport** for each bookmark.



Bookmark positions with gizmos are shown.



### Tip:

Turn on bookmarks visibility when creating a CityEngine scene that will be exported to [360 VR Experience](#).


## Create clipping planes

The near and far clipping planes are virtual planes perpendicular to the camera's view axis and help determine which parts of a scene are visible. Objects closer to the camera than the near plane or farther away than the far plane are not rendered. You can create clipping planes using **Bookmarks** .

To create clipping planes, do the following:

1. Click one of the **Bookmark gizmos** to open **Bookmarks** in the **Inspector** window .
2. Turn on the **Custom Clipping Planes** toggle button.  
This sets the near and far clipping planes based on the values of the near and far clipping distances. Otherwise, the clipping planes are automatically set so that the whole scene is visible.
3. Set the **Near Plane Distance** and **Far Plane Distance** values.
  - **Near Plane Distance** is the distance in meters between the camera and the near clipping plane. Objects located between the camera and the near clipping plane are not rendered.
  - **Far Plane Distance** is the distance in meters between the camera and the far clipping plane. Objects located behind the far clipping plane are not rendered.
4. Click **Bookmarks** to see a preview of the clipping plane in the **Inspector** window .
5. To apply the bookmark with the set clipping planes to the **Viewport** window , click the bookmark in the **Bookmarks** menu .  
To reset the scene in the **Viewport** window , click the **View Settings** menu and choose **Reset clipping planes**.

# Scene light and panorama

Options for light source and environment and reflection maps for 3D viewports may be configured using the **Scene Light and Panorama** tool . You can access the tool the following ways:

- Click the **Scene Light and Panorama** tool  on the toolbar.
- Click **Edit** > **Edit Scene Light and Panorama** in the main menu.

## Light

The Light parameters control how objects are lit in the **Viewport** window .

<b>Sun position source</b>	Switches between Time and Month and Direct Solar Entry. For Time and Month to work, the <a href="#">Scene Coordinate System</a> must be set correctly.
<b>Time</b>	Time of sun position.
<b>Time Zone</b>	Time zone of sun position.
<b>Month</b>	Month of sun position.
<b>Solar elevation angle</b>	Altitude of the sun, the angle between the horizon and the center of the sun's disc.
<b>Solar azimuth angle</b>	Azimuth angle of the sun.
<b>Solar intensity</b>	Scene light intensity.
<b>Ambient intensity</b>	Scene light ambient intensity.
<b>Shadow attenuation</b>	Attenuation of shadows (blend to black).
<b>Ambient occlusion attenuation</b>	Attenuation of screen space ambient occlusion (blend to black).
<b>Shadow quality</b>	Switches between Low, Medium, High, and Interactive.
<b>Radius mode</b>	Switches between Manual and Interactive.
<b>Ambient occlusion radius</b>	Radius in meters of screen space ambient occlusion samples.
<b>Ambient occlusion samples</b>	Number of screen space ambient occlusion samples, or Interactive.

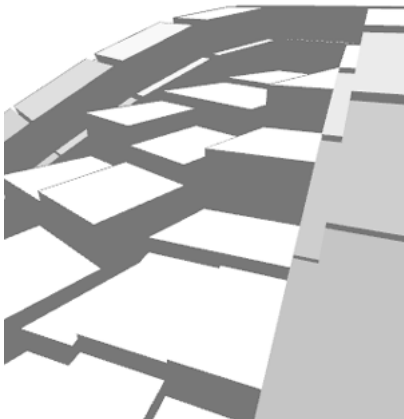
## Interactive modes

The Interactive modes are designed to deliver great visual quality without the need to manually adjust parameters.

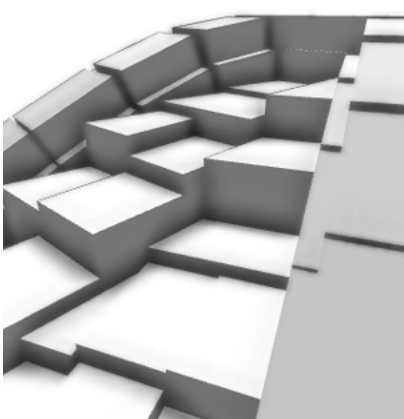
- When **Shadow quality** or **Ambient occlusion samples** are set to **Interactive**, the best visual quality is used when the camera stands still, while the quality is reduced based on performance when the camera moves.
- When setting **Radius mode** to interactive, the ambient occlusion radius is automatically adjusted for best visual quality in the center region of the viewport. To tweak this, set the mode to **Manual**.

## Ambient occlusion

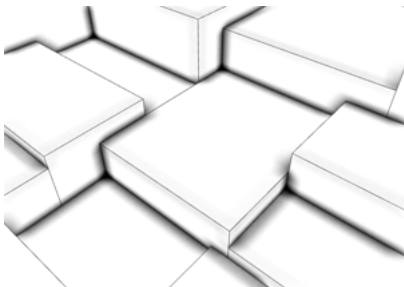
CityEngine ambient occlusion uses screen space ambient occlusion, a special real-time rendering technique that approximates ambient light occlusion. Ambient occlusion is a shading method used in 3D computer graphics that helps add realism by taking into account attenuation of light due to occlusion. Ambient occlusion attempts to approximate the way light radiates in real life and is especially well suited for outdoor scenes.



*Solid shading without ambient occlusion is shown.*



*Ambient occlusion shown as enabled.*



*Low ambient occlusion radius is shown.*



*High ambient occlusion radius is shown.*

## Panorama

The panorama settings contain the background skydome shown in the 3D viewports:

<b>Environment Map</b>	<ul style="list-style-type: none"><li>• Image to be used as the environment map.</li><li>• The environment map is used to texture the sky in a scene.</li></ul>
<b>Reflection Map</b>	<ul style="list-style-type: none"><li>• Image to be used as the reflection map.</li><li>• The reflection map is used on reflections on generated models that have reflection enabled in their material settings.</li></ul>
<b>Visible</b>	Show or hide the panorama.

The directory `ce.lib/maps/panoramas` contains a selection of panorama maps.

# Georeferencing

CityEngine provides various tools to georeference geometry and models. The coordinate system determines how data is georeferenced in the scene.

## Scene Coordinate System

The **Scene Coordinate System** defines your scene's reference coordinate system for georeferenced data import, export, and coordinate values in the scene. You can display the **Scene Coordinate System** and other scene information by turning on **Information display**. You can also view the **Scene Coordinate System** by displaying the **Status line** window.

To display the **Information display** beneath the scene, do the following:

1. Click **View settings > Information display**.  
This displays scene information, such as **Scene coordinate system**, grid size, and coordinates of the pointer.
- 2.

To display the **Status line**, do the following:

1. Click **Window > Show Status Line**.
2. The **Status Line** displays in the lower right of the main CityEngine layout. The **Status Line** also displays valuable memory information.

## Default Scene Coordinate System

A CityEngine scene can have a **Scene Coordinate System** (SCS) set when it was saved previously. By default, the SCS is set to **CityEngine CS (meters)**.

## Set the Scene Coordinate System

When working with georeferenced data, it is often important that your scene has a valid georeferenced coordinate system set. There are three ways to set the **Scene Coordinate System**:

- Set a coordinate system when georeferenced data is added to the scene.
- Set a coordinate system when creating a scene.
- Change the **Scene Coordinate System** in the preferences.

### *Set a coordinate system when adding georeferenced data*

As soon as georeferenced data is imported into a CityEngine scene, a **Scene Coordinate System** is required to locate it correctly. CityEngine automatically opens the [Select Coordinate System](#) menu on your first import of georeferenced data. Taking the data that is going to be imported into account, the dialog box suggests a matching scene coordinate system.

### *Set a coordinate system when creating a scene*

When creating a CityEngine scene, you have the option to set your **Scene Coordinate System** in the new scene wizard. The scene [Select Coordinate System](#) menu appears when you click **Choose**.

*Change the Scene Coordinate System in the preferences*

The **Scene Coordinate System** can be set or changed in the CityEngine preferences by clicking **Edit > Preferences > Scene**. Click the **Scene Coordinate System** button to open the [Select Coordinate System](#) menu.

Once a **Scene Coordinate System** is set, it is displayed in the **Information display** and the **Status Line**.

 **Note:**

- When changing the **Scene Coordinate System**, no reprojection is applied to the content of the scene. Using this option only changes the scene's reference system, used for viewing coordinates as well as for future data imports.
- CityEngine currently does not support datum transformations. It is recommended that you reproject your data to match the target SCS before importing into a CityEngine scene.
- In CityEngine, the **Scene Coordinate System** can only be a projected coordinate system (no geographic coordinate systems).

## View coordinate system

The **View coordinate system** option allows you to view the **Information display** bar details when different coordinate systems are applied.

If the **Scene Coordinate System** is set, you can choose the coordinate system by selecting **View Settings > View coordinate system**.

<b>CityEngine CS [meters]</b>	CityEngine coordinates y-up, meters
<b>Scene CS [feet]</b>	Current Scene Coordinate System in feet
<b>Scene CS [meters]</b>	Current Scene Coordinate System in meters
<b>UTM</b>	Universal Transverse Mercator
<b>MGRS</b>	Military Grid Reference System
<b>Long/Lat [decimal degrees]</b>	Longitude/latitude in decimal degrees
<b>Long/Lat [degrees min sec]</b>	Longitude/latitude in degrees, minutes, and seconds

After changing the **View Coordinate System**, the **Information display** bar shows the coordinates in the changed coordinate system.

 **Note:**

- When the **Scene Coordinate System** isn't set, the option to change the coordinate system isn't available. See [Set the Scene Coordinate System](#) for more information.
- The **Scene Coordinate System** in the **Status Line** window remains unchanged.

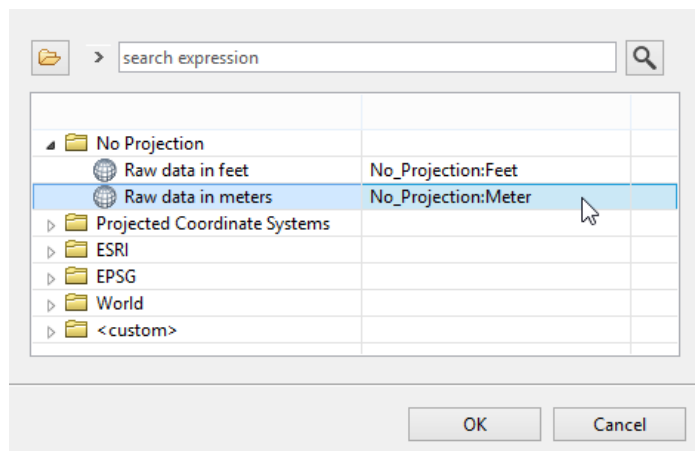
## Coordinate System dialog box

The **Coordinate System** dialog box allows you to [set the coordinate system](#) when creating a scene, accessing the scene preferences, or importing georeferenced data.

## Select the coordinate system

Use the **Select Coordinate System** menu to set the **Scene Coordinate System**. Browse through the coordinate systems or use the search field to select the desired coordinate system.

When this dialog box appears during the first import of georeferenced data, the previously selected coordinate system for data import is suggested. If you do not need a georeferenced coordinate system, the **No Projection** > **Raw Data in Meters** option is a good choice.



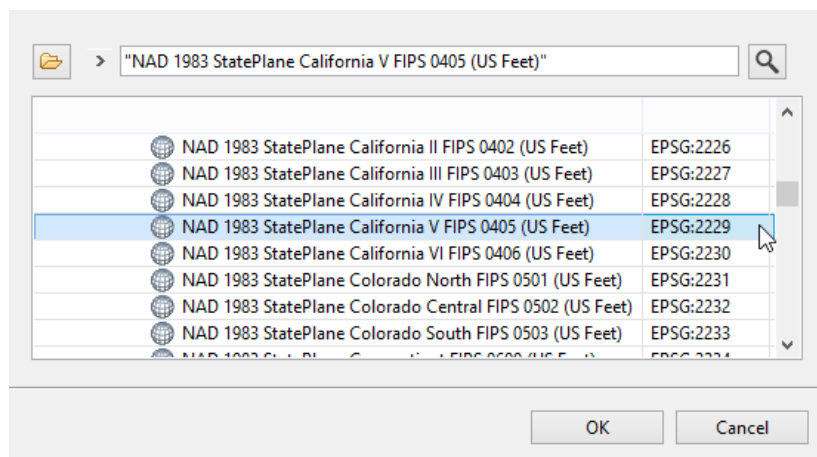
*No projection in meters is shown.*

### Note:

The **Scene Coordinate System** can only be a projected coordinate system, hence the geographic coordinate systems are not available on this dialog box.

## Select data coordinate system

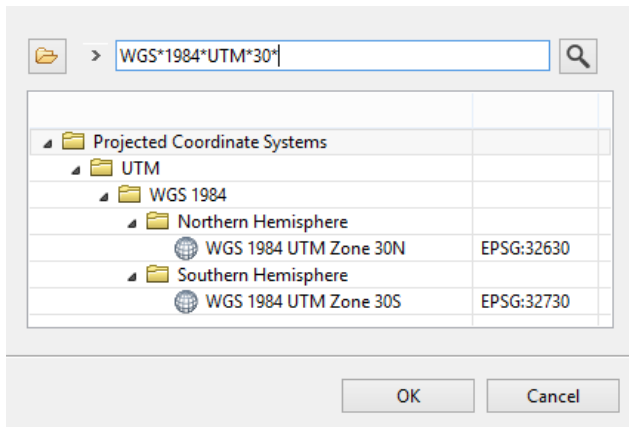
The data coordinate system pops up whenever georeferenced data is going to be imported and no projection details are found with the data. Usually, you will see a coordinate system suggested the **Scene Coordinate System** when this dialog box appears, which is a good option.



*Set the coordinate system.*

### Use the search field

Use the search expression field to filter the list of coordinate systems and search for a coordinate system by name or authority code. Use the wildcard character \* to define your search query. To reset the search filter and show all available coordinate systems again, clear the search field and press **Enter** (or the search button on the right).

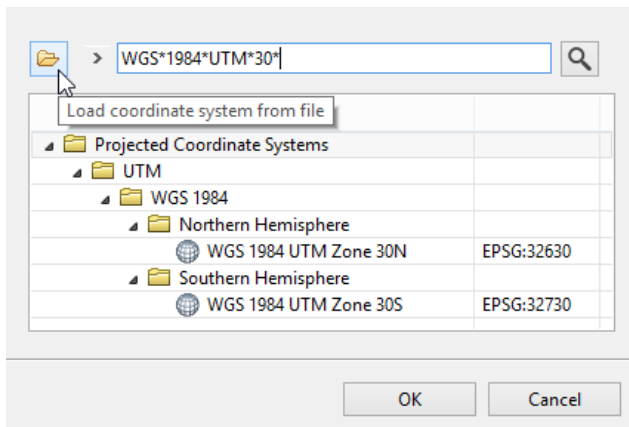


Search for a coordinate system.

### Load the coordinate system from a .prj file

When the required coordinate system is not available in the list, you can choose a new projection by browsing to an arbitrary .prj file. Use the folder button on the upper left to open the file dialog box, and browse to the .prj file.

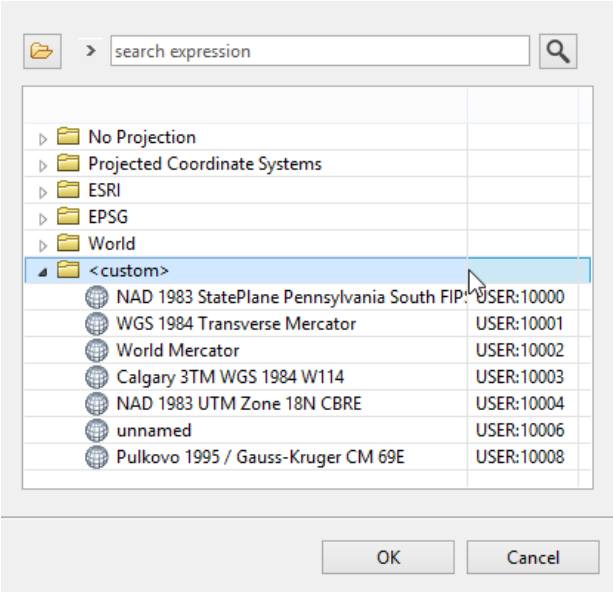
This feature also allows you to define your own custom coordinate system by creating your own .prj file with the desired parameters.



Load coordinate system from a file.

### Custom coordinate systems

If a projection definition is found that can't be matched to one of the predefined coordinate systems, CityEngine adds it as a new custom coordinate system. These have their authority set to USER, and an incremental number for the authority code.



A custom coordinate system is shown.

## Import georeferenced data

CityEngine handles importing georeferenced data differently, depending on the data type.

### Vector data

All georeferenced vector data is reprojected to the **Scene Coordinate System** during import.

- Georeferenced vector data requires a coordinate system on import; if no such data is found, the [Select Coordinate System](#) menu opens during import, allowing the user to choose the coordinate system for the data.
- If the **Scene Coordinate System** isn't defined for the scene previous to the data import, the [Select Coordinate System](#) opens and allows the user to set the **Scene Coordinate System**.

<b>Shapefiles (.shp)</b>	CityEngine looks for a .prj file with the same name in the same folder.
<b>File Geodatabase (.gdb)</b>	The coordinate system is read per layer of the .gdb dataset.
<b>KML/KMZ</b>	KML/KMZ latitude/longitude data is always interpreted as WGS84. When KML/KMZ data is imported into a scene with no coordinate system set, UTM with a matching zone is suggested as the Scene Coordinate System.
<b>OpenStreetMap (OSM)</b>	OSM latitude/longitude data is always interpreted as WGS84. When OSM data is imported into a scene with no coordinate system set, UTM with a matching zone is suggested as the <b>Scene Coordinate System</b> .

### Georeferenced image data

Image data is considered to be georeferenced if the following occurs:

- It contains embedded georeferencing metadata.
- A world file is found that belongs to the image.
- A .prj file is found that belongs to the image.

Reprojection or rotation isn't applied to image data on import. The image's coordinate system is used only as a reference system for location, extent, and unit (translate and scale transformation) for the image data during import. It is therefore important to have image data ready in the target coordinate system before importing into CityEngine. (See the GeoTIFF section below.)

#### *Reference coordinate system*

CityEngine looks for a .prj file with the same name in the same folder to use as the coordinate system for the data. If no .prj file is found, the [Select Coordinate System](#) menu opens and allows the user to choose the coordinate system for the data.

#### *Map extent*

CityEngine looks for a world file (for example, .jgw, .tfw) in the same folder to read the extent of the data. The world file data is interpreted in the unit found in the coordinate system of the data, and calculated accordingly if required. If no world file is found, the size and position of the map can be entered manually in the [Map Layer](#) menu.

#### *Elevation range*

CityEngine looks for minimum and maximum range values in the metadata embedded in the image. The range data is interpreted in the unit found in the coordinate system of the data, and calculated accordingly if required. If no elevation range is found, the elevation range of the map can be entered manually in the [Map Layer](#) menu.

Elevation range data is only used when the image is used as a height map.

### GeoTIFF (.tif, .tiff with embedded metadata)

GeoTIFF data is handled the same way as other georeferenced image data with the following differences:

- If a coordinate system is defined with authority and code (for example, ESRI:102132) in the GeoTIFF metadata tags, this information is used to set the data's coordinate system, and has precedence over a .prj file.
- If no world file is present, the GeoTIFF metadata tags are searched for map extent data. If present, the data is treated the same way as in the world file case.

Parameters for location, extent, and elevation range can still be entered manually for nongeoreferenced images in the [Map Layer](#) menu.

# Scene Objects

# Scene objects

A scene consists of various types of scene objects organized in corresponding layers:

- [Shape layer](#)—Contains static shapes, typically used as building footprints for generation of CGA models.
- [Graph layer](#)—Contains street networks and blocks, dynamic shapes (street shapes, building footprints), and generated models.
- [Static model layer](#)—Contains static models, such as Collada files.
- [Map layer](#)—Contains arbitrary maps (images) and can be used to globally control various parameters for scene objects. The scene [terrain](#) is also created using a map layer.
- [Analysis layer](#)—Contains analysis tools such as viewshed or view corridor tools.

Each layer type can hold only elements of its own kind. To group elements of different types, use [group layers](#).

# Shapes

Shapes are commonly used to represent building footprints that are extruded either manually using the [Push Pull tool](#) or serve as starting points for [CGA rules](#). In the latter role, they are referred to as initial shapes.

## Create shapes manually

Shapes can be drawn and edited using a set of tools. See [Polygons, rectangles, circles](#) for more information about shape creation tools.

## Import shapes

Shapes from various file formats can be imported. The most important are the georeferenced formats, Esri FileGDB (.gdb), Esri Shapefiles (.shp), OpenStreetMap (.osm), and KML (.kml, .kmz).

A convenient way to import georeferenced shapes is through [ArcGIS Online](#) and the [Get map data](#) feature.

The nongeoreferenced 3D formats Wavefront OBJ (.obj), Autodesk FBX (.fbx), and Khronos Group glTF (.gltf, .gltb) are supported. These 3D models can be imported either as static models or as shapes. Import as shapes in case you plan to use the data as initial shapes for CGA modelling.

**Note:**

The term [shape](#) is also used in a different context when referring to rule-based modeling.

## Shapes in the graph layer

When drawing or importing graph layers, dynamic shapes are created automatically for nodes and edges.

**Note:**

These shapes can be found in the accompanying graph layer as children of the edges and nodes.

- These shapes are adjusted automatically when the underlying graph is edited, therefore, they are dynamic.
- These shapes are not editable by push pull tools.
- They are used for procedural modeling only. See [Graphs](#) for more information.

# Graphs

A graph consists basically of two elements: a group of points called nodes and segments that connect the nodes so that a mesh is formed. A graph layer holds segments and nodes as well as shapes that are created automatically.

Each segment and node has three shapes as child elements: one shape is referred as the lane, the other two represent sidewalks. To control basic attributes of those shapes such as widths, use the **Inspector** with the corresponding segment or node selected.

As with any other shape in CityEngine, the shapes associated to graph elements can be assigned with a CGA rule. You'll find a CGA rule that generates fully decorated street geometry in the `rules/Streets` folder in [ESRI.lib](#).



## Note:

Shapes that are children of graph elements can't be copied or moved to a shape layer.

## Create graphs

A graph can be [drawn manually](#), [generated procedurally](#) or by [importing](#) suitable data.

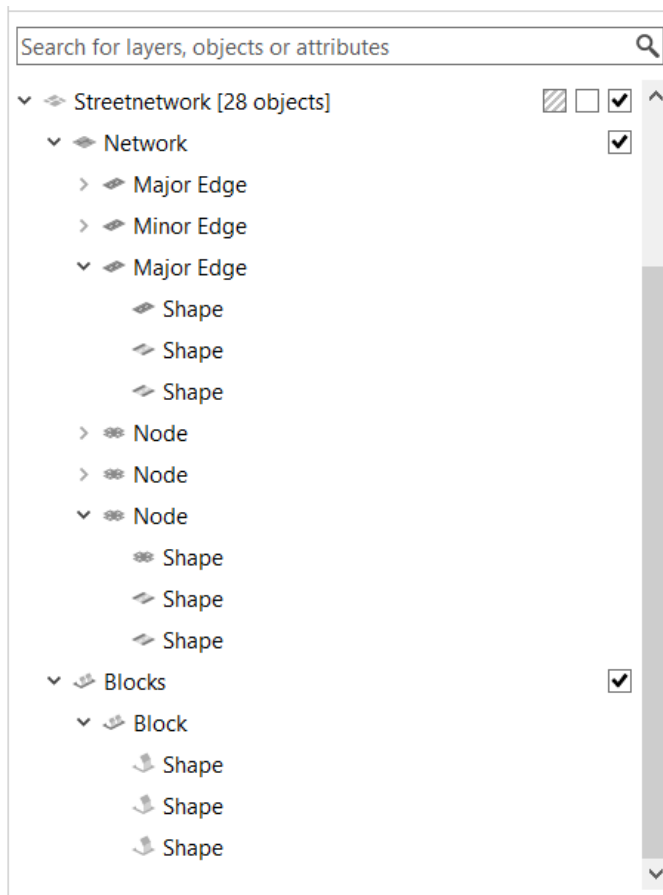
## Blocks

Whenever a closed loop of streets is formed by one of the creation methods described above, a block is created inside the loop. The block is tied to the surrounding streets. This means that when streets are moved, the shape of the block adjusts automatically. When the loop is opened by deleting a segment, for example, the block disappears.

A block is selected by clicking the dashed line. In the [Inspector](#), select the method to subdivide the block into lots.

## Lots

Lots are represented as shapes so that CGA rules can be assigned:



Expanded Street Network is shown with Segments, Nodes, Blocks, and Shapes in Scene Editor.



#### Note:

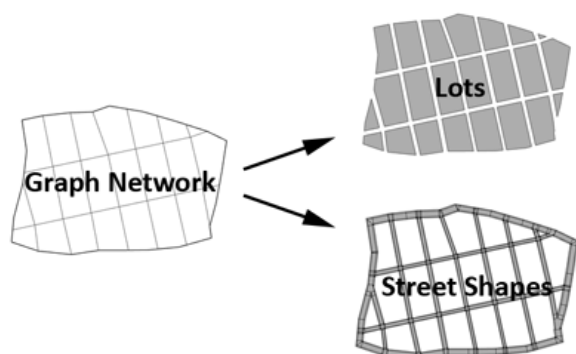
While street networks are the most common use case for graphs, CGA rules can be written to generate other linear features such as underground pipe networks, power lines, transportation lines, or even the walls of a medieval castle.

# Shapes from graphs

# Create shapes from graph networks

[Graph networks](#) can automatically create shapes, such as lots or street shapes.

To enable or disable shape creation, use the **Create Shape** parameter at the block, street, or node parameters in the inspector. By default, shape creation is enabled.



*Input: A graph network; Output: Lots and street shapes.*

## Shape creation parameters

You can manage lots and street shapes (Lot, Segment, Node, Sidewalk, Block) in the **Inspector**.



- Blocks create lots from block default start rules (Lot, LotCorner, LotInner); see [Block parameters](#).
- Segments create street shapes; see [Segment parameters](#) and [Sidewalk parameters](#).
- Nodes create intersection shapes; see [Node parameters](#).
- Shapes generated by nodes and segments from default start rules, such as Street, Sidewalk, or Roundabout, are specified by their shape types; see [Street and intersection shapes](#).

Additionally, the [Street tool](#) can be used to edit street widths or setup curve handles.

### **Note:**

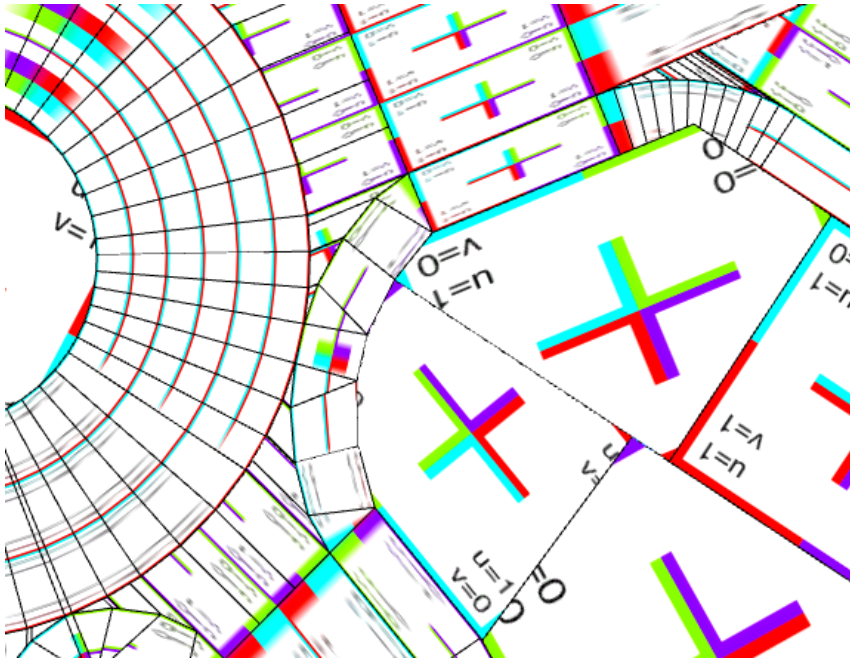
For each loop in the graph network, a block is automatically created; see [Graphs](#) for details.

## Object attribute inheritance

- Lots inherit the attributes of the block.
- Street shapes inherit the attributes of the segment.
- Intersection shapes inherit the attributes of the node.
- New object attributes will always be added to the node or segment.

## UV coordinates

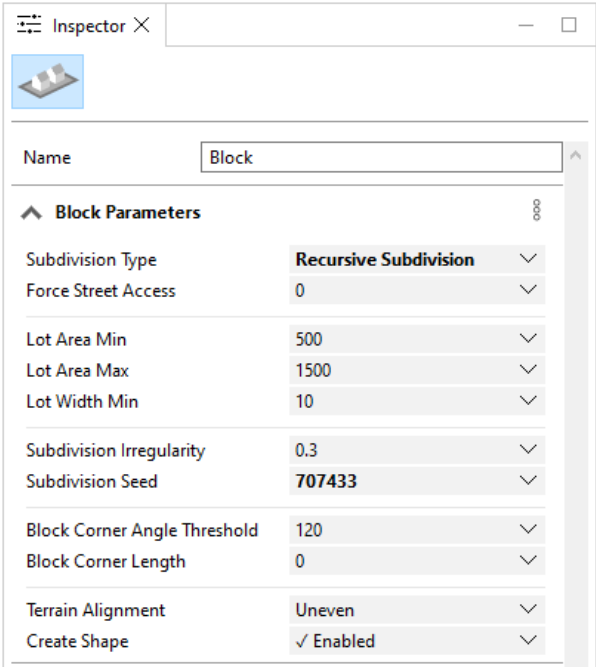
UV coordinates are generated for each shape. They can be used for [UV Splits](#) and texturing. For details of the UV coordinates; see [Street and intersection shape UV](#).



UV coordinates are shown.

# Block parameters

The parameters for a block subdivision can be specified under **Block Parameters** in the **Inspector**. Block parameters can be individually set for each block.



Block Parameters

## Note:

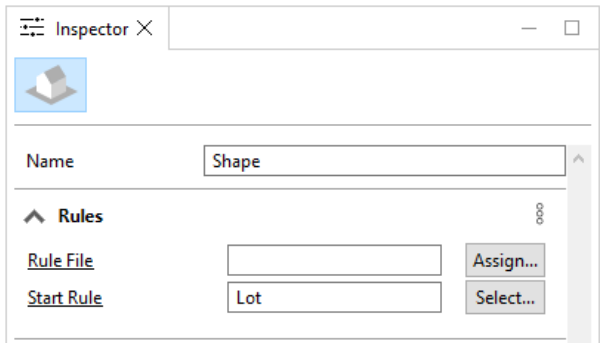
Parameters (attributes) can be mapped to Default, User, Object, or to a map layer. See [Mapping Attributes](#) for details.

## Lot default rules

You can divide blocks into polygonal shapes or lots. The lots have the following default start rules:

<b>Lot</b>	Polygonal shape touching a street.
<b>LotInner</b>	Lots that reside within a block but do not touch a street.
<b>LotCorner</b>	Created when the corners of a lot are given a width when you set the Block Corner Length parameter on a block with recursive or offset subdivision.

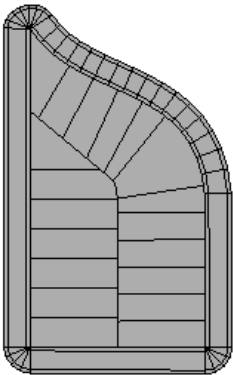
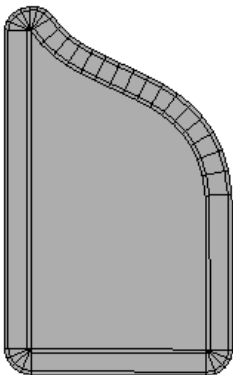
When you select a lot shape within a block, it displays the one of the start rules:



## Subdivision type parameters

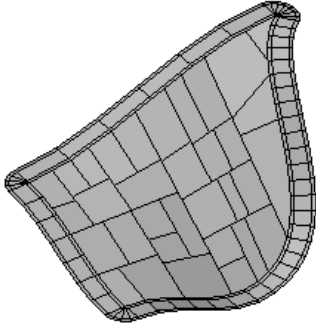
Several parameters are available that allow you to control the street shapes.

Subdivision Type (attr type)	
Specifies the subdivision algorithm to use: Recursive Subdivision, Offset Subdivision, Skeleton Subdivision, or No Subdivision	
Recursive Subdivision creates rectangular lots by repeatedly splitting the block.	
Offset Subdivision creates lots only within a given distance from the street edges of the block.	

<p>Skeleton Subdivision creates street-aligned lots that always have access.</p>	
<p>No Subdivision doesn't perform subdivision on this shape.</p>	

## General block parameters

The following tables describe parameters that are common to all the block types.

Terrain Alignment (attr alignment)	
This parameter is only used if the initial shapes are uneven. This sets the alignment of the lot over the terrain. There are four numeric options, as illustrated below.	
Valid values are [0,1,2,3]	
<p>0. Uneven. The lots follow the terrain, given uneven heights.</p>	

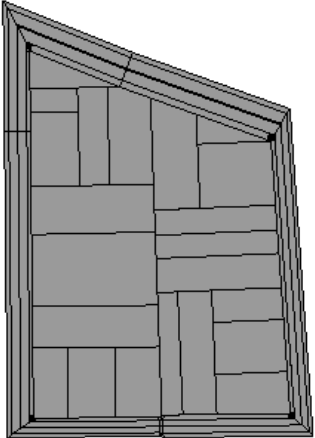
1. <b>Minimum.</b> The lots lie at their lowest point of the terrain that they cover.	
2. <b>Maximum.</b> The highest point of the lot is used.	
3. <b>Average.</b> The average height of the lot is used.	

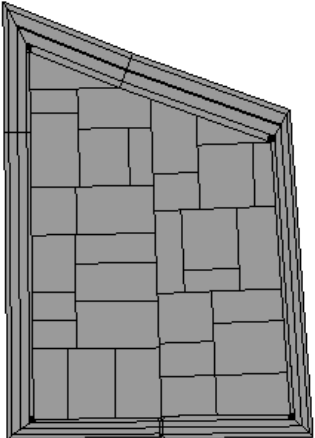
<b>Create Shape (attr shapeCreation)</b>
Enable or disable the shape geometry creation from the segment.

Recursive subdivision

The recursive subdivision technique is the default. It subdivides the block into rectangular lots of various sizes.

<b>Force Street Access (attr forceStreetAccess)</b>
The factor indicating the preference for lots with street access. A higher value results in more lots having street access.

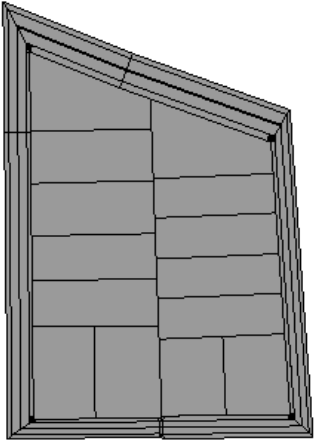
Given in range [0.0,1.0]	
Subdivision obtained for a Force Street Access value close to 0.	
Subdivision obtained for a Force Street Access value close to 1.0.	

<b>Lot Area Min (attr lotAreaMin) and Lot Area Max (attr lotAreaMax)</b>	
The approximate lower and upper bounds of the area of lots obtained after subdivision.	
Given in absolute area units	
Subdivision obtained for smaller Lot Area Min and Lot Area Max values.	

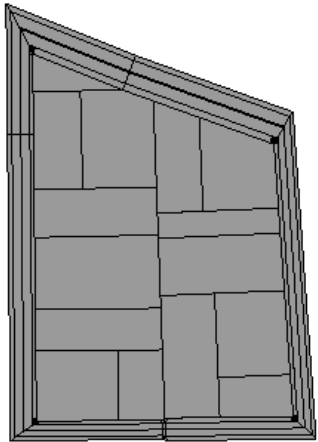
<p>Subdivision obtained for larger Lot Area Min and Lot Area Max values.</p>	
<p>Subdivision obtained when the difference between Lot Area Max and Lot Area Min is small.</p>	
<p>Subdivision obtained when the difference between Lot Area Max and Lot Area Min is large.</p>	

<p><b>Lot Width Min (attr lotWidthMin)</b></p>
<p>The minimum width of the side of a lot. Subdivision stops if the length of any of the sides of any of the resulting lots is less than this value. If this value is high, the area of resulting lots may be larger than the area specified by Lot Area Max.</p>
<p>Given in absolute length units</p>

<p>Subdivision obtained for a smaller Lot Width Min value.</p>	
<p>Subdivision obtained for a larger Lot Width Min value.</p>	

<p><b>Subdivision Irregularity (attr irregularity)</b></p> <p>The relative deviation of the split line from the middle point of the center of the oriented bounding box (OBB). If this value is 0.0, the split line will be pivoted at the middle point of the OBB of the parent lot. A higher value results in the split line being farther away from the middle point, and generally, in a higher difference in the areas of the two children nodes.</p> <p>Given in range [0.0,1.0]</p>	
<p>Subdivision obtained for a Subdivision Irregularity value close to 0.</p>	

Subdivision obtained for a Subdivision Irregularity value close to 0.5.



**Subdivision Seed (attr seed)**

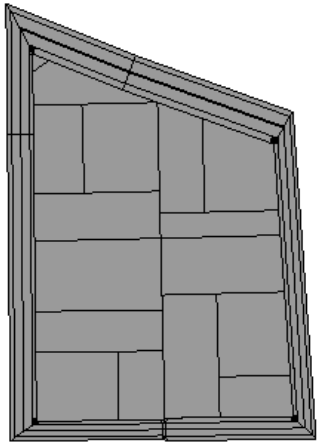
To increase the stability of subdivisions under interactive editing operations, the random seeds for the children lots of a given lot are computed before the recursive call to the subdivision function.

**Block Corner Angle Threshold (attr cornerAngleMax)**

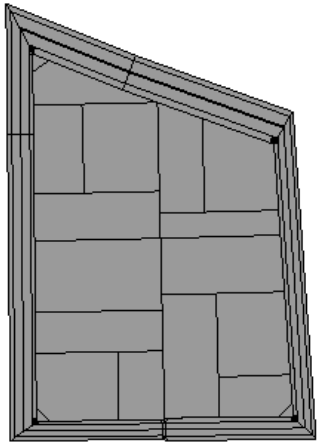
Corner angle threshold. If the angle at the vertex of a block contour is less than this value, a corner lot is inserted. A larger value results in a more relaxed criterion for inserting corners, and thus in more corners being created. If this value is 0.0, no corners are created.

Given in degrees

Subdivision obtained for a smaller Block Corner Angle Threshold value.



Subdivision obtained for a larger Block Corner Angle Threshold value.

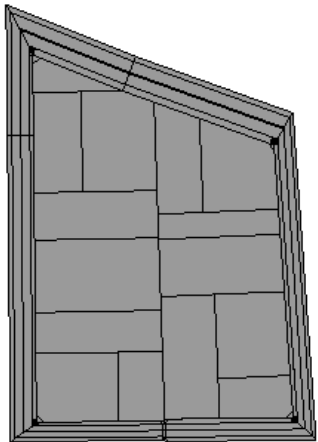


**Block Corner Length (attr cornerWidth)**

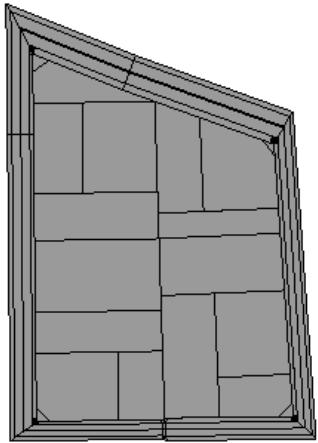
Width of the interior side of the created corners. If this value is 0.0, no corners are created. The maximum value for this attribute is automatically computed to avoid self-intersections.

Given in absolute length units

Subdivision obtained for a smaller Block Corner Length value.



Subdivision obtained for a larger Block Corner Length value.



## Offset subdivision

A block that uses offset subdivision is offset to create a fixed-width strip along the street edges, which is then subdivided into lots.

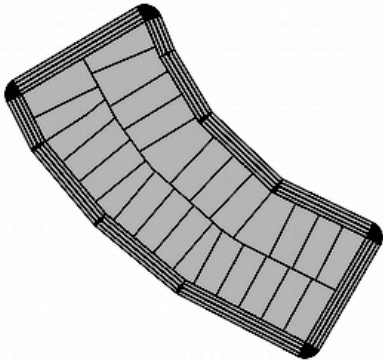
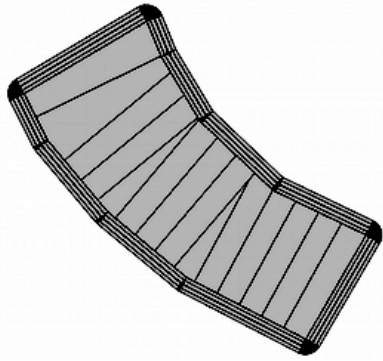
Offset Width (attr offsetWidth)	
<p>The perpendicular distance from the block contour to the inward offset polygon. Intuitively, this value corresponds to the depth of the lots that are created when offset subdivision is used.</p> <ul style="list-style-type: none"><li>• If this value is close to 0.0, OBB subdivision is used.</li><li>• If this value is high enough so that the offset polygon is collapsed, OBB subdivision is used.</li></ul>	
Given in absolute length units	
Subdivision obtained for a smaller Offset Width value.	
Subdivision obtained for a larger Offset Width value.	

Recursive Subdivision
<p>After the offset routine, there is an option to also run the recursive subdivision on the result. This is controlled by the same set of parameters as the recursive subdivision scheme.</p>

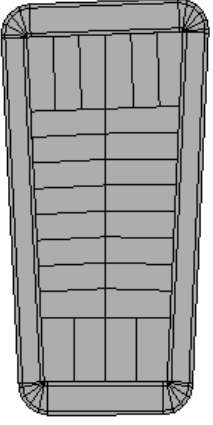
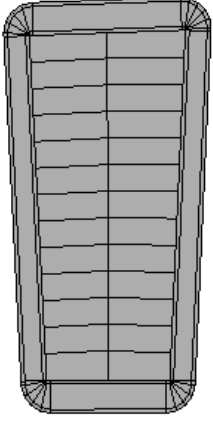
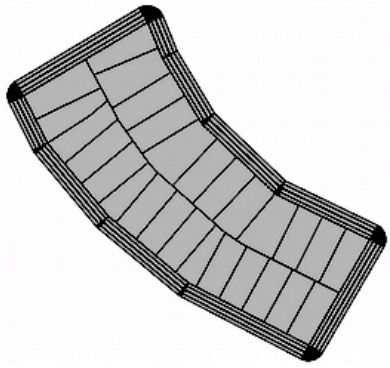
See [Recursive subdivision](#) for the rest of the specific parameters.

## Skeleton subdivision

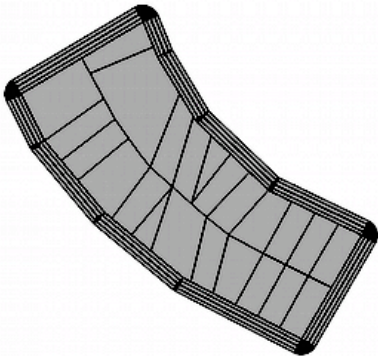
Skeleton subdivision attempts to subdivide a block such that every lot has access to the street. The sides of the lots are perpendicular to the roads to which they are adjacent.

Shallow Lot Fraction (attr shallowLotFrac)	
Limit for merging triangular lots.	
Subdivision obtained for a smaller Shallow Lot Fraction value.	
Subdivision obtained for a larger Shallow Lot Fraction value.	

Corner Alignment (attr cornerAlignment)
Skeleton subdivided lots face their nearest streets. At the corner of two streets, one will normally take priority. The corner alignment determines how this priority is assigned, either by Street length or by Street width.

<p>Street width—The widest street takes priority. If the streets have similar average widths, the street length is used instead.</p>	
<p>Street length—The longest street takes priority.</p>	
<p><b>Simplify (attr simplify)</b></p>	
<p>Amount of simplification that occurs. A high value creates irregular lots with fewer vertices.</p>	
<p>Given in range [0.0,1.0]</p>	
<p>Subdivision obtained for a smaller Simplify value.</p>	

Subdivision obtained for a larger Simplify value.



**Lot Area Min (attr lotAreaMin)**

After subdivision, lots with a small area are repeatedly combined with their neighbors until they are larger than this minimum. This reduces the number of smaller lots, but may create lots of more irregular shape.

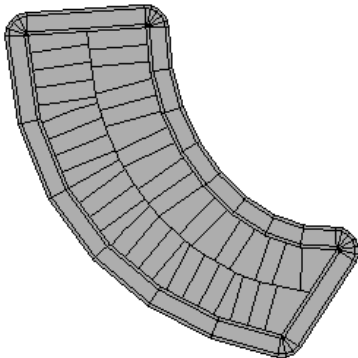
Given in absolute area units

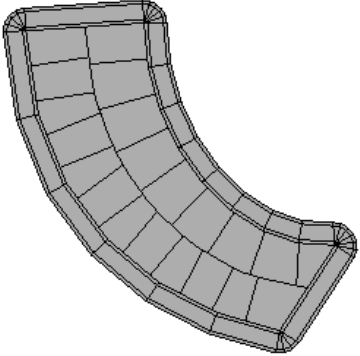
**Lot Width Min (attr lotWidthMin)**

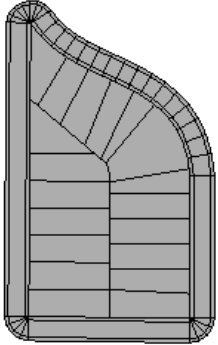
The ideal length of street front that each lot should possess. This is increased or decreased by several other processes. A low lot width relative to the block size may create many narrow lots.

Given in absolute length units

Subdivision obtained for a Lot Width Min value of 15.



<p>Subdivision obtained for a Lot Width Min value of 30.</p>	
--	---

<b>Subdivision Irregularity (attr irregularity)</b>	
As this parameter increases, it introduces a stochastic element into the lot width and lot edge direction.	
Given in range [0.0,1.0]	
<p>Subdivision obtained for a smaller Subdivision Irregularity value.</p>	
<p>Subdivision obtained for a larger Subdivision Irregularity value.</p>	

<b>Subdivision Seed (attr seed)</b>
To increase the stability of subdivisions under interactive editing operations, the random seeds for the children lots of a given lot are computed before the recursive call to the subdivision function.

## No subdivision

### No Subdivision

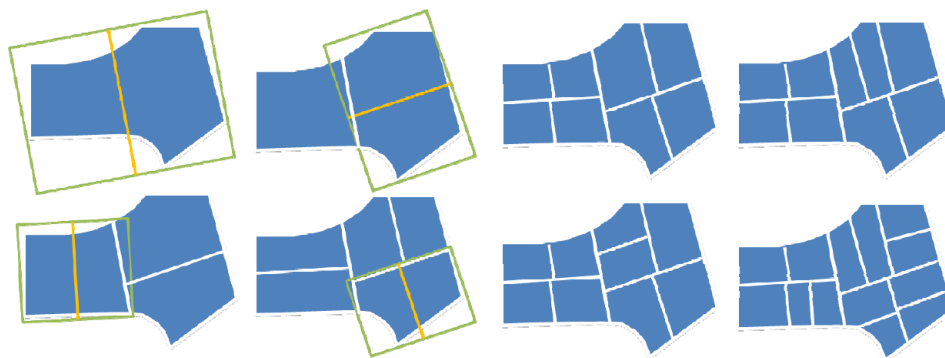
This simple subdivision technique subdivides the block into a single lot of the same shape. There is an option to remove the lot's corners.

## Description of algorithms

### Recursive OBB algorithm

The recursive OBB algorithm computes a split line at each step. If the two lots resulting from the split meet the user-specified constraints, the algorithm recurses on them. To determine the pivot point and direction of the split line, the minimum-area oriented bounding box (OBB) of the lot is computed. By default, the pivot point is set to the midpoint of the largest edge of the OBB, and the split line direction is set to the direction of the smallest edge of the OBB. The split line pivot and direction can be modified by the following criteria:

- **Street access**—If one of the lots resulting from a split has no street access, the orthogonal vector to the initial direction vector is used.
- **Snap to block contour vertices**—If the split line is within a threshold distance from one of the vertices of the contour of the original block, the pivot point of the split line is set to that vertex.
- **Edge alignment**—To increase the stability of subdivisions under interactive editing operations, the sampling angle space to compute an approximation of the OBB uses one of the lot edges as reference.
- **Random seeds**—To increase the stability of subdivisions under interactive editing operations, the random seeds for the children lots of a given lot are computed before the recursive call to the subdivision function.



*Successive steps of the recursive OBB algorithm*

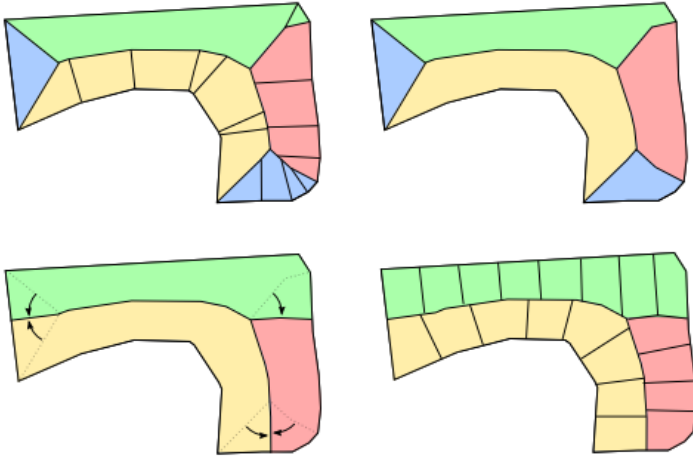
### Offset algorithm

The offset subdivision algorithm computes the inward offset of the block contour and subdivides into lots the stripe between the block contour and its offset. The inward offset is computed with CGA. A set of sample points is computed along the offset. Consecutive points are separated by a distance computed as a function of the user-specified lot areas. Lines orthogonal to the offset at the sample points and passing through the sample points are used to split the stripe between the block contour and the offset.

### Skeleton subdivision algorithm

The skeleton subdivision algorithm uses the straight skeleton (below, top left) to identify the center lines of the block. Given a set of skeleton faces, you identify those whose street edges are adjacent and of a similar curvature

(below, top right). These faces are then grouped together. For each corner, the alignment priority (see [Corner Alignment](#)) determines how you assign the corner sections of these face-groups (below, bottom left). Finally, each of the face-groups are sliced in a direction perpendicular to their street edges to create lots, and small lots are merged together until they are larger than [Lot Area Min](#).



*Successive steps of the skeleton subdivision algorithm*

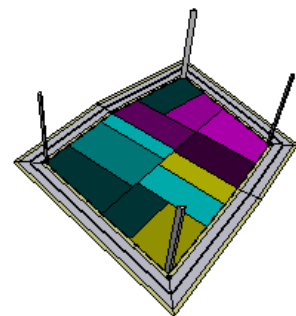
## Consistent Indexing

As a result of the recursive nature of the subdivision algorithm and the different criteria dictated by shape attributes, the ordering of the lots resulting from subdivision may significantly vary after an editing operation. This is particularly inconvenient if models have been generated inside the lots.

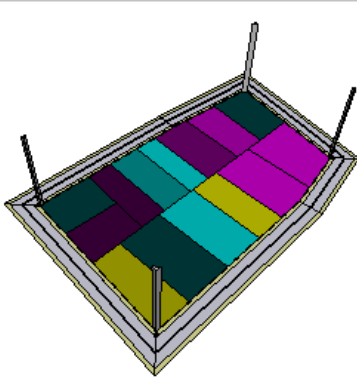
To improve the consistency in the lot indexing among two consecutive subdivisions, the algorithm computes the relative position of each lot for each one of the two subdivisions, using a metric based on generalized barycentric coordinates. Pairs of lots that are the closest to each other in this barycentric space, are assigned the same index.

The same approach is also used to improve the consistency of the seed of each lot. As a result, two lots that are relatively in the same position of the block at two different subdivision configurations, have higher chances of sharing the same seed and attributes. The figure below shows a subdivision together with the shapes generated from a grammar that assigns one of 15 possible random colors to each lot. Due to the consistency logic above, the colors of lots that have similar relative positions inside the block are preserved, even though the topology and geometry of the subdivisions are different as a result of an editing operation.

Subdivision for initial block

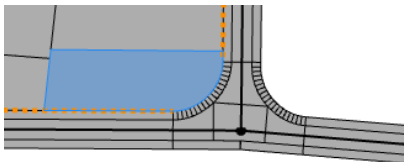


Subdivision for block after interactive editing



## Autogenerated street width attributes

For each resulting lot, an array of street width object attributes is generated.



A typical lot selected in the Viewport

Object Attributes	
Name	Value
streetWidth[0..22]	7.0;7.0;7.0;7.0;7.0;7.0;

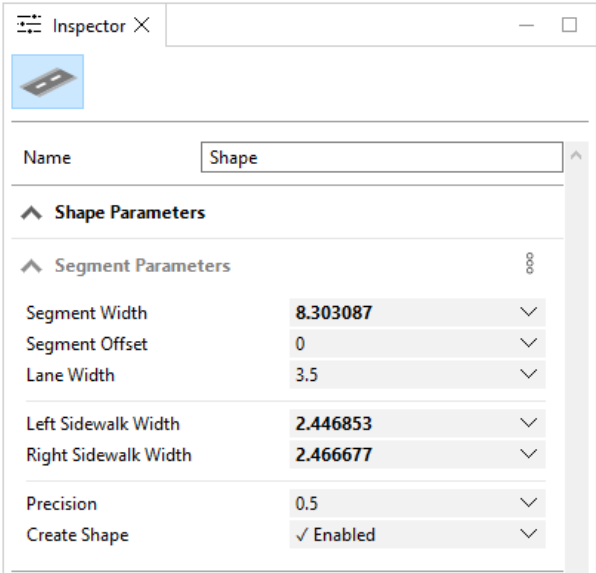
Street width attributes in the Inspector

### Note:

The first edge of a lot is the edge with maximal street width.

# Segment and sidewalk parameters

When segment shapes are selected, you manage segment parameters under **Segment Parameters** in the **Inspector**. These parameters define the [generated shapes](#).



Segment Parameters

## Note:

Parameters (attributes) can be mapped to Default, User, Object, or to a map layer. See [Mapping Attributes](#) for details.

## Segment parameters

The following parameters are available for the user to control the resulting street shapes:

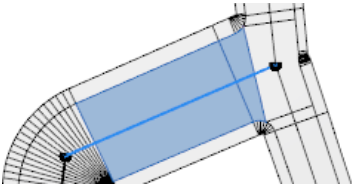
<b>Segment Width (attr streetWidth)</b>	Defines the width of the main street shape.
<b>Segment Offset (attr streetOffset)</b>	Defines the distance the geometry will be offset from the center line.
<b>Lane Width (attr laneWidth)</b>	Determines the width of lanes used for UV texture mapping of streets.
<b>Left Sidewalk Width (attr sidewalkWidthLeft)</b>	Defines the width of the left sidewalk; see Sidewalk parameters.
<b>Right Sidewalk Width (attr sidewalkWidthRight)</b>	Defines the width of the right sidewalk.
<b>Precision (attr precision)</b>	Graph nodes are interpolated using Bezier splines, resulting in curved streets. This parameter defines the spline sampling precision, in other words, 0 leads to a minimal and 1 to a maximal number of spline sampling points.
<b>Create Shape (attr shapeCreation)</b>	Enable or disable the shape geometry creation from the segment.

 **Note:**

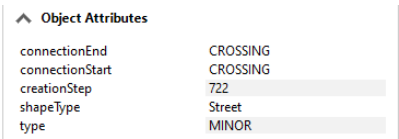
Street widths are given in absolute length units; precision is given in a normalized [0, 1] range.

## Autogenerated connection attributes

Connection attributes provide basic information about the underlying graph and give context information. CGA rules may want to access the following attributes.



A street shape selected in the Viewport



Street shape attributes in the Inspector

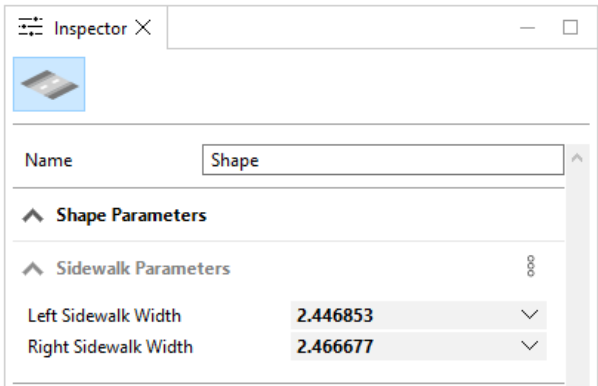
<b>connectionEnd</b>	Hints as to the adjacent geometry at the start or end of a street segment shape. Values include STREET, CROSSING, JUNCTION, JUNCTION_ENTRY, DEAD_END, FREEWAY, FREEWAY_ENTRY and ROUNDABOUT.
<b>connectionStart</b>	
<b>shapeType</b>	The shapeType is set to one of the following: <ul style="list-style-type: none"><li>• Street—Street shape</li><li>• Sidewalk—Sidewalk shape</li></ul>

 **Note:**

Connection attributes are object attributes of the graph segment and are inherited to the shape.

## Sidewalk parameters

When sidewalk shapes are selected, you can manage parameters under **Sidewalk Parameters** in the **Inspector**. See the [Segment parameters](#) section above for descriptions.



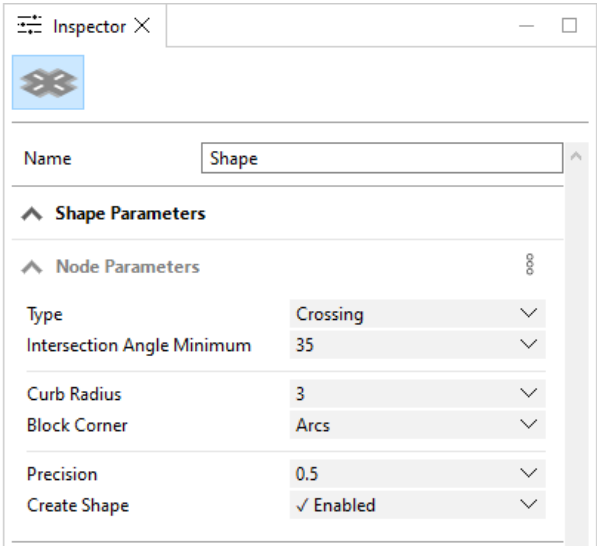
Sidewalk parameters

Under **Object Attributes**, the shapeType attribute is set to Sidewalk and the new sidewalkSide object attribute is added:

<div>^ Object Attributes</div> <div><div>connectionEnd</div><div>STREET</div></div> <div><div>connectionStart</div><div>CROSSING</div></div> <div><div>creationStep</div><div>148</div></div> <div><div>shapeType</div><div>Sidewalk</div></div> <div><div>sidewalkSide</div><div>Left</div></div> <div><div>type</div><div>MAJOR</div></div>	
<b>sidewalkSide</b>	Which side of the street this sidewalk shape is on, relative to the street direction: either Left or Right. SidewalkSide is only added to sidewalk shapes.

# Node parameters

Node parameters can be individually set for each graph node. The parameters for node shapes are defined under **Node Parameters** in the **Inspector**.



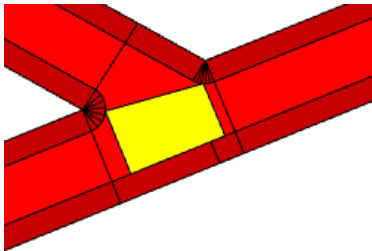
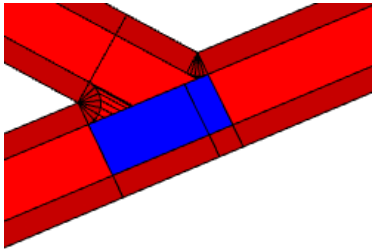
Node Parameters

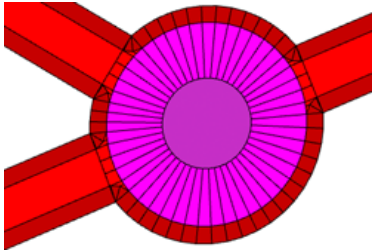
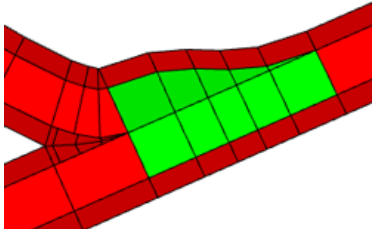
 **Note:**

Parameters (attributes) can be mapped to Default, User, Object, or to a map layer. See [Mapping Attributes](#) for details.

## Node type parameter

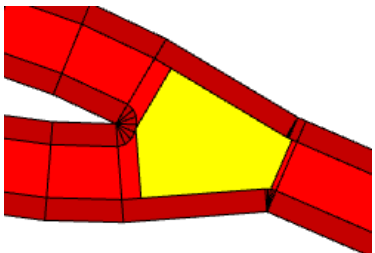
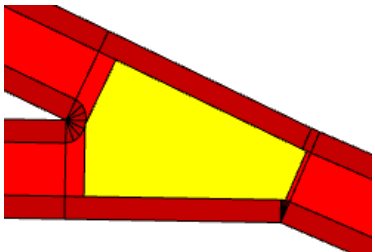
Several parameters are available for the user to control the resulting node shapes. Node parameters define the type of the node and specify geometry details such as the radius of the arcs.

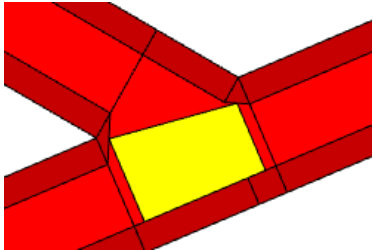
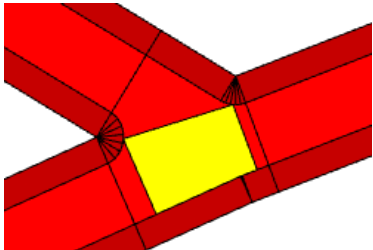
Type (attr type)	
Specifies the type of the node. Crossing, Junction, Roundabout, or Freeway.	
Crossing	
Junction	

Roundabout	
Freeway	

General node parameters

The following tables describe the parameters that are common to all the node types:

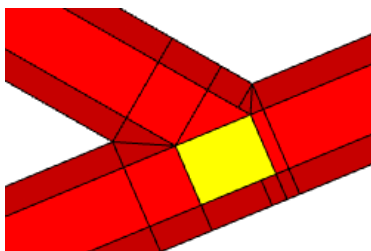
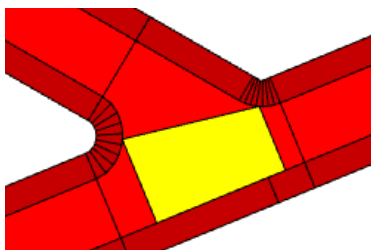
<b>Intersection Angle Minimum (attr angleThreshold)</b>	
Minimum angle between streets before they automatically start bending to avoid each other. It is ignored for freeways.	
Crossing with Intersection Angle Minimum set to 30. Note the streets bending to avoid each other.	
Crossing with Intersection Angle Minimum set to 10.	
<b>Precision (attr precision)</b>	
Specifies the level of detail. Value must be in the range [0, 1].	

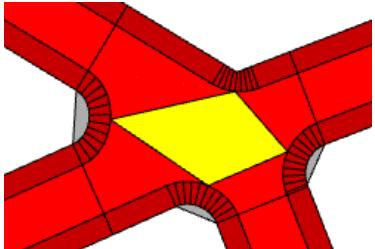
Crossing shapes with Precision = 0.1.	
Crossing shapes with Precision = 0.3.	

<b>Create Shape (attr shapeCreation)</b>
Enable or disable the shape geometry creation from the node.

### Crossing and junction parameters

The Crossing, Freeway, and Junction parameters have the following parameters:

<b>Curb Radius (attr minArcRadius)</b>	
The minimal arc radius. For freeways, a higher value (>20) is better suited.	
Given in absolute length units.	
Crossing shapes with Curb Radius = 0.	
Crossing shapes with Curb Radius = 5.	
<b>Block Corner (attr cornerStyle)</b>	
Either Arcs or Straight. When set to the latter, blocks get simpler.	

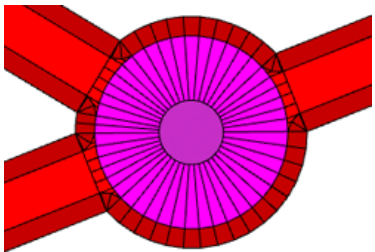
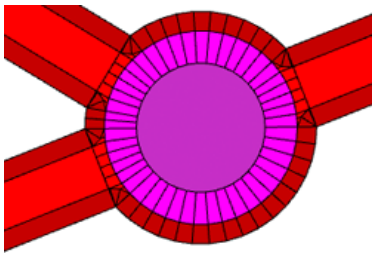
Border with Block Corner set to Arcs.	
Border with Block Corner set to Straight.	

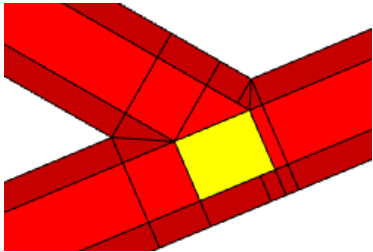
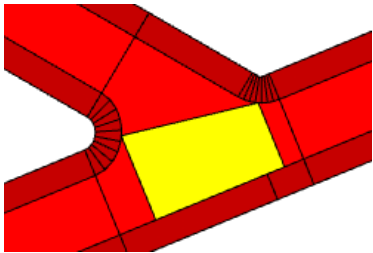
 **Note:**

The Block Corner parameter is not available when Type is set to Freeway.

## Roundabout parameters

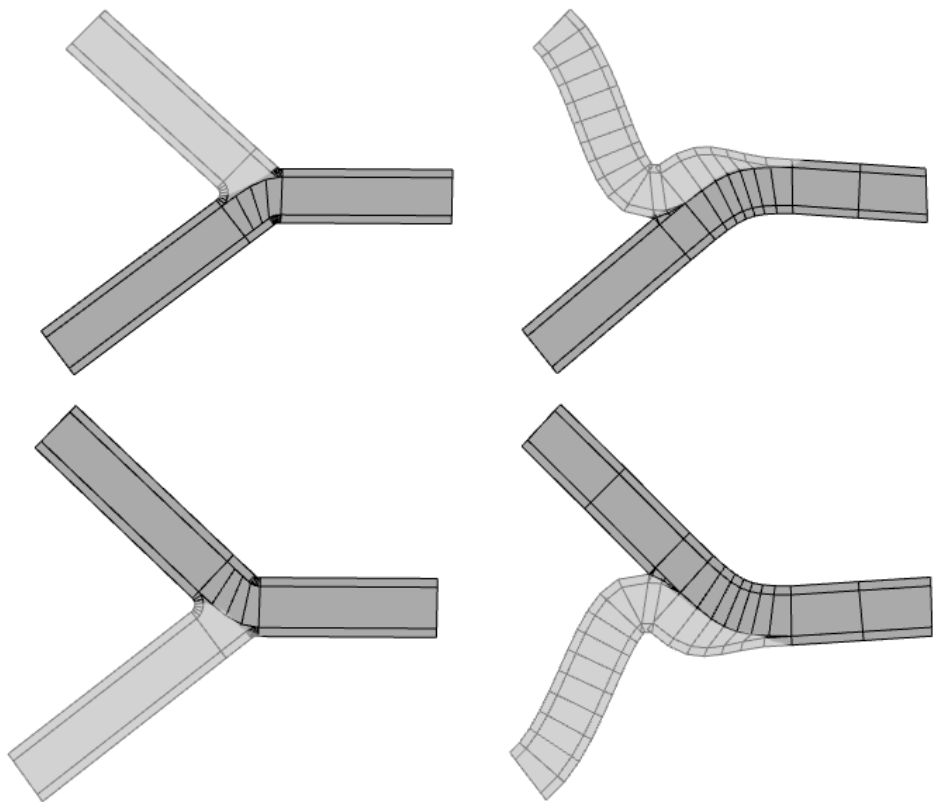
When Type is set to Roundabout, the roundabout creation uses the following parameters:

<b>Roundabout Island Radius (attr innerRadius)</b>	
Defines the radius of the inner circle (the island shape).	
<b>Roundabout Street Width (attr streetWidth)</b>	
Defines the width of the roundabout street lane.	
A roundabout with Roundabout Island Radius = 5 and Roundabout Street Width = 10.	
A roundabout with Roundabout Island Radius = 10 and Roundabout Street Width = 5.	

Curb Radius (attr minArcRadius)	
The minimal arc radius. For freeways, a higher value (>20) is better suited.	
Given in absolute length units.	
Crossing shapes with Curb Radius = 0.	
Crossing shapes with Curb Radius = 5.	

## Principal street selection

The Junction and Freeway node types make use of the principal street to determine the node geometry.


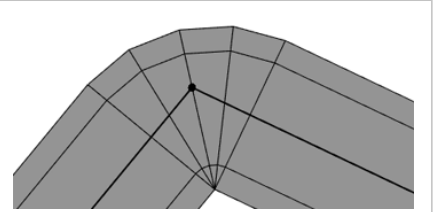
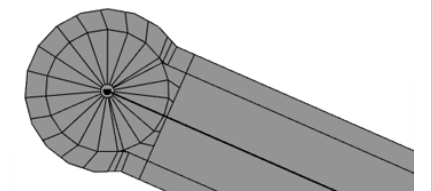


Left: Two junctions with different principal streets. Right: Two freeway intersections with different principal streets. In each case, the principal streets are drawn in a darker shade of gray.

 **Note:**


The principal street is specified using either the [street tool](#), or by setting the object attribute `principleStreetStart` or `principleStreetEnd` on adjacent streets as appropriate.

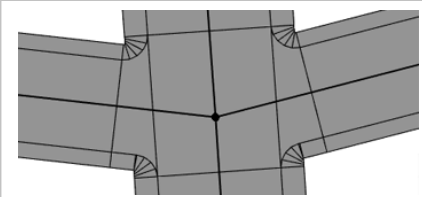
## Examples

Simple curve	
Valence-two nodes (nodes between two graph segments) usually lead to curves or links between the segments.  <b>Note:</b> For valence-one (nodes at the end of a row of segments / cul-de-sac) or valence-two nodes, it does not matter whether the type is <code>Crossing</code> or <code>Junction</code> .	
Dead end street	
By setting the type of a valence-one node to <code>roundabout</code> , you can model a cul-de-sac.	

### Junction

Junctions, as opposed to crossings, don't break a major street. Minor streets are connected to the major street by junction entries.


 **Note:**  
The two segments with the maximal street widths are automatically treated as a major street.



## Autogenerated connection attributes

Connection attributes provide basic information about the underlying graph and give context information. CGA rules may want to access the following attributes.

<b>shapeType</b>	Specifies the type of the node. Crossing, Junction, Roundabout, or Freeway.
<b>valency</b>	The number of street segments adjacent to a street node. Valency is added to all node shapes.

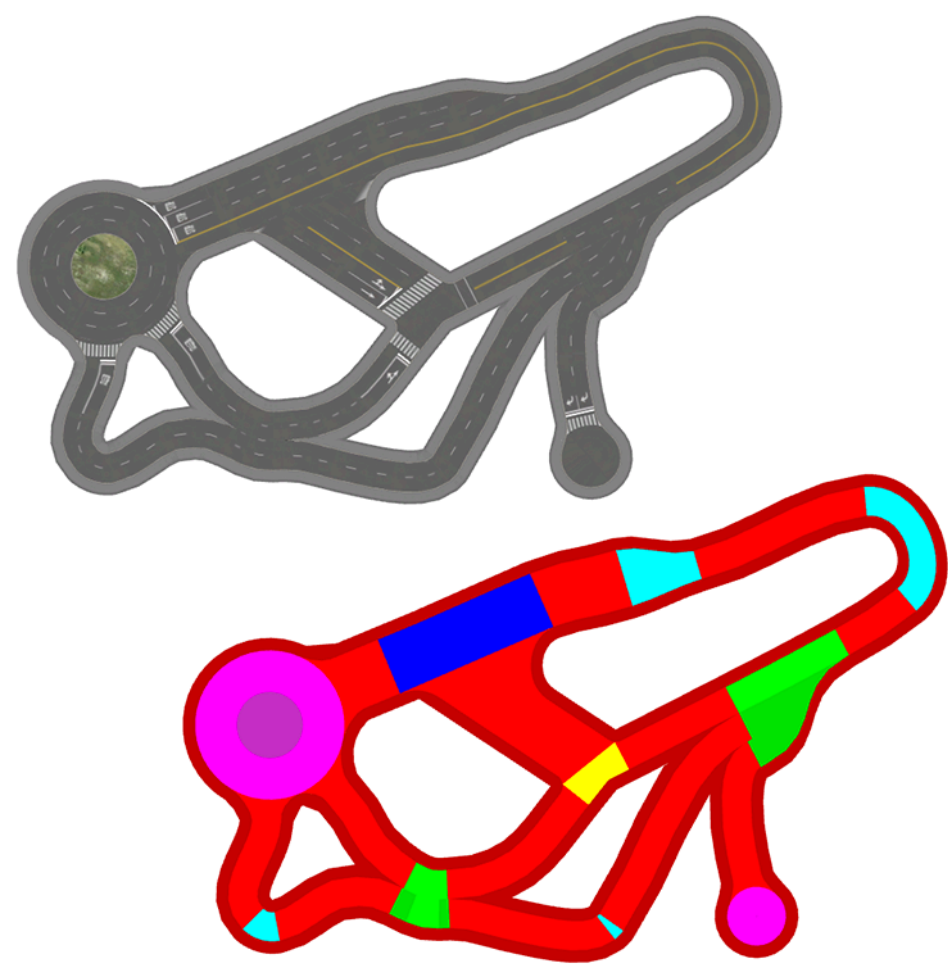
 **Note:**  
Connection attributes are object attributes of the graph node and are inherited to the shape, indicated by the italic font in the shape's **Inspector**.







# Street and intersection shapes




Shapes generated by nodes and segments have a different type (attribute shapeType) assigned. Each shape type is associated with a start rule.

## Default start rules for street shapes

Standard start rules are Street, Sidewalk, Crossing, Junction, Freeway, FreewayEntry, Roundabout, RoundaboutIsland, and Joint.



Color	Shape start rule	Created by
	Street	Segment
	Sidewalk	Segment, and all node types
	Joint	Nodes with only two adjacent streets
	Crossing	Crossing node type
	Junction	Junction node type
	Freeway	Freeway node type

Color	Shape start rule	Created by
	Freeway Entry	Freeway node type
	Roundabout	Roundabout node type
	Roundabout Island	Roundabout node type

 **Note:**

- Shapes have by default no rule file assigned. Therefore, if you like to work with these default start rules, you have to define the CGA rules `Street`, `Sidewalk`, `Crossing`, `Junction`, `Freeway`, `FreewayEntry`, `Roundabout`, `RoundaboutIsland`, and `Joint`. These rules will be the starting point for geometry generation.
- For information on the Block default start rules, see [Block parameters](#).

## Street shape UV values

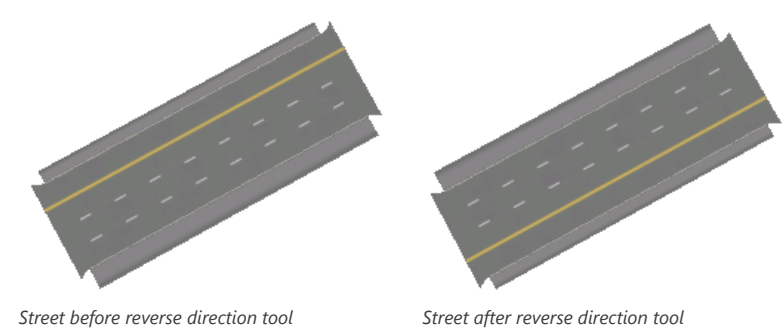
For details of the UV coordinates, see [Street and intersection shape UV](#).

## Reset shape attributes

Since start rules can be overridden, users may want to use the **Reset Shape Attributes** tool to revert the start rules to the default values. To run the tool, select a set of shapes and choose **Graph > Reset Shape Attributes** in the main menu.

## Reverse street direction

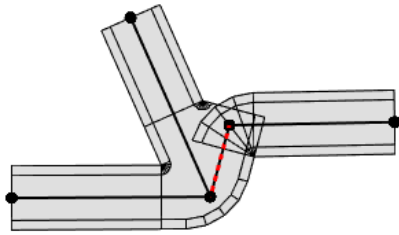
You can reverse the direction of your streets, including their sidewalks, in a CityEngine scene. Select a single or multiple street segments and click **Graph > Reverse Direction** in the main menu to change the direction.



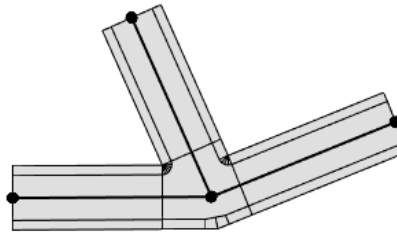
## Conflicts

Because the internal shape creation algorithm computes the node shapes individually, conflicts can occur. Usually, conflicts occur when the distance between two nodes is very small. In this case, at least one node is located inside the shape of a neighbor node.

Conflicting segments are marked with a red dashed line. The error color can be changed in the [Viewport preferences](#).



*There is a conflict (red dashed line) because a street node is inside a crossing.*



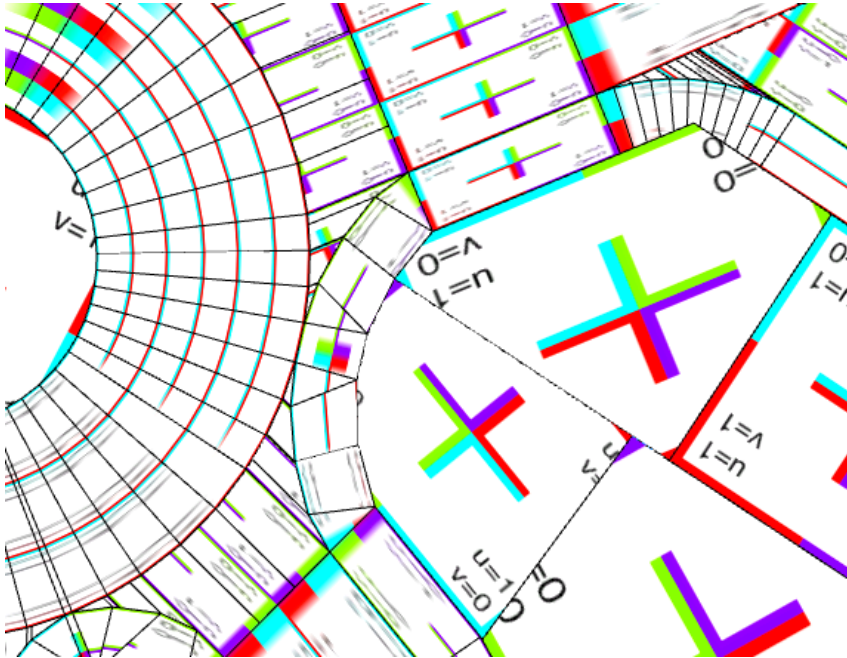
*The conflict has been resolved by running the Graph Cleanup tool.*

To resolve conflicts, you can choose from the following actions:

- Clean up the graph; see [Graph Cleanup](#) tool
- Edit the graph network; see [Street creation tools](#)
- Change shape creation parameters (in other words, street width); see [Segment and sidewalk parameters](#) and [Node parameters](#).
- Use the [Edit graph network](#) tool to edit curves, change street widths, or move nodes.

# Street and intersection shape UV

UV coordinates are generated for each shape. They can be used for [UV Splits](#) and texturing. Up to three UV sets are supplied for each shape to describe different surface parameterizations.



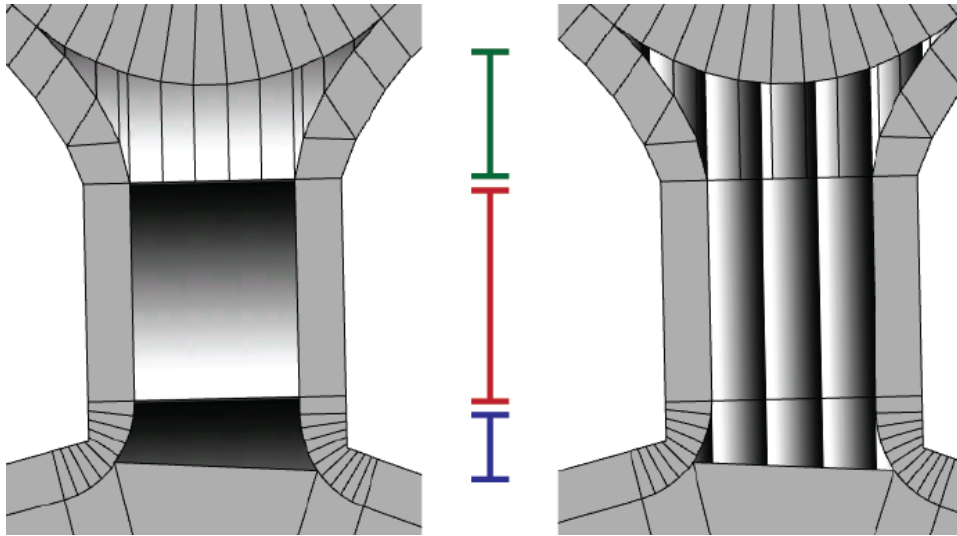
An example of generated UV coordinates for street, intersection, and lot shapes is shown.

## Street UVs

There are three sets of UVs for street shapes.

### Street UV Set 0

The first UV set provides a set of street lanes. The central region of the geometry is normalized along its length from 0 to 1, and across its width by 0 to the number of lanes. The number of lanes is specified by the [street parameters](#). In the following images, you see a street shape between a crossing (bottom) and roundabout (top). The left image illustrates the U values using shades of gray, while the right image illustrates the V values.

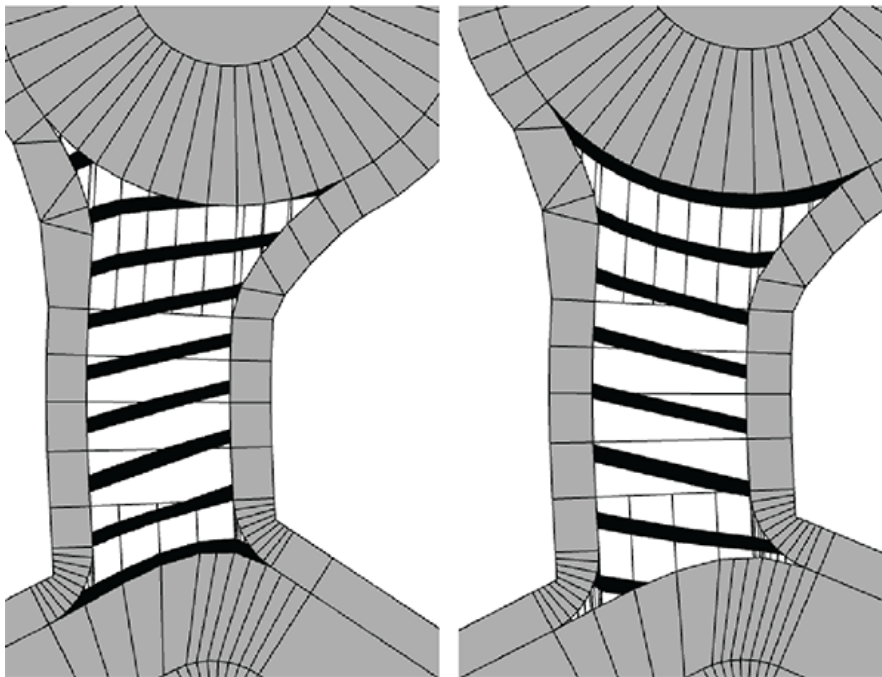


Street UV set 0

The entry and exit of a street shape are parameterized to blend with the central region. For example, the entry (above, blue) and exit (green) u-values are oriented and scaled to match the central region (red). The entry has negative U values, and the exit has U values greater than 1.

## Street UV Set 1 and 2

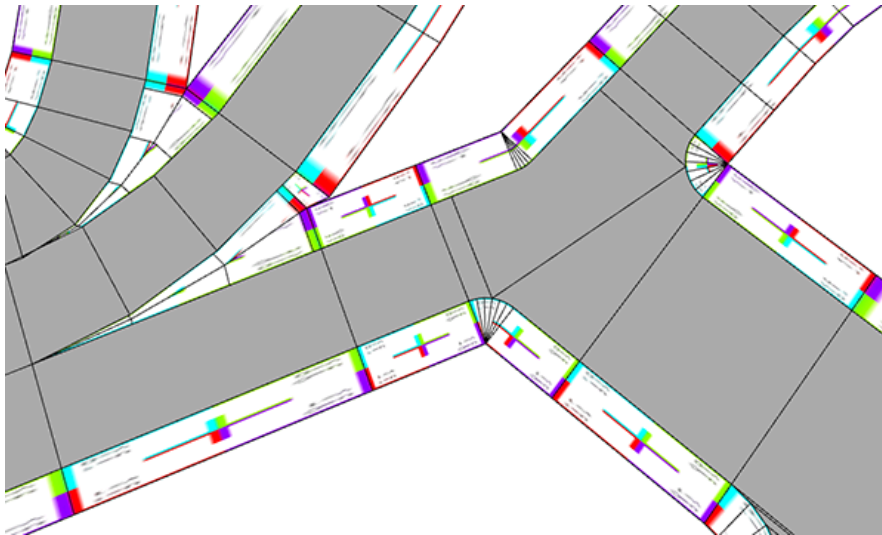
UV Sets 1 and 2 provide distance fields from the start of the entry and end of the exit, respectively. This information is provided in the U channel. The V channel is undefined. In the following image, the street direction is from the bottom to the top, and shows the orientation of UV Set 1 (left) and UV Set 2 (right).



Street UV Set 1 and 2

## Sidewalk shapes UV Set 0

The sidewalk shapes only provide a UV set 0. This is stitched to the street-side edge of the geometry; all street-adjacent edges have v-values of 0.



*Sidewalk shapes UV Set 0*

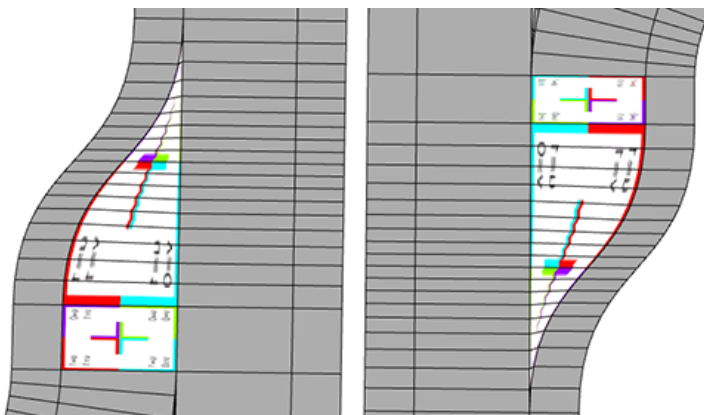
## Intersection UVs

Intersection shapes that are street-like use the same UV set 0 parameterization as streets, without entries or exits. These shapes are Joint, Junction, Roundabout, Freeway, and FreewayEntry. In general, intersection shapes do not have other UV sets, with the exception of FreewayEntry. Similarly, intersection sidewalk shapes use the same UV set 0 parameterization as street sidewalk shapes.

The remaining shapes use a rectilinear projected UV set 0. These shapes are RoundaboutIsland and Crossing.

## Freeway entries

Freeway entries supply a UV set 1 to identify the inside edge shape that adjoins another street shape. The edge  $V = 0$  is always on the inside of the freeway intersection.



*Freeway entries*

## Work with static models

A static model is any geometrical model imported as is, without modification by CGA rules, textures, or individual vertices within CityEngine. Static models can be moved, scaled, and rotated, but the geometry and the textures cannot be edited. This makes static models a good choice for landmark or hero buildings in a scene.

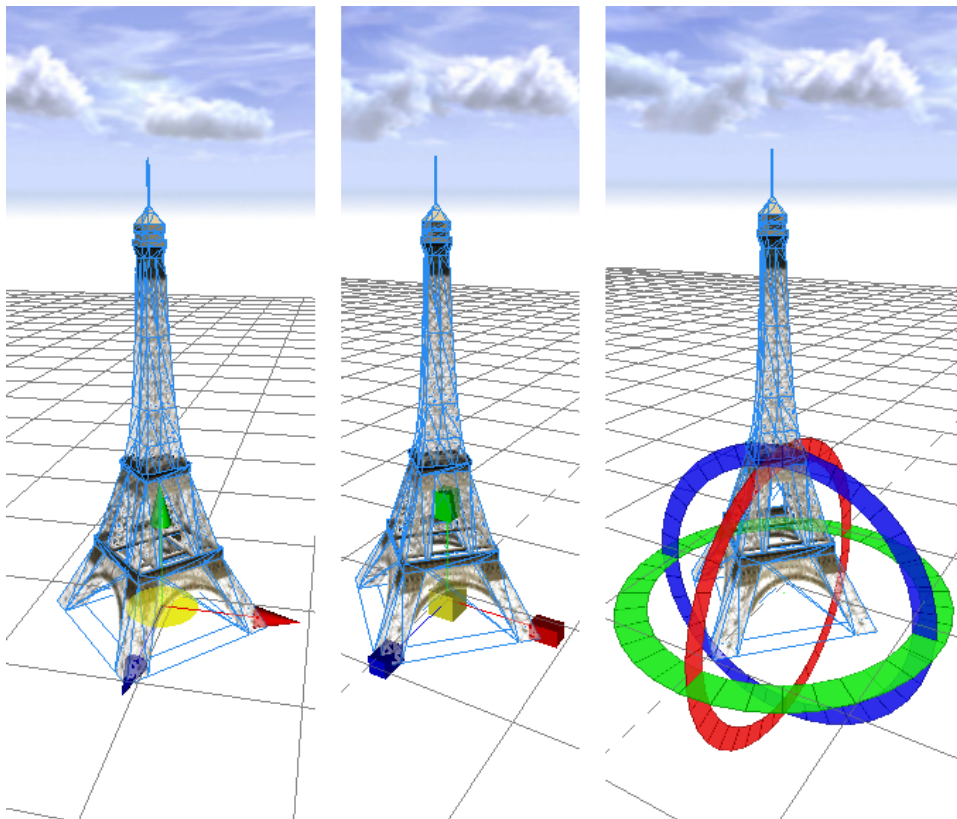
### Note:

- Since it isn't possible to apply CGA rules to static models use [shapes](#) instead. Static models are imported from formats such as OBJ, DAE, FBX, glTF, USD, and KMZ/KML.
- After importing, static models are linked to the source file. Consequently, renaming or moving the source file breaks the reference in the scene. In this case, CityEngine replaces the missing model with a white generic cube. Select it and check the **Asset\_file** entry in the **Inspector** to fix the reference.

## Place a static model

A static model can be positioned and scaled within the scene by tusing the **Move Tool (W)** , **Scale Tool (E)**, and the **Rotate Tool (R)**.

The origin of the model is centered and bottom-aligned (except for KML import). This can be changed in the [Inspector](#).



Move, Scale, Rotate tools. Note the bottom-aligned point of attack / origin of the tools.

## Settings in the Inspector

As static models are referenced only, the adjustments made here are not to be saved in source data. The settings are saved in the scene and reapplied to the source when reopening the scene.

<b>Asset_File</b>	Location of the asset in the workspace. This is especially useful if the reference to the asset got broken.
<b>Material_Colorize</b>	<ul style="list-style-type: none"> <li>Colors the model to increase visibility of an object in the scene.</li> <li>Setting it to white (#ffffff) turns tinting off.</li> </ul>
<b>Material_Transparency</b>	Controls the transparency of the model.
<b>Mesh</b>	Cleans up and reduces the geometry. It can help to improve the frame rate in the viewport.
<b>Normals</b>	Controls the shading of the model. Choose between all soft or all hard edges in the model. Auto determines the hardness of an edge based on the angle between the adjacent faces and the hardness of adjacent edges. By default, the settings as defined in the source data are (keep original).
<b>Pos_Bottom_Align</b>	Places the origin of the model to the bottom or to the center of the model.
<b>Pos_Center</b>	Places the origin of the model to the center or to the location as it is defined in the source file. If <b>Pos_Bottom_Align</b> is set, the vertical center is overridden (default, except for KML import).
<b>Pos_zUp</b>	Some models are created in a z-up coordinate system instead of the CityEngine y-up system. Use this switch to correct this.

## Align static models terrain

Static models alignment is a tool to align static models to arbitrary terrains (map layers with attribute elevation defined) or to the y=0 level. All currently selected static models and all static models of the selected layers are aligned.

1. Select the models you want to align to a terrain
2. Click **Terrains > Align Static Models to Terrain**.

### Parameters

<b>Heightmap</b>	The terrain to align the shapes to. All terrain layers and attribute layers with an attribute named elevation are listed here.
<b>Offset</b>	Specifies how far the model is placed above or below after alignment.

#### **Note:**

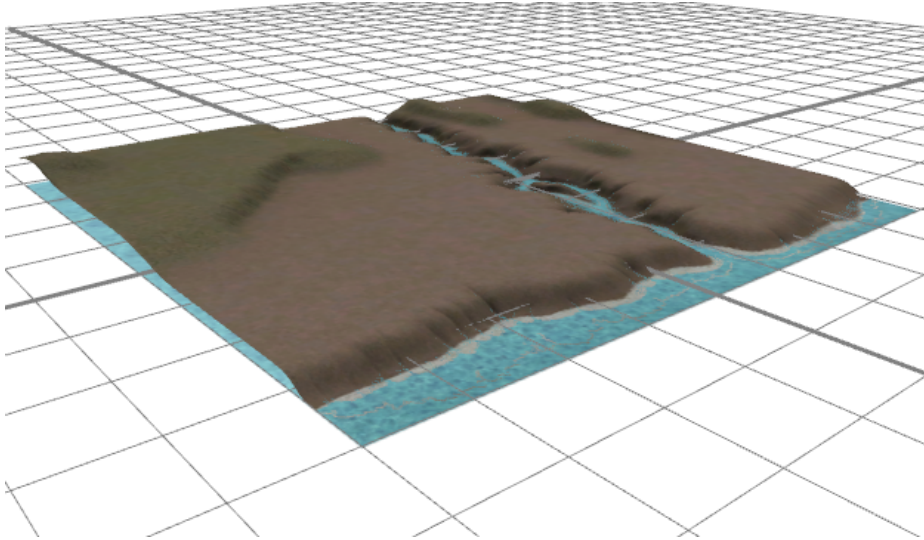
The model is not draped onto the terrain. The point of origin only is placed exactly onto the terrain (offset not included). Therefore, other parts of the model may disappear in the terrain or hover above it.

# Map Layer

# Map layers

A map layer is used mainly to add terrain and satellite imagery to a scene. Additionally, you can use a map layer to control the [automatic street creation tool](#) and to control CGA rule attributes.

- You select a map layer in the [Scene Editor](#). They are not selectable in the [Viewport](#).
- You can move and scale a map layer but not rotate.



*Terrain map layer with a height map used to generate the terrain mesh with texture image for colorizing*

## Map layer types

The following are common layer types:

- [Terrain layer](#)—Creates a textured terrain based on a heightmap image and texture image. This is the most common map layer type.
- [Texture layer](#)—Creates a flat horizontal plane in a scene. This layer type is useful for water bodies.

## Auxiliary map layer types

The following layer types are used to influence the behavior of procedural functions:

- [Obstacle layer](#)—Defines the areas where the street grow algorithms are prohibited to create streets. This layer type can also be used to select objects based on their location on the map layer. An obstacle layer includes a true and false information for each location on the map.
- [Mapping layer](#)—An arbitrary combination of image map channels and mathematical functions. This layer type is typically used to control the height or usage type of a building based on its location. A map layer can include multiple values for each location on the map.
- [Function layer](#)—Instead of an image, this layer type takes a mathematical function to generate information for each location on the map.

# Terrain layer



The terrain layer is a special map layer that visualizes the elevation of the scene topography using image data. It also serves as reference elevation for align operations for scene objects such as shapes or graph nodes.

## Create a terrain layer

A terrain layer can be created in the following ways:

- Drag an image from the [Navigator](#) into the scene.
- Import a terrain with **File > Import > Terrain Import**.
- Create a terrain layer with **Layer > New Map Layer > Terrain** .

## Terrain options

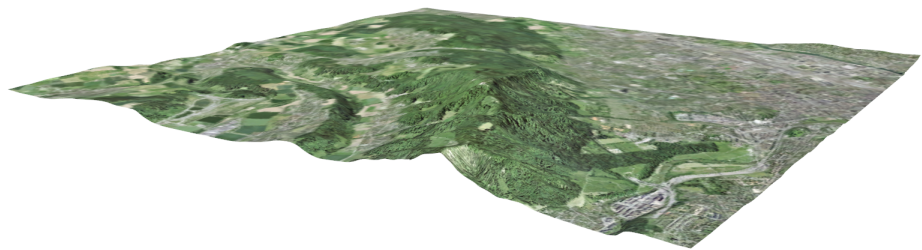
Heightmap File	Choose an image from your workspace that will be used as a heightmap mesh / digital elevation model. Normally, this is a grayscale image. When a <a href="#">georeferenced image</a> is selected, elevation and bounds are set automatically.
Texture File	Choose an image from your workspace that will be used to texture the heightmap mesh.  <b>Note:</b> The texture file is applied over the exact extent of the heightmap mesh. It is therefore important that your texture file has the same extent as the heightmap file.
Channel	Choose the source channel from the image that is used to read the data for the elevation. For most images, brightness is the best choice.
Min. elevation	The minimum value / the lower bound for the elevation in meters. For dedicated file types such as GeoTIFF, this value is read automatically from the file.
Max. elevation	The maximum value / the upper bound for the elevation in meters. For dedicated filetypes such as GeoTIFF, this value is read automatically from the file.
Bounds Dimensions	The Width and Height of the resulting terrain in meters. When a georeferenced image is selected, this value is set automatically.
Bounds Location	The location of the resulting terrain in meters. When a georeferenced image is selected, this value is set automatically.  <b>Note:</b> <ul style="list-style-type: none"><li>• The button right of <b>Location</b> can be used to change the reference point of the terrain's position.</li><li>• It is recommended that you use 32-bit float GeoTIFF files.</li><li>• In addition to 8-bit images, 16-bit and 32-bit images are also supported for heightmap files. The 16/32-bit range is scaled to the elevation bounds similar as is done with standard images.</li></ul>

## Terrain layer in a CityEngine scene

The new terrain is added as a new layer in the [Scene Editor](#). If the terrain is not visible in the [Viewport](#), right-click the terrain layer and choose **Frame Layer**.

 **Note:**

Terrain layers (like all map layers) cannot be selected in the [Viewport](#) directly, but only through the [Scene Editor](#).



A terrain layer is shown.

## Inspector options

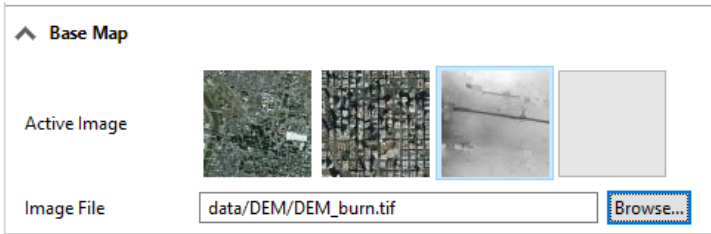
To change the options of a terrain, you can select the terrain layer in the [Scene Editor](#) and review its attributes in the **Inspector**, with the following options:

<b>Name</b>	Layer name
<b>Visibility</b>	Layer visibility
<b>Locked</b>	Lock terrain layer
<b>Transparency</b>	Layer transparency
<b>Color</b>	Color value multiplied onto layer
<b>Wireframe</b>	Wireframe enabled or disabled
<b>Dimensions</b>	Extent of layer
<b>Location</b>	Location of layer
<b>Elevation Offset</b>	Elevation offset applied to layer. This value is also available as a built-in function in the <b>Layer Attributes</b> code.
<b>Heightmap File</b>	Heightmap file location
<b>Apply Alignments</b>	If enabled, the built-in function elevationDelta in the <b>Layer Attributes</b> code returns the elevation deltas resulting from the <a href="#">terrain alignment tool</a> . If disabled, the function returns 0.
<b>Heightmap Sampling</b>	<b>Bilinear</b> and <b>Nearest Neighbor (hard edges)</b> determine the smoothness of the elevation.

<b>Minimum / Maximum Height</b>	<ul style="list-style-type: none"><li>• The elevation data from the image is mapped into the range [Minimum, Maximum].</li><li>• Both values are also available as built-in functions in the <b>Layer Attributes</b> code.</li></ul>
---------------------------------	--

## Basemap

Select the image you want active as your basemap. Click the blank active image to add another basemap.



Base map properties in Inspector are shown.

Under **Layer Attributes** in the **Inspector** window, you can modify the (editable) CGA code of the elevation.

## Overlapping terrain layers

A common scene setup is to have a small but high-resolution terrain layer for your study area and a second overlapping larger but low-resolution terrain layer for the surrounding area.

By default, multiple terrain layers are masked in overlapping regions, such that only the terrain with the highest resolution is shown. This masking avoids blurry artifacts in the overlapping regions stemming from the lower-resolution terrain piercing through the higher-resolution terrain due to the resolution difference.

To disable terrain masking, click **View settings**  > **Terrain Masking** in the [Viewport](#).

## Export terrains

You can export terrains as tile packages (TPK), images, or geometries by selecting the map layer or layers to export.

### Export terrains to a tile package

To export terrains to TPK, do the following:

1. Click **File** > **Export** > **CityEngine** > **Export Selected Layers as TPK**.
2. Choose to export both the basemap and elevation terrain files as TPK files or just select one of them.
  - For **Scene Environment**, you have the option of exporting the terrain for either a global or local scene.
  - If you choose to export both basemap and elevation maps, you will have a {filename}\_Basemap.tpk and a {filename}\_Elevation.tpk file in your output folder.
3. Sign in to ArcGIS Online or ArcGIS Enterprise to share the .tpk files with your organization or the public.
4. Right-click the .tpk file you want to share and click **Share as**.  
The share **Tile Package** dialog box appears.
5. Fill in the necessary fields and click **Share**.

Go to the Content tab in your account in ArcGIS Online or ArcGIS Enterprise and publish the TPK as a hosted elevation or tile layer.

## Export terrains as an image

To export to image files, select **File > Export > CityEngine > Export Selected Terrains as Image** and choose a format and resolution. Refer to [Export terrain](#) for more information.

## Export terrains to geometry files

To export to geometry files, select **File > Export > CityEngine > Export Models of Selected Shapes and Terrain Layers** and choose a format. The rest of the process is similar to the [model exporter](#).



### **Note:**

The KML and the Esri Scene Layer Package formats do not support exporting terrains.

# Texture layer

The texture layer is a special map layer that adds an image as a flat map to a scene.

## Create a texture layer


A texture layer can be created in the following ways:

- Import a texture layer by clicking **File > Import > Texture Import** in the main menu.
- Click **Layer > New Map Layer > Texture** in the main menu.

The **Texture** dialog box appears.

## Texture layer options

You can modify the following options on the **Texture** dialog box:

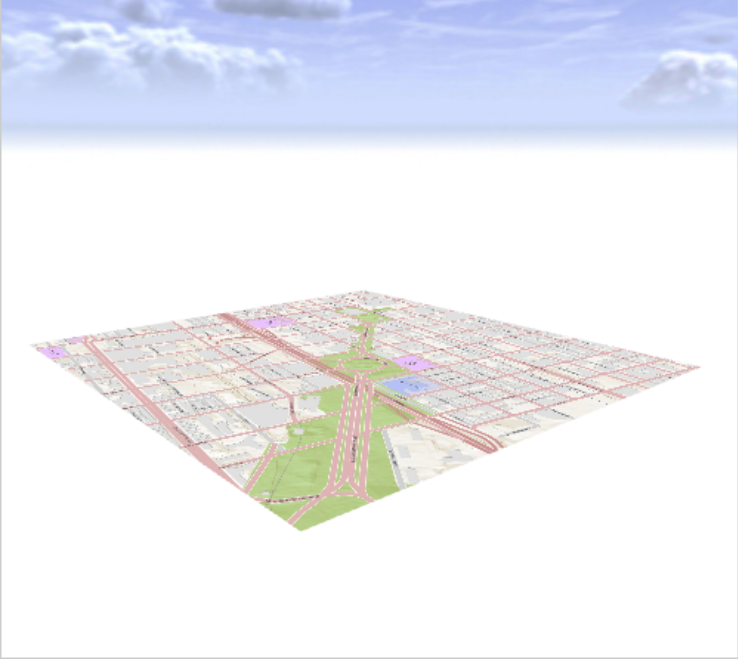
<b>Texture File</b>	Choose an image from your workspace. When a <a href="#">georeferenced image</a> is selected, bounds are set automatically.
<b>Bounds Dimensions</b>	The Width and Height of the resulting texture in meters. When a georeferenced image is selected, this value is set automatically.
<b>Bounds Location</b>	The location of the resulting texture in meters. When a <a href="#">georeferenced image</a> is selected, this value is set automatically.  <b>Note:</b> The button right of <b>Location</b> can be used to change the reference point of the texture's position.

## Texture layer in a CityEngine scene

The new texture is added as a new layer in the [Scene Editor](#). If the texture is not visible in the [Viewport](#), right-click the texture layer and choose **Frame Layer**.

 **Note:**

All map layers, such as texture layers, aren't available for selection in the [Viewport](#) directly. Select texture layers through the [Scene Editor](#).



A texture layer is shown.

## Inspector options

You can select a texture layer in the [Scene Editor](#) to change the following parameters in the [Inspector](#):

<b>Name</b>	Layer name
<b>Visibility</b>	Layer visibility
<b>Locked</b>	Lock terrain layer
<b>Transparency</b>	Layer transparency
<b>Color</b>	Color value multiplied onto a layer
<b>Wireframe</b>	Wireframe enabled or disabled
<b>Dimensions</b>	Extent of layer
<b>Location</b>	Location of layer
<b>Elevation Offset</b>	Elevation offset applied to a layer. This value is also available as a built-in function in the <b>Layer Attributes</b> code.
<b>Heightmap File</b>	Heightmap file location
<b>Resolution</b>	The number of terrain mesh vertices in u and V direction.
<b>Apply Alignments</b>	If enabled, the built-in function elevationDelta in the <b>Layer Attributes</b> code returns the elevation deltas resulting from the <a href="#">terrain alignment tool</a> . If disabled, the function returns 0.
<b>Heightmap Sampling</b>	<b>Bilinear</b> and <b>Nearest Neighbor (hard edges)</b> determine the smoothness of the image.

<b>Minimum / Maximum Height</b>	<ul style="list-style-type: none"><li>• The elevation data from the image is mapped into the range [Minimum, Maximum].</li><li>• Both values are also available as built-in functions in the <b>Layer Attributes</b> code.</li></ul>
<b>Layer Attributes</b>	Has no effect on the rendered texture.

Basemap

Select the image you want active as your basemap. Click the blank active image to add another basemap.

Base Map

Active Image




Image File

data/DEM/DEM\_burn.tif

Browse...

Basemap properties in Inspector are shown.

# Obstacle layer

The obstacle map defines a Boolean attribute that guides the creation of street networks.


## Obstacle map

A common usage of the obstacle map is to create a land-water map in which the water is marked in a dark color (in other words, brightness below 0.5) and the land in a bright color. The obstacle attribute layer can then be selected in the automatic street generation wizard and streets are generated accordingly (in other words, no streets in dark regions).

You can create an obstacle map by clicking **Layer > New Map Layer** in the main-menu, and choosing **Obstacle**.

## Obstacle layer options

You can modify the following options when you open the **Obstacle** dialog box:

<b>Obstacle file</b>	Choose an image from your workspace that will be used as the obstacle in your scene.
<b>Channel</b>	Choose the source channel from the image that is used to read the data for the elevation. For most images, <b>brightness</b> is the best choice.
<b>Obstacle Threshold</b>	The <b>Obstacle Threshold</b> defines the image brightness, which distinguishes between obstacle and nonobstacle. For example, if you set the <b>Obstacle Threshold</b> to 0.5 in the <b>brightness</b> channel, any cell value below 0.5 is considered dark, and any cell value above is bright and part of the obstacle.
<b>Bounds Dimensions</b>	The <b>Width</b> and <b>Height</b> of the resulting texture in meters. When a <b>georeferenced image</b> is selected, this value is set automatically.
<b>Bounds Location</b>	The location of the resulting texture in meters. When a georeferenced image is selected, this value is set automatically.  <b>Note:</b> The button right of <b>Location</b> can be used to change the reference point of the obstacle's position.

## Create an obstacle layer in a CityEngine scene

After the obstacle type has been selected in the wizard, the obstacle attribute layer can be created as follows:

1. Browse your project and select the image map.

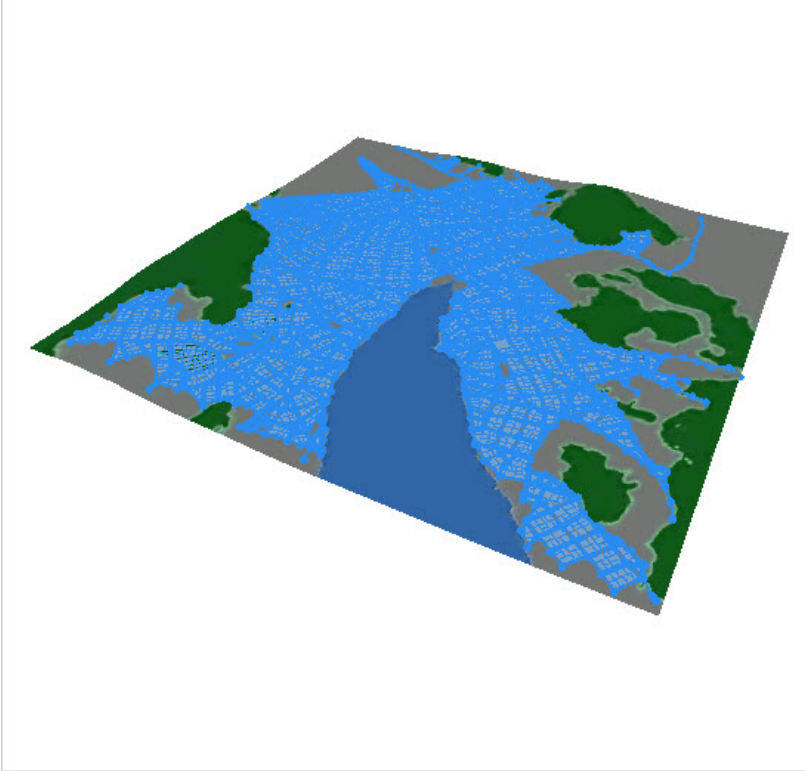
 **Note:**

Only image maps in the workspace can be selected. Therefore, the image has to be copied or imported into the corresponding project folder (typically maps).

2. Select the source channel and the corresponding threshold. Values below the threshold are interpreted as obstacles.
3. Set the dimensions and location of the attribute layer in the scene.

4. Click **Finish**.

An obstacle attribute layer is selected where the channel threshold function is represented as a function returning a Boolean function. This function and all other attribute layer properties can still be edited after creation.



*A Street network generation is controlled by an obstacle map.*

 **Note:**

All attribute layer functions that evaluate a Boolean value such as the obstacle layer can also be used for selecting objects in the scene by using the **Select > Select by Map Layer** menu. See also [Selection with image maps](#).

# Mapping layer

The mapping attribute layer wizard allows the most generic form of attribute definition in which you can create mappings from an image file to attribute values.

Typically, mapping attribute layers are used to control CGA shape grammar rule attributes; if an attribute defined in an attribute layer matches an attribute defined in a rule file, the attribute layer can be selected as a source for the rule attribute in the [Inspector](#). See [Map image data to rule attributes](#).

To create a mapping layer, do the following:

1. Select **Layer > New Map Layer**.
2. Click **Mapping**.
3. Select the image map in the wizard.
4. Set the dimensions and location of the attribute layer in the scene.
5. Click the upper left insert-row button (left of **Attribute**) to create an attribute.  
An arbitrary number of mappings can be defined per layer.
6. Right-click in the attribute area and select **Add Row** to create an attribute.
7. Enter the attribute name.
8. Select the source channel and the mapping range.
9. Click **Finish**.

The result is a mapping attribute layer with the named attribute function that returns values between **Minimum** and **Maximum** by sampling the given Channel from the image map file.

**Note:**

All attribute layer properties can still be changed after creation through the **Inspector**.

# Function layer

The function layer wizard allows you to create the most generic form of an attribute layer. You can write any mathematical function by using a subset of the CGA Shape Grammar language.

You can create a function layer in the following ways:

- From the main menu, click **Layer > New Map Layer**.
- From the **Scene Editor**, right-click **New > New Map Layer**.

After the **New Map Layer** dialog box appears, click **Function**.

After the function type has been selected in the wizard, the function attribute layer can be created in the function dialog box by entering a function that defines an arbitrary attribute. The "u" and "v" parameter correspond to normalized coordinates in the range [0..1] in x and z direction. After creation, the dimensions and the location of the layer can be edited as well (resulting in a scaling and translation of u and v accordingly).

The syntax is similar to the shape grammar and described in more detail in [Edit map layer attributes](#).

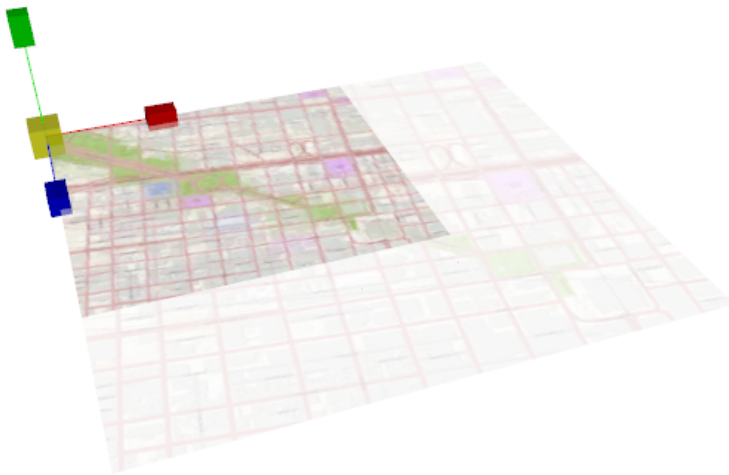
## Edit a map layer

After selecting a map layer in the [Scene Editor](#), its properties are visible in the [Inspector](#). You can edit the layer parameters such as map files, positioning bounds, and the elevation offset. In addition, an overlay color and alpha value for the map can be specified. Depending on the layer type, some options may not be available.

The mapping functions can also be edited in the [Inspector](#). See the [Edit map layer functions](#) section below.

### Move and scale a map layer in the Viewport

If you select one or more attribute layers in the [Viewport](#), you can use the transform or scale tool to move or scale the layer or layers.



*Scaling a map layer*

### Edit map layer attributes

Map layers can have their own attributes. These are defined in the **Layer Attributes** pane of the [Inspector](#), with the map layer selected.

### Edit map layer functions

Map layer function editing is similar to CGA shape grammar editing, but only a subset of functions are available for attribute layers and no rules or shape operations. Use **Ctrl+Spacebar** to see a list of available functions.

```
attr elevation = map_01(brightness, 1.17549435E-38, 27.620806)
+ elevationDelta + elevationOffset
```

There are two predefined attributes that will be used for street generation and other generative parts of CityEngine:

- **attr elevation** controls the elevation of the heightmap of a terrain layer.
- **attr obstacle** controls the obstacle avoidance of the street generation.

### Examples

<code>attr elevation = sin(u * 6.3) * cos(v * 6.3) * 100</code>	Create a terrain as a function of sine and cosine.
<code>attr obstacle = brightness &gt; 0.5</code>	Define all bright parts of an image map as obstacles.
<code>attr height = exp(u * 5)</code>	Control the height attribute of a rule file with this exponential function.
<code>attr selection = rand &gt; 0.5</code>	Define a Boolean attribute that can be used for a selection to select 50 percent of the objects randomly.
<div><code>attr landuse =   case u &gt; 0.5:     50%: "industrial"     else: "retail"   else:     "residential"</code></div>	Define a string attribute that can be used by a CGA shape grammar rule to control, for example, building appearance.

Layer attribute code in detail

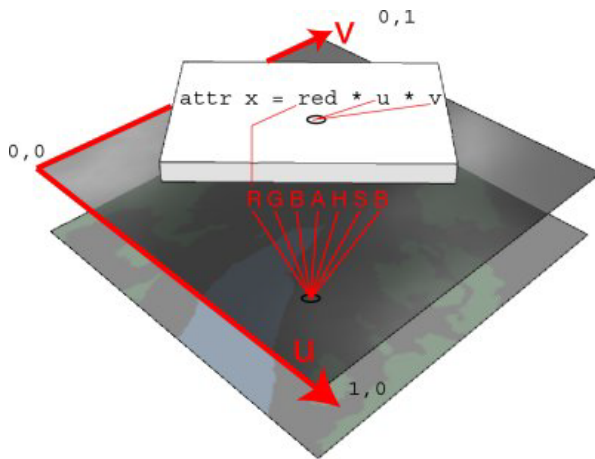
Map layers in general define one or more attributes as a function of the location and, optionally, a mapping channel. The dimensions of the map are normalized to the interval [0..1]. Thus the lower left corner of the map has the coordinates (0, 0) and the upper right corner of the map has the coordinates (1, 1). The normalized position is available as the predefined values "u" and "v", respectively, for attribute functions. For example, the following function will control the elevation by trigonometric functions:

```
attr PI2 = 3.141 * 2 // approx. 2 x PI
```

```
attr elevation = sin(u * PI2) * cos(v * PI2) * 100
```

In addition, inside an attribute function, "red", "green", "blue", "alpha", "hue", "saturation", and "brightness" address the individual channels of the map. For each object, the attribute function is evaluated with the projection of the center of gravity (centroid) to the x-z plane.

In the following illustration, the attribute "x" is evaluated at the center of gravity (centroid) of the object which is mapped onto the standard [0..1] range for the "u" and "v" parameters. In addition, the map is sampled at the position "u,v" and its red channel is used for the calculation of "x".

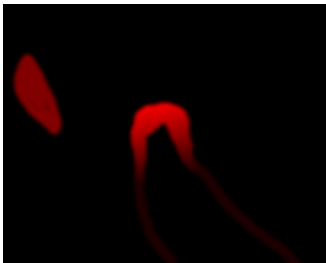


*attr x at center of centroid*

## Map image data to rule attributes

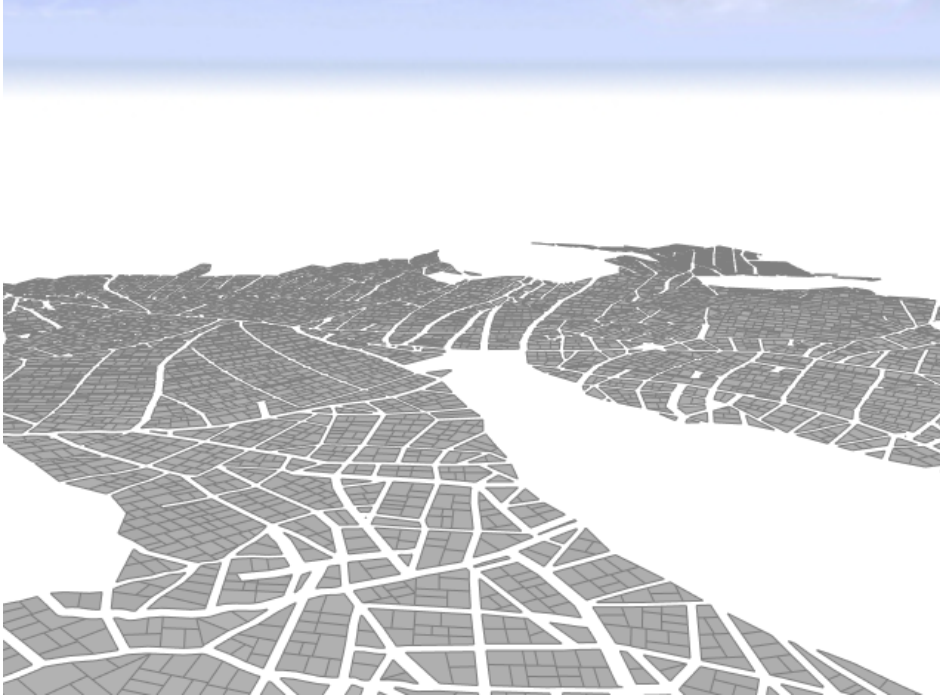
Map layers are a tool to control CGA shape grammar rules. Any attribute that you have defined in your CGA shape grammar rules can be mapped from an attribute layer. This allows you to guide your rules by maps. Typically, maps are used for controlling building attributes such as height or appearance, level of detail, and land-use mixes.

The image below is going to be used as a source image to control the height of a set of buildings.



*Source image used to control building heights.*

Assume you are starting from a scene with a set of building footprints such as in the image below:



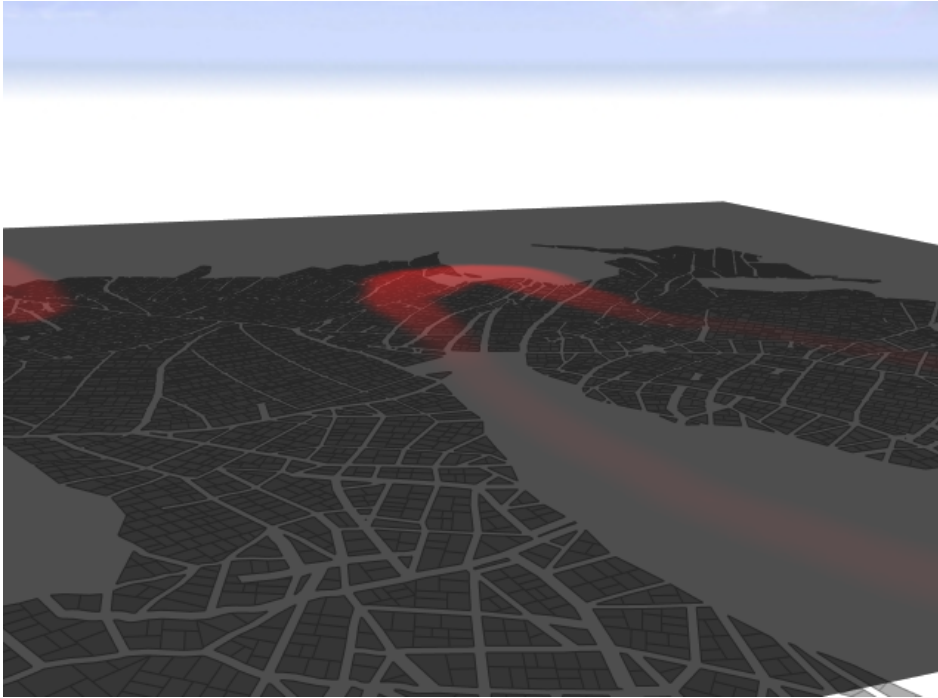
*Scene with building footprints*

All shapes have a rule file assigned that extrudes the footprints to a certain height.

```
attr height = 30  
Lot --> extrude(height)
```

## Create the mapping layer

Create a [mapping layer](#), choose your skyline map, set the bounds to fit your scene, and add a new attribute, `skylineValue`. Its range will define the range of the building heights.

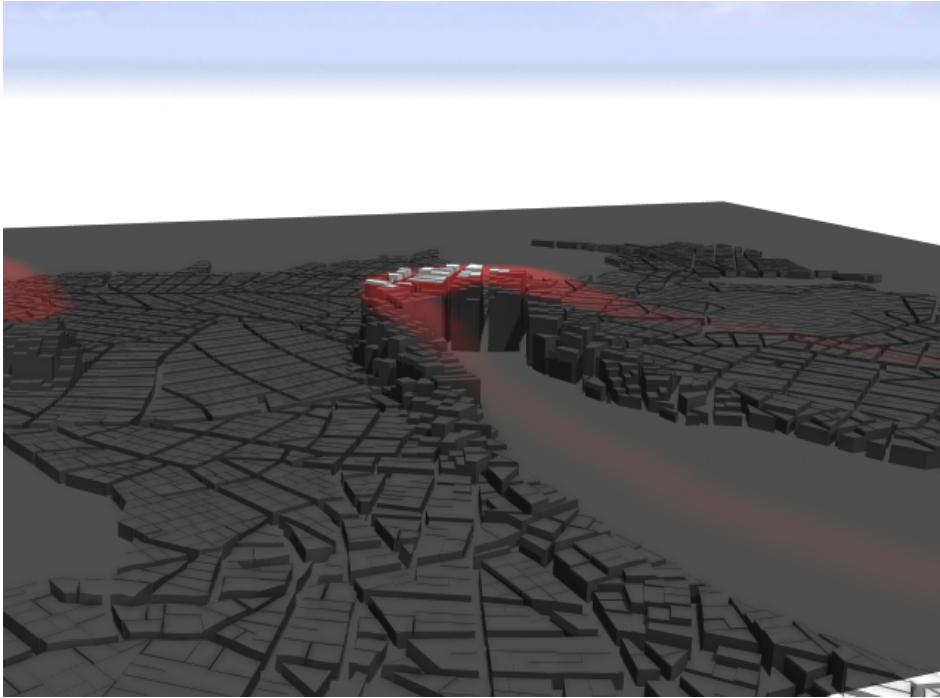


*Building heights before being defined by mapping layer range*

## Connection Editor

To connect attributes, do the following:

1. Select all footprint shapes, and choose **Connect Attribute** for the rule attribute height in the **Inspector** to open the **Connection Editor**.
2. Choose a layer attribute.
3. Select the new **Skyline map** layer.
4. Select the attribute **skylineValue**.



*Building heights after defined by mapping layer*

# Selection with image maps

If you need to work with selections, depending on a map layer or a map, define a Boolean attribute for the selection that you require.

## Use the terrain for a selection

Your terrain layer contains an elevation attribute of a similar form as below:

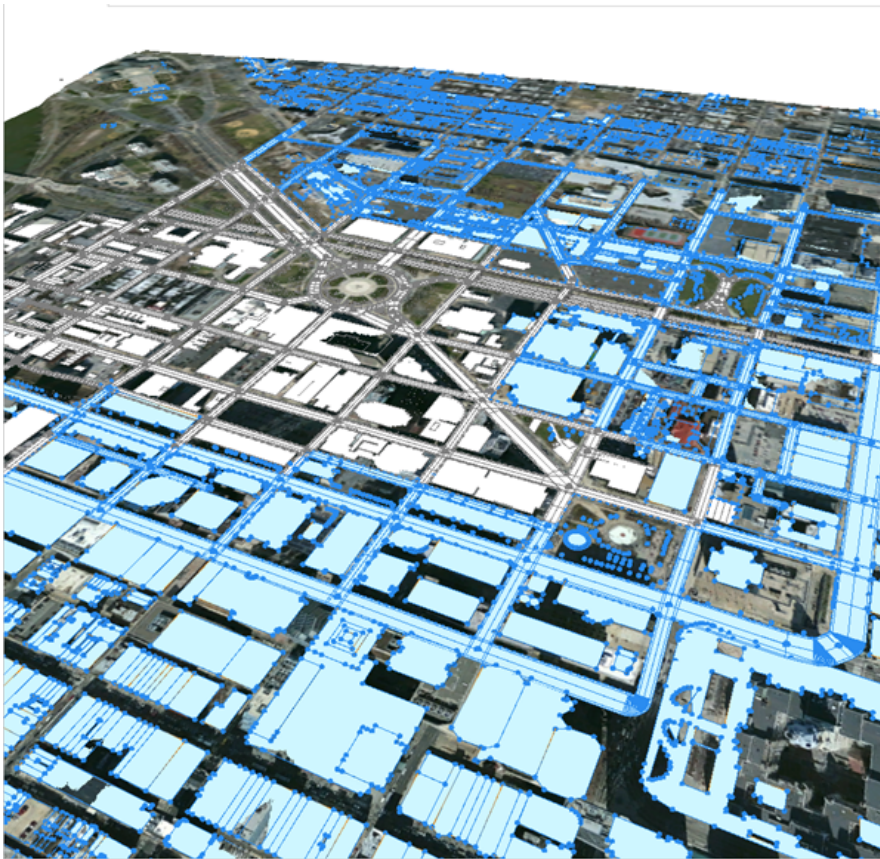
```
attr elevation = map_01(brightness, 100, -100)
```

You want to select all elements that have an elevation of 10 meters or higher. Add a new attribute that evaluates to true when elevation is higher than 10.

```
attr high = elevation > 10
```

Boolean attributes in map layers are automatically added to the selection menu. Click **Select > Select Objects in Map Layer > terrain: high** in the main menu.

The resulting selection is shown in the image below:



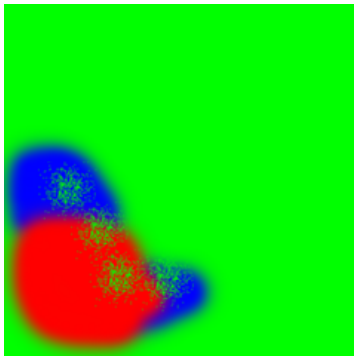
*Selected objects in Map layer*

**Note:**

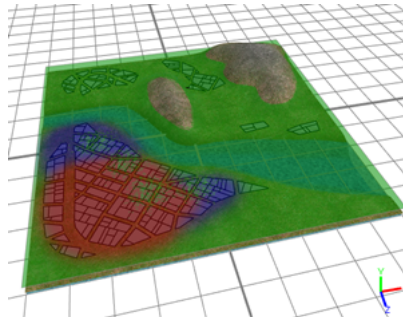
By selecting a layer before selecting objects by map layer, you can select only objects from the selected layer. If you have a layer selected that does not contain objects, this leads to nothing being selected.

## Use a land-use map for selection

Land-use types are often used to define certain areas of a scene. The map below defines commercial (red), urban residential (blue), and residential areas.



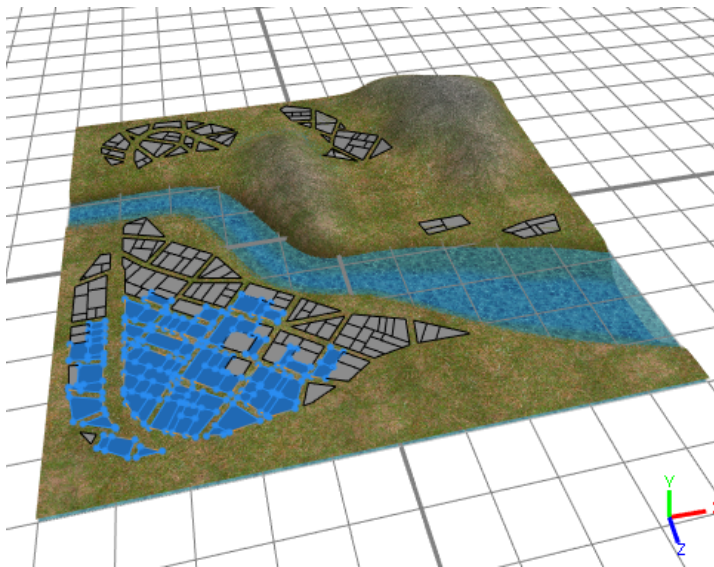
Land-use map in red, blue, and green



Land-use map placed in the scene.

After adding a new map layer with the land-use map, three new Boolean attributes are defined in the [Inspector](#) view of the new layer. Depending on the color of the map, land-use types are evaluated.

```
attr industrial = green > 0.8  
attr retail = red > 0.8  
attr residential = blue > 0.8
```



Land-use types defined by Map layer.

## Selection using the u,v coordinates of a map layer

For the following land-use attribute definition:

```
attr landuse =  
  case u > 0.5:  
    50%: "industrial"  
    else: "retail"  
  else: "residential"
```

Add the following attributes to the same map layer:

```
attr isIndustrial = landuse == "industrial"  
attr isRetail = landuse == "retail"  
attr isResidential = landuse == "residential"
```

This will give you additional choices in the **Select > Sect Objects by Map Layer** menu:

- Landuse: isIndustrial
- Landuse: isRedential
- Landuse: isRetail

# Drawing

# Drawing

CityEngine offers a variety of drawing tools to create and edit shapes and graphs.

Shapes	<a href="#">Shapes</a> can be polygons or 3D models that are made of multiple polygons. You can import them or draw them manually using shape drawing tools. Use the <a href="#">Push Pull tool</a> to extrude shapes and <a href="#">guides</a> to perform 3D shape editing with precision. In this section, you will also learn how to align shapes and terrains as well as <a href="#">manual shape texturing</a> .
Graphs	A <a href="#">graph</a> is a structure consisting of edges and nodes. In CityEngine, graphs are mainly used to lay out street networks. When drawing and editing edges and nodes, dynamic shapes for the drive and pedestrian lanes are created automatically.



# Draw shapes

# Polygons, rectangles, and circles

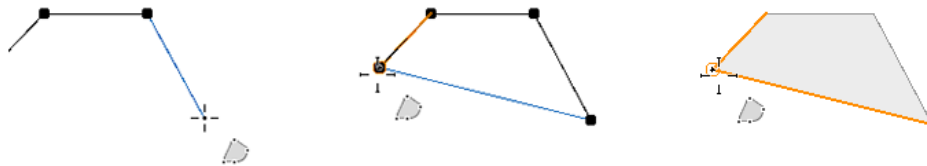
CityEngine provides tools to draw polygons, rectangles, and circles.

- See [Snapping](#), [Intersecting shapes](#), and [Drawing modes](#) for more shape creation options.
- See [Push Pull Tool](#) to create volumes out of 2D shapes.
- Use [Guides](#) to draw parallel and with an offset to existing features.

## Draw a polygon


To draw a polygon, click the **Polygonal Shape Creation** tool  (S) to open the **Tool Options** window . You can also click **Shapes** > **Polygonal Shape Creation** in the main menu.

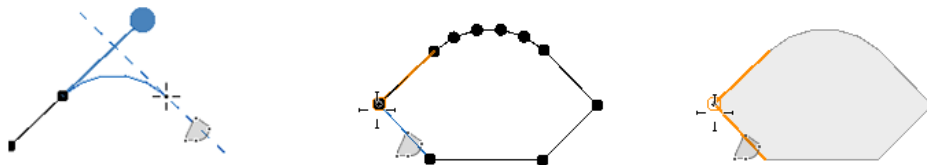
Click to create the first vertex and click again to place additional vertices. To finish the shape, click the first vertex. You can also double-click or press **Enter** to finish the shape. Double-click an edge to add a vertex at that place. See the [Polygonal Shape Creation tool options](#) section below for shape creation options.



*Polygonal Shape Creation tool*

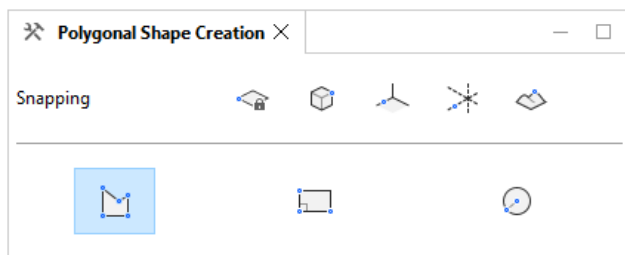
## Draw an arc

While you are drawing, you can switch between a line and an arc with the **Arc Mode** toggle key in the **Polygonal Shape Creation** tool options  or press (A). You can adjust the tangent of the arc with the handles.



*Drawing an arc*



## Polygonal Shape Creation tool options



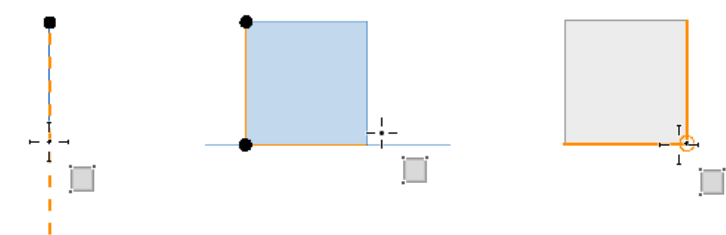
The **Polygonal Shape Creation** tool options  include the following:

✖ Polygonal Shape Creation	
Snapping	Turn snapping options on and off in the scene. See <a href="#">Snapping</a> .
Edge Length (m)	The length of the current edge in meters. You can enter a value before or while drawing to lock the edge length. Press <b>Enter</b> to apply the value.
Arc Mode	<div>Change the edge into an arc.</div> <ul style="list-style-type: none"><li>• Enter a value for the resolution of the arc. The smaller the value, the more sides the arc has.</li><li>• Change the value by using the scroll wheel or the up and down arrow keys.</li><li>• Press ( <b>A</b> ) to switch between an edge and arc.</li></ul>
Force New Shape	<a href="#">Force a new shape</a> to be created, including when drawing intersecting shapes.
Force Planar	Force the polygon to be planar. Press ( <b>T</b> ) to turn force planar mode on and off. See <a href="#">Force Planar</a> .
Automatic Closing	Create a shape that closes along the edges of connected shapes. See <a href="#">Automatic closing</a> .

## Draw a rectangle

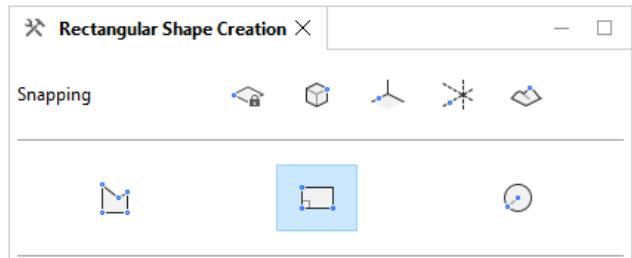
To draw a rectangle, click the **Polygonal Shape Creation** tool  ( **S** ) and click the **Rectangular Shape Creation** tool  ( **Shift+S** ) in the **Tool Options** window ✖. You can also click **Shapes > Rectangular Shape Creation** in the main menu.

Click to start drawing and click again to define the first edge. Drag the rectangle to the size you want and click. The rectangle is constrained to the line perpendicular to the first edge. Squares can be drawn by snapping the second perpendicular edge to the length of the first. See the [Rectangular Shape Creation tool options](#) section below for shape creation options.



Rectangular Shape Creation tool

## Rectangular Shape Creation tool options



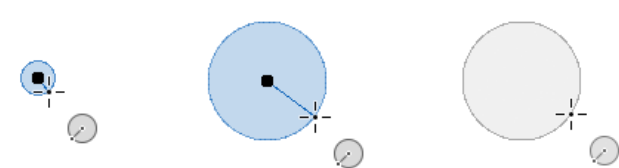
The **Rectangular Shape Creation** tool options ✂ include the following:

✂ Rectangular Shape Creation	
Snapping	Turn snapping options on and off in the scene. See <a href="#">Snapping</a> .
Edge Length (m)	The length of the current edge in meters. You can enter a value before or while drawing to lock the edge length. Press <b>Enter</b> to apply the value.
Force New Shape	<a href="#">Force a new shape</a> to be created, including when drawing intersecting shapes.
Force Planar	Force the rectangle to be planar. Press ( <b>T</b> ) to turn force planar mode on and off. See <a href="#">Force Planar</a> .

### Draw a circle

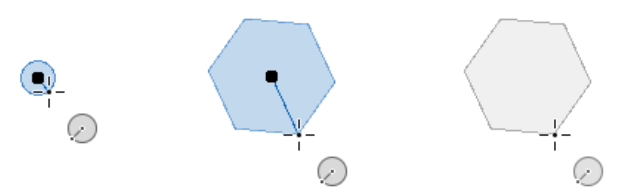
To draw a circle, click the **Polygonal Shape Creation** tool  (**S**) and click the **Circular Shape Creation** tool  (**Shift+C**) in the **Tool Options** window ✂. You can click **Shapes > Circular Shape Creation** in the main menu.

Click once to set the center of the circle, move the pointer to define the radius, and click again to finish. See the [Circular Shape Creation tool options](#) section below for shape creation options.



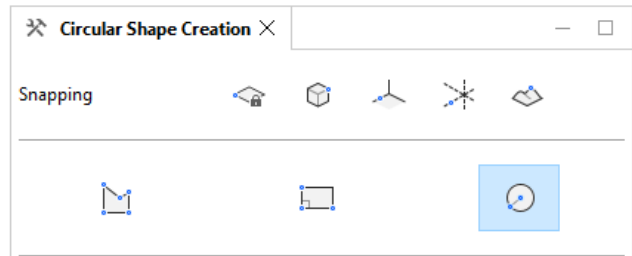
Circular Shape Creation tool

Similarly to the arc mode, you can change how many sides the circle has.



Circle changed to multiple sides.

### Circular Shape Creation tool options




The **Circular Shape Creation** tool ✂ includes the following options:


✂ Circular Shape Creation	
Snapping	Turn snapping options on and off in the scene. See <a href="#">Snapping</a> .

<b>Radius (m)</b>	The radius length in meters. You can enter a value before or while drawing to lock the radius length. Press <b>Enter</b> to apply the value.
<b>Segments</b>	The number of segments the circle has. You can enter a value or use the scroll wheel or the up and down arrow keys to change the number of segments.
<b>Force New Shape</b>	<a href="#">Force a new shape</a> to be created, including when drawing intersecting shapes.
<b>Force Planar</b>	Force the circle to be planar. Press ( <b>T</b> ) to turn force planar mode on and off. See <a href="#">Force Planar</a> .

# Push Pull tool


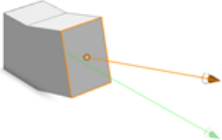
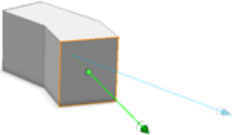
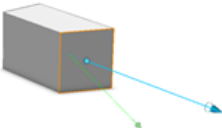
Extrude 2D shapes and modify existing 3D extrusions with the **Push Pull** tool . You can hover over any shape to display handles to extrude an edge or face.

To extrude a shape, do one of the following:


- Click the **Push Pull** tool .
- Press **P**.
- Click **Shapes > Push Pull Tool** in the CityEngine main menu to open the tool.

## Extrude face

Depending on the shape, different arrows appear. This allows you to create shapes along different directions. Hover over a direction arrow for immediate feedback. There are four types of directions, each with a unique color and mouse icon.


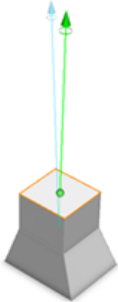
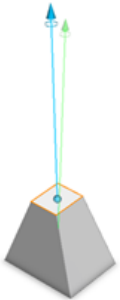
Global axes	
Face normal	
Face normal projected on the ground plane	
Special edge directions from adjacent faces	

 **Tip:**

- Use [guides](#) to snap to other shapes.
- Turn on [Force New Edges](#) in the **Push Pull Tool** options  or press **Ctrl** to force the creation of new edges when extruding.

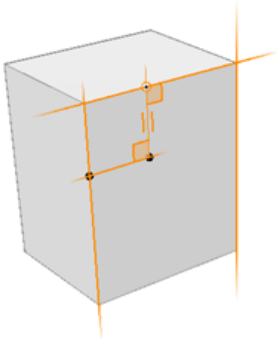
### Special edge direction

When dragging along the special edge direction arrow, all edges are extended along their adjacent faces during dragging. The arrow appears with a slight offset when it has the same direction as another arrow. The table below shows the difference between the up and special edge direction dragging.

Starting shape.	
Shape is dragged directly up along the green arrow.	
Shape is dragged along blue arrow extending adjacent faces.	

### 3D shape editing


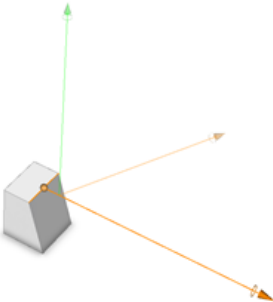
3D shapes allow polygonal editing, such as [snapping](#), [splitting](#), and automatic closing. Split parts can be moved to further refine the 3D model.

Preliminary split	
-------------------	---

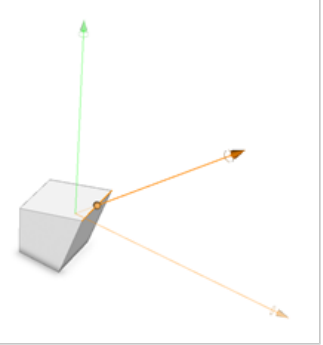
3D move	
Final split and 3D move	

### 3D edge move

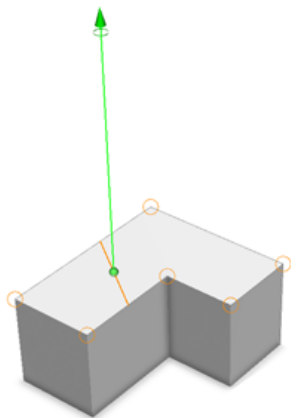
Hover over edges with the **Push Pull** tool to display edge handles. Similar to dragging faces, you can drag edges in multiple directions.

Edge move along global axis	
Edge move along adjacent faces	

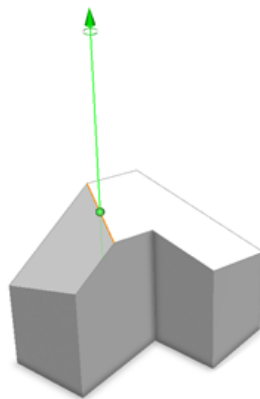
Edge move along average face normal



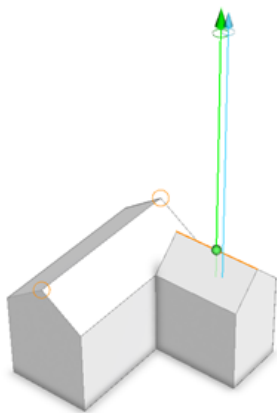
While moving the edges with the **Push Pull** tool, connected faces update to maintain planarity. In addition, the moved edge is intersected with neighboring polygons. Both features are useful for creating roofs as shown below.



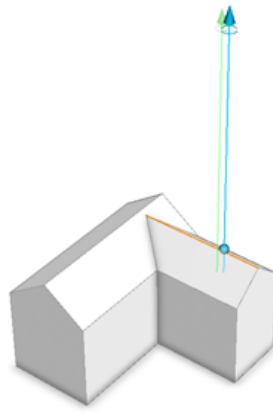
*The edge before moving the orthogonal faces is shown.*



*The orthogonal faces moving with the edge are shown.*



*An inset edge along a roof is shown.*



*An edge move along the blue axis insets the edge along the existing roof.*

## Tool options


The **Push Pull** tool options  include the following:

<b>Distance (m)</b>	<p>Distance in meters of the edge or face extrusion.</p> <ul style="list-style-type: none"><li>• You can enter a value to lock the extrusion distance along the available directions.</li><li>• Click again to start a new extrusion.</li></ul>
<b>Snapping</b>	<p>Turn snapping options on and off.</p> <ul style="list-style-type: none"><li>• Extrude a face and snap to faces, edges, or vertices of the same shape. Also snap to guides.</li><li>• Extrude an edge and snap to vertices of the same shape.</li><li>• Press <b>Shift</b> to temporarily enable or disable snapping.</li></ul>
<b>Force New Edges</b>	<ul style="list-style-type: none"><li>• Force the creation of a new edge when extruding.</li><li>• Press <b>Ctrl</b> to temporarily turn on or off the creation of new edges.</li></ul>

# Guides

You can create lines as guides in 3D that help you position and align with other existing objects as you draw.

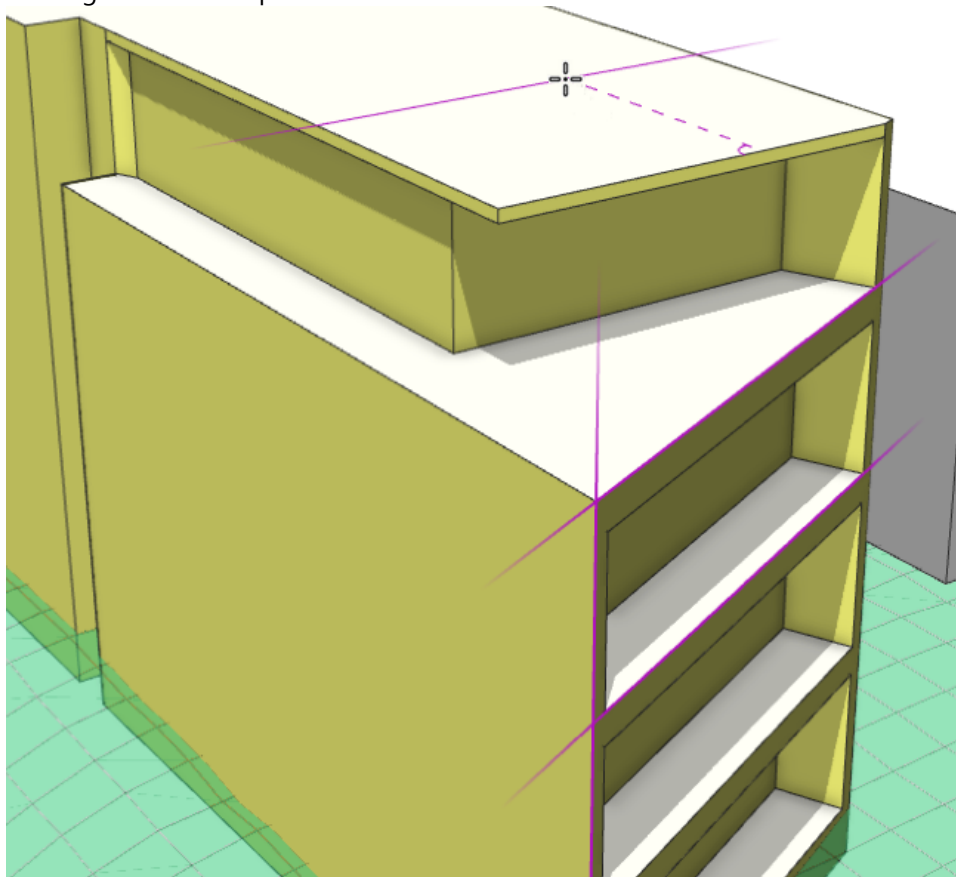
To use the guide tool, do one of the following:

- Click the **Guide Creation** tool  on the toolbar.
- Click **Shapes** > **Guide Creation** in the main menu.

## Create guides

To place guides, do the following:

1. Click any shape, model, or street edge to create a guide from the edge.
2. Move the guide to the location you want or snap to other objects.  
Alternatively, you can manually specify an [offset](#).
3. Click again to set the placement.



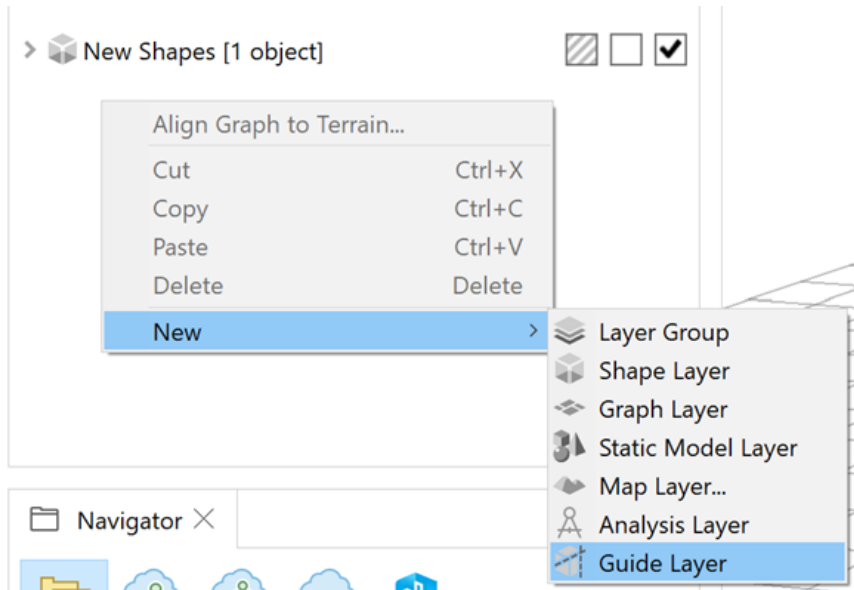
The guide is an additional snap target that allows parallel as well as extension snapping.

## Guide layers

When you create a guide, a scene object is added under a guide layer in the **Scene Editor** window. If there is not an existing guide layer, a new guide layer is automatically created. Otherwise, the guide is assigned to the existing guide layer or the selected one.

You can also create a guide layer in the following ways:

- Click **Layer > New Guide Layer** in the main menu.
- Right-click in the **Scene Editor** window and click **New > Guide Layer**.

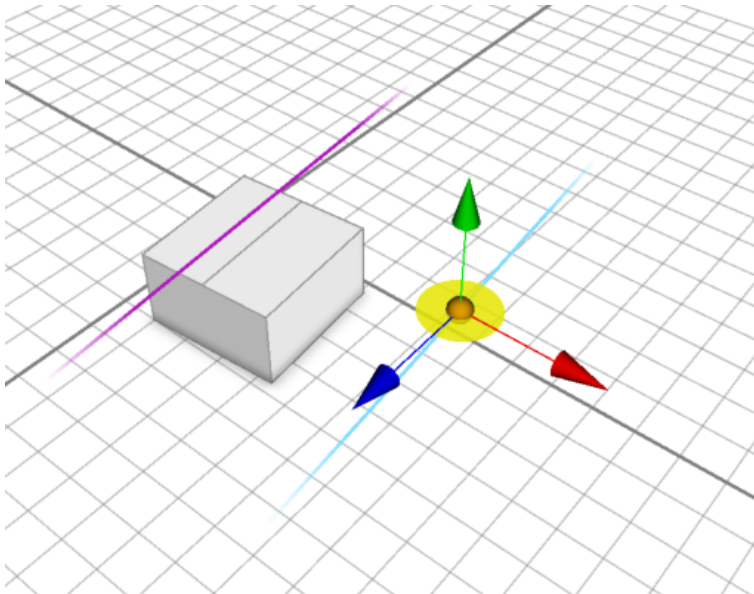


All the default layer functionalities such as visibility, color adjustments, or lock settings, apply to the guide objects .


Guide layers and guide objects are saved in the scene.

## Modify guides

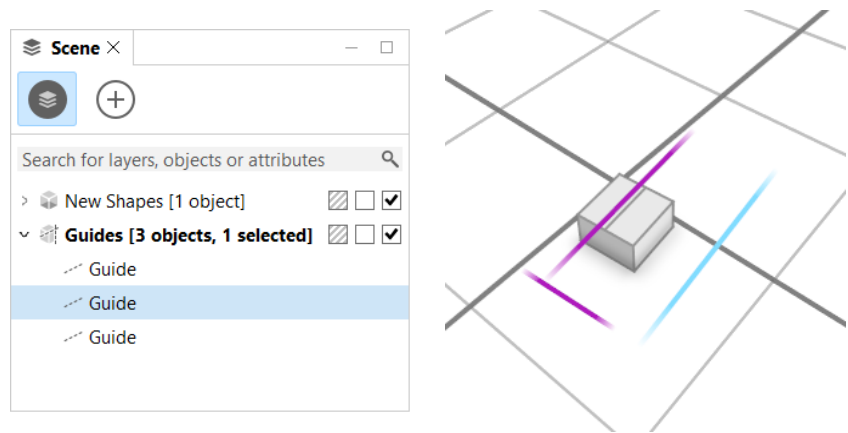
Guides can be moved, scaled, and rotated using the transform tools.



To delete guides, do the following:


1. Select the guide using one of the following ways:
  - Use the **Select** tool  to select the guide in the **Viewport** window.


- Click the guide object in the **Scene Editor** window.



2. Press the **Delete** key.
- Optionally, you can also delete the **Guide Layer** in the **Scene Editor** window to remove all the guides belonging to that guide layer.



## Tool options

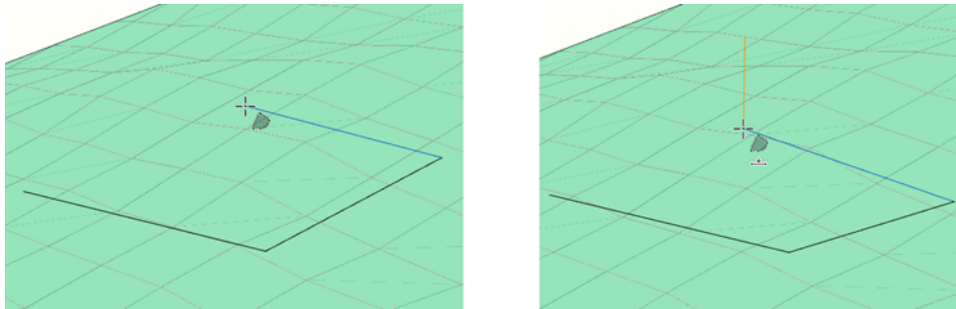
The **Guide Creation** tool options  include the following:

 <b>Guide Creation Tool</b>	
<b>Offset (m)</b>	<p>The offset distance in meters from the edge that is clicked.</p> <ul style="list-style-type: none"><li>You can enter a value to lock the offset.</li><li>Preview the offset guide placement until the next click sets the guide.</li></ul>
<b>Force Planar</b>	<p>Force the guide to be planar. Press <b>T</b> to turn force planar mode on and off. See <a href="#">Force Planar</a>.</p>

# Drawing modes

When using the [shape drawing](#) tools or the [guides](#), you have the choice to turn planar mode on or off in your scene.

To turn planar mode on and off, click the **Force Planar** toggle button in the **Tool Options** window  for the selected drawing tool. See [Work with polygons, rectangles, and circles](#). You can also press ( **T** ). The planar icon  displays when the option is on in your scene.



*Left: The Polygonal Shape Creation tool with planar off. All vertices are placed onto the terrain. Right: Planar is on. All vertices are placed in the same horizontal plane, defined by the first vertex. The orange plumb line shows the current pointer position projected onto the terrain.*

## Force Planar

When drawing shapes with the planar option off, vertices and edges are placed directly onto the terrain, such as with building footprints. For nonplanar terrains, this usually results in footprints that aren't horizontal and are often nonplanar. However, in many cases, planar footprints are required.

When you turn planar mode on, the first vertex or edge you place defines an imaginary horizontal drawing plane, and additional vertices and edges are placed on this drawing plane. Therefore, the resulting shapes are always planar.

## Guides

With the planar option on, when you draw a shape, the guides are projected onto the drawing plane and serve as snap targets. When the shape is finished or you turn planar mode off, the guides reappear at the original position.

When you turn planar mode on and [create guides](#), you place guides on the same drawing plane as the edge that is clicked. Otherwise, you can create guides on different drawing planes.

## Plumb line






Drawing on terrain can be challenging, as the relation of pointer position, terrain, and drawing plane isn't always clear. To illustrate this relation, an orange vertical line is displayed that connects the pointer position on the drawing plane with the point on the terrain that is vertically above or below. This is especially helpful when tracing features from satellite imagery while in the 3D view.

# Snapping

When using the [shape drawing tools](#), you can snap to existing elements in your scene, such as shapes, streets, or terrain. When snapping is available, the pointer changes and orange highlights on edges and vertices are displayed.

## Snapping options

You can turn snapping on and off in the drawing tools by clicking the following types of snapping elements:


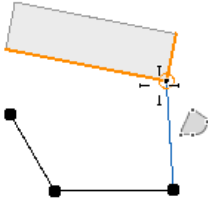
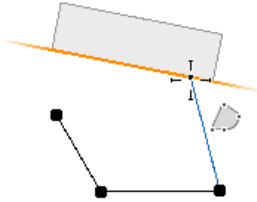
 <b>Locked objects</b>	Snap to locked objects in the Scene Editor.
 <b>Shapes</b>	Snap to shapes such as footprints, 3D models, and streets.
 <b>Global Axes</b>	Snap to global axes.
 <b>Guides</b>	Snap to guides.
 <b>Terrain</b>	Snap to the terrain. See also <a href="#">Force Planar</a> .

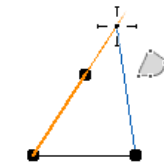
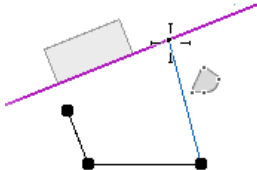
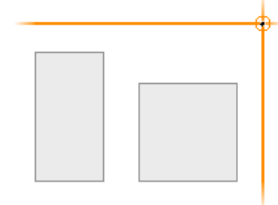

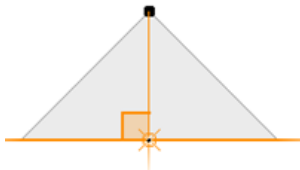
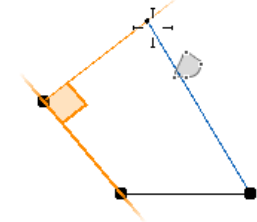
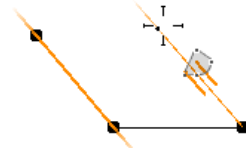
 **Note:**

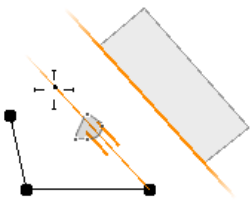
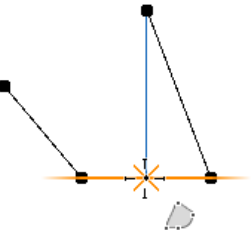
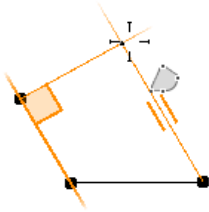
Press **Shift** while drawing to temporarily disable the selected snapping elements.

## Snapping features

The shape drawing tools allow you to snap to the following features:

<b>Global axes</b> Snap to the global axes.	
<b>Vertex</b> Snap to shape or street vertices.	
<b>Edge</b> Snap to shape or street edges.	

<div>Edge extension</div> <div>Snap to the extension line of a previous edge in the shape.</div>	
<div>Guide</div> <div>Snap to guides and guide extension lines.</div>	
<div>Intersection</div> <div>Snap to intersecting edges, guides, and guide extensions.</div>	
<div>90 degrees</div> <div>The current segment is 90 degrees to the previous segment.</div>	
<div>90 degrees to edge or guide</div> <div>The current segment is 90 degrees to edges, guides, and guide extensions.</div>	
<div>90 degrees extension line from starting vertex</div> <div>The extension line between the starting vertex and next vertex is 90 degrees to the starting segment.</div>	
<div>Parallel edge</div> <div>The current segment is parallel to edges in the shape.</div>	

<p><b>Parallel guides</b></p> <p>The current segment is parallel to guides.</p>	
<p><b>Midpoint</b></p> <p>Snap to the midpoint of an edge.</p>	
<p><b>Combination</b></p> <p>Snap to the intersection of combined snapping features.</p>	




**Tip:**

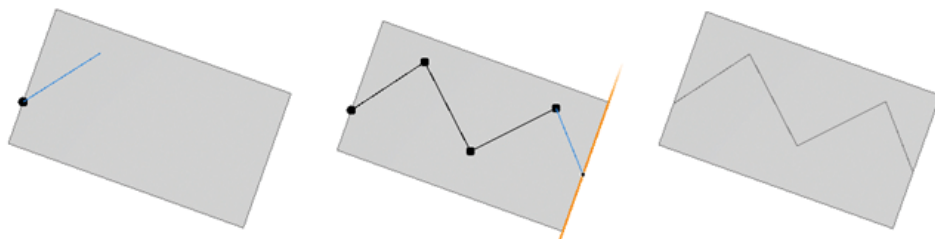
- To finish the shape when drawing, double-click or press **Enter**.
- Snapping is also available with the [Transform Move](#) tool.

# Intersecting shapes

When you draw shapes that overlap other shapes, the shape creation features such as splitting, combining, or automatic closing are available depending on the configuration of the tool options.



## Splitting

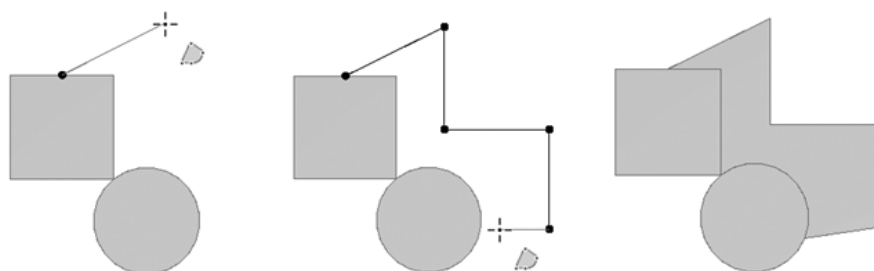
You can split shapes using the **Polygonal Shape Creation** tool  (S). When the first and last vertex are snapped to an edge or vertex of the same shape, and the whole line is contained in this face, it is automatically split into two parts.\*



*Splitting a polygonal shape is shown.*

## Automatic closing

To create a shape that closes along the edges of connected shapes, click the **Automatic Closing** toggle key in the **Polygonal Shape Creation** tool options . Ensure that the **Shapes** option  is enabled under **Snapping**. Snap the first point to an edge or vertex of a shape and continue to the last point in which you snap to an edge or vertex that is indirectly or directly connected.\*




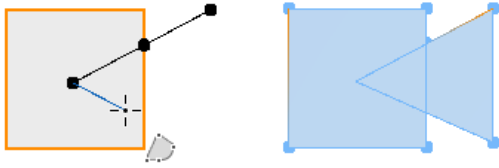
*The polygon is automatically closed along the indirect connection.*

### Note:

To automatically close shapes, ensure that the toggle key is on before placing the first point.

## Combine shapes

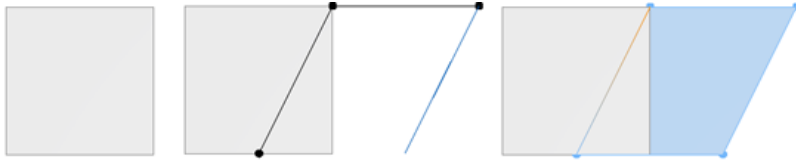
To combine shapes into one shape, click any of the shape creation tools and draw a shape that snaps to an existing shape. Ensure that the **Shapes** option  is enabled under **Snapping** in the shape creation tool options.\*



Combining shapes is shown.

## Force New Shape

The **Force New Shape** option forces a new shape to be created, including when drawing intersecting shapes. Click the **Force New Shape** toggle key in the [tool options](#) of any of the shape creation tools.




Drawing a new shape is shown.



### Note:

\*This option is unavailable when **Force New Shape** is enabled.

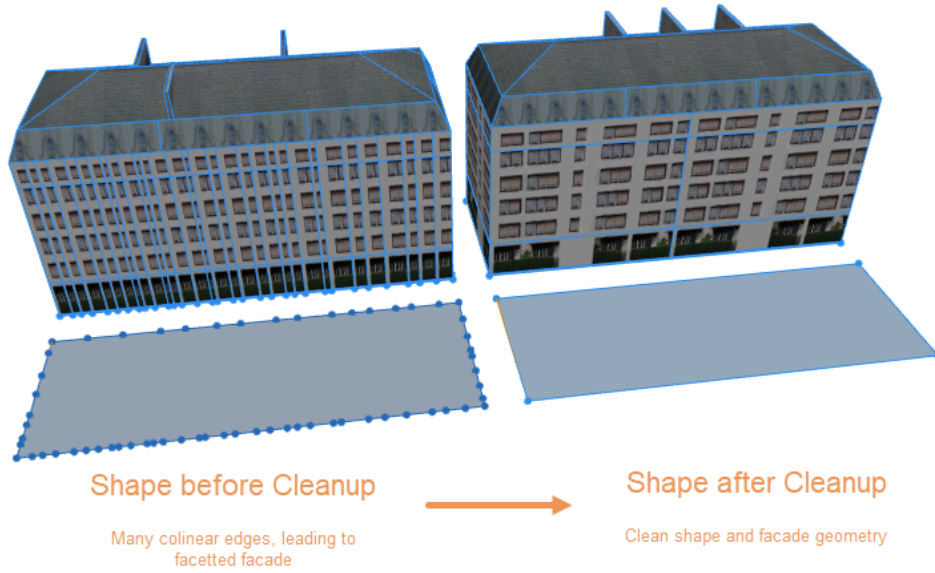
# Cleanup shapes

The **Cleanup shapes** tool  simplifies and cleans the geometry for polygons (faces), vertices, and edges. It is useful to prepare imported meshes for 3D editing. The 3D editing works best after cleanup is performed, as this ensures correct connectivity information in the mesh. If you ever encounter problems with 3D editing, try a cleanup operation with the default values.

## Cleanup shapes options



The individual operations are performed in the direction indicated by the dialog box, and work as follows:

<b>Merge Vertices</b>	If the distance between two vertices is lower than the threshold, they are combined into one.
<b>Remove Coplanar Edges</b>	Merge connected coplanar polygons.
<b>No Cleanup on Discontinuous Textures</b>	When the texture coordinates are not continuous for a vertex, all operations are skipped for this vertex.
<b>Remove Collinear Vertices</b>	Multiple vertices on one straight line are removed.
<b>Remove Double Faces</b>	Faces that have identical vertices (up to shift and inversion) are removed but one is kept.
<b>Remove Zero Faces</b>	Faces with zero size are removed.
<b>Intersect Edges</b>	All edges are intersected, and new vertices are inserted for every intersection point.
<b>Split Coplanar Polygons</b>	Overlapping polygons on the same plane are split into nonoverlapping polygons. This requires the Intersect Edges operation.
<b>Conform normals</b>	Computes consistent normals using the connectivity and a heuristic that favors the world's up direction. It may be necessary to merge vertices and remove double faces to be successful.
<b>Distance Tolerance, Angle Tolerance</b>	Thresholds for the above operations.

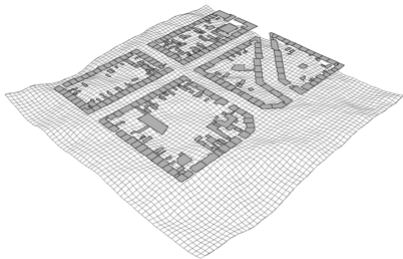


*An example of the Cleanup shapes tool in action, removing collinear vertices, is shown.*

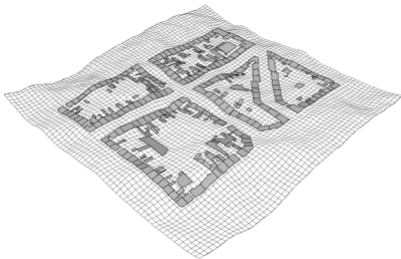
# Align shapes to terrain

The **Align shapes to terrain** tool  aligns shapes to arbitrary terrains (map layers with attribute elevation defined) or to the y=0 level. All currently selected shapes and all shapes of the selected layers are aligned. The shapes are aligned to a terrain, using an alignment function and an optional offset. You can access the **Align shapes to terrain** tool  the following ways:

- Click the **Align shapes to terrain** tool  on the toolbar.
- Click **Shapes > Align shapes to terrain** in the main menu.



Shapes not aligned to terrain.




Shapes aligned to a terrain.


## Align shapes to terrain options

The following parameters control the alignment:

<b>Align function</b>	<p>The alignment function to apply to the vertices of the shape polygons.</p> <ul style="list-style-type: none"><li>• <b>Project All</b>—Projects all shape vertices onto the terrain.</li><li>• <b>Project Below</b>—Projects the vertices located below the terrain only.</li><li>• <b>Project to Object Average</b>—Projects the shape vertices to the average of the vertices.</li><li>• <b>Translate to Average</b>—Translates the shape to the average elevation of the projected vertices.</li><li>• <b>Translate to Maximum</b>—Translates the shape to the maximum elevation of the projected vertices.</li><li>• <b>Translate to Minimum</b>—Translates the shape to the minimal elevation of the projected shape vertices on the terrain.</li></ul>
<b>Terrain</b>	<p>The terrain to align the shapes. All attribute layers with an elevation attribute plus the y=0 level are listed here.</p>
<b>Offset</b>	<p>The offset to add after alignment to the y-coordinate of the shape points.</p>

# Texture shapes


The **Texture shapes** tool  is used for the manual texturing of shapes and individual faces. For exact texture mapping, several modes are available. You can open the tool in the following ways:

- Click the **Texture shapes** tool .
- Click **Shapes > Texture Shapes** in the main menu.
- Drag an image file from the **Navigator** window onto shapes.

 **Note:**

The tool dialog boxes are usually modal, meaning they stay in front of the CityEngine main window to allow repeated application to different shape selections.

## Choose a texture image

The **Texture shapes** tool  provides an interactive means for texture assignment. Choose an image to use as a texture:

- Click the **Browse/pick** button and browse to the file within the project. The upcoming dialog box also shows (at the bottom) a list of images that are already used in the current selection.
- Click or drag an image file from the **Navigator** window.

## Texture shapes options

The following is a list of the **Texture shapes** tool  parameters:

<b>Orientation Alignment</b>	Select an orientation for the following parameters, either <b>global</b> or <b>local</b> .
<b>Rotation</b>	Correct the image orientation using the following options in the <b>Image Transformation</b> group: <ul style="list-style-type: none"><li>• <b>Rotation</b>—None   90° counter clock wise   180°   90° clock wise   Arbitrary</li><li>• <b>Flip horizontally</b></li><li>• <b>Flip vertically</b></li></ul>
<b>Arbitrary Rotation</b>	Specify an arbitrary rotation angle in degrees, if <b>Rotation</b> above is set to <b>Arbitrary</b> .

<b>Texture Coordinates Mapping Mode</b>	<p>There are four modes with according attributes for the texture coordinate mapping, which are specified in the <b>Texture Coordinates Mapping</b> group:</p> <ul style="list-style-type: none"><li>• <b>Keep current mapping</b>—This mode leaves the currently set mapping (no attributes).</li><li>• <b>Stretch to polygon</b>—This mode is used for stretching the texture across the selected face. Note that if entire shapes are selected, the texture is stretched across all present coplanar face groups.<ul style="list-style-type: none"><li>▪ <b>Align to</b>—Bottom left corner   Bottom right corner   Top left corner   Top right corner (only has an effect if one of the repetition numbers is not a whole number).</li><li>▪ <b>Horizontal repetitions</b>—This is the number of times the texture is repeated horizontally (decimal number allowed).</li><li>▪ <b>Vertical repetitions</b>—This is the number of times the texture is repeated horizontally (decimal number allowed).</li></ul></li><li>• <b>Dimension</b>—This mode allows you to set the actual dimension of the used texture.<ul style="list-style-type: none"><li>▪ <b>Align to</b>—Bottom left corner   Bottom right corner   Top left corner   Top right corner</li><li>▪ <b>Absolute texture width</b>—This is given in meters.</li><li>▪ <b>Snap horizontally to bounds</b>— This stretches the given texture's width such that it horizontally fits the selected face with a whole number of repetitions.</li><li>▪ <b>Absolute texture height</b>—This is given in meters.</li><li>▪ <b>Snap vertically to bounds</b>—This stretches the given texture's height such that it vertically fits the selected face with a whole number of repetitions.</li></ul></li></ul>
---	---

Scene selection issues

The **Texture shapes** tool  acts differently depending on the scene selection (colored blue):

- Single or multiple face selection—The texture is individually applied to each selected face.
- Single or multiple shape selection—The texture is individually applied to each group of coplanar faces within the selection.

## Examples



*Stretch to Polygon is shown using 1.8 horizontal repetitions, 1.3 vertical repetitions with Align to Top left corner.*



*Dimension is shown using Absolute texture width 5.75, Absolute texture height 8.0, Align to Bottom right corner and snap horizontally to bounds, so the texture fits exactly three times.*



# Draw graphs

# Graph networks

CityEngine includes a suite of intuitive tools that allow you to create and edit graph networks. You can do the following with these tools:

- Create graph networks with street creation tools.
- Use the [selection tool](#) to select street nodes and edges. The selection displays in the Inspector, in which you can edit the attributes.
- Modify graph segments and nodes using the [transform tools](#).
- See [Segment and sidewalk parameters](#) for more information.




## Draw a polygonal street

To draw a polygonal street, click the **Polygonal Street Creation** tool  (G) to open the **Tool Options** window . You can also click **Graph > Polygonal Street Creation** in the main menu.




Click to create the first vertex and click again to place additional vertices. To finish the street, double-click or press **Enter**. The streets automatically snap to shape vertices and edges. Additionally, you can snap to [guides](#) and align segments parallel and perpendicular to them. Press **Shift** to temporarily disable snapping. Double-click a street segment to add a vertex at that place.

## Polygonal Street Creation tool options

The **Polygonal Street Creation** tool options  include the following:


 <b>Polygonal Street Creation</b>	
<b>Segment Length (m)</b>	<p>The length of the current street segment in meters.</p> <ul style="list-style-type: none"><li>• Enter a value before or while drawing to lock the length. Press <b>Enter</b> to apply the value.</li><li>• Lock and unlock the length by clicking either the lock  or the unlock button .</li></ul>
<b>Re-use settings from neighbors</b>	<p>If enabled, settings are copied from neighbor streets, if available. If an existing street is extended, settings are typically copied from that segment.</p>
<b>Street width</b>	<p>The street width.</p>
<b>Left sidewalk width</b>	<p>The left sidewalk width.</p>
<b>Right sidewalk width</b>	<p>The right sidewalk width.</p>
<b>Rule file</b>	<p>The rule file is assigned to all new street shapes.</p>
<b>Apply rule-based model generation</b>	<p>If enabled, model generation is automatically triggered.</p>
<b>Align terrain</b>	<p>Aligns the street with the terrain.</p>
<b>Intersect Segments</b>	<p>Create a new node at the intersection of segments.</p>


## Draw a street freehand

To draw a street freehand, click the **Polygonal Street Creation** tool  ( S ) and click the **Freehand Street Creation** tool  ( Shift+G ) in the **Tool Options** window . You can also click **Graph > Freehand Street Creation** in the main menu.

Click and drag to start drawing a street and release when finished. The streets automatically snap to shape vertices and edges. Additionally, you can snap to [guides](#) and align segments parallel and perpendicular to them. Press **Shift** to temporarily disable snapping. Double-click a street segment to add a vertex at that place.

## Freehand Street Creation tool options


The **Freehand Street Creation** tool options  include the following:


 <b>Freehand Street Creation</b>	
<b>Re-use settings from neighbors</b>	If enabled, settings are copied from neighbor streets, if available. If an existing street is extended, settings are typically copied from that segment.
<b>Street width</b>	The street width.
<b>Left sidewalk width</b>	The left sidewalk width.
<b>Right sidewalk width</b>	The right sidewalk width.
<b>Rule file</b>	If set, this rule file is assigned to all new street shapes.
<b>Apply rule-based model generation</b>	If enabled, model generation is automatically triggered.
<b>Align terrain</b>	Aligns the street with the terrain.
<b>Intersect Segments</b>	Create a new node at the intersection of segments.

## Additional graph tools

CityEngine also includes additional tools to help with creating and editing streets, such as the [Edit streets and curves](#), [Cleanup streets](#), and [Align streets to terrain](#) tools.

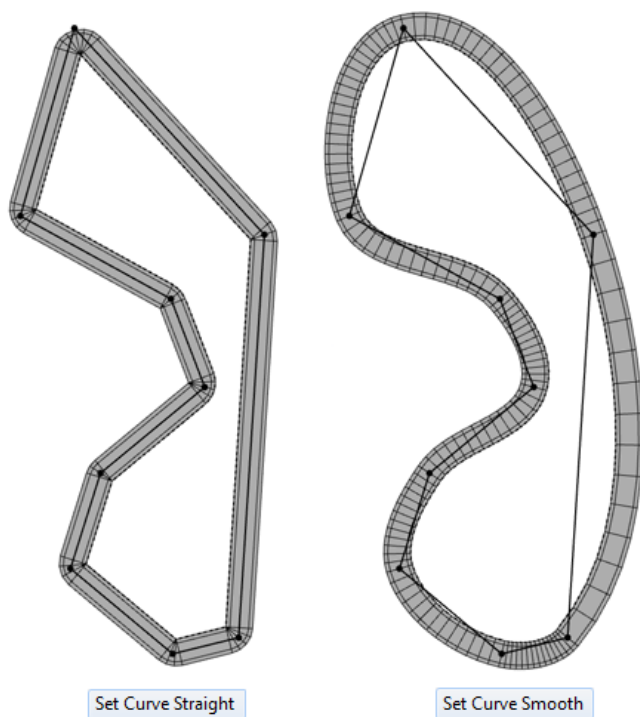
## Edit street and curves

The **Edit street/curves** tool  provides handles to graphically edit street width, sidewalk width, and overall curvature. To edit streets, do either of the following:

- Click the **Edit street/curves** tool .
- Press **C**.
- Click **Graph** > **Edit Streets/Curves** from the main menu.

### Straight versus smooth

Streets can be created either straight (default) or smoothly curved. To quickly switch between the two states, click **Graph** > **Set Curve Straight** and click **Graph** > **Set Curve Smooth** in the **Graph** main menu.




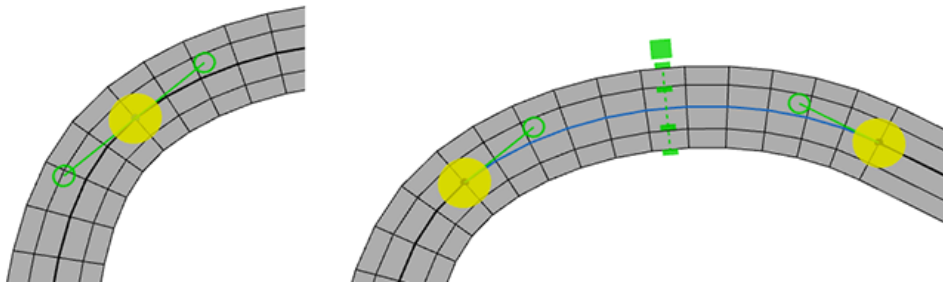
*Curve Straight and Curve Smooth tools are shown.*

To automatically choose between those types, you can use **Graph** > **Curves Auto Smooth**. Here you have two parameters:

- **Threshold angle** determines the minimum angle for curve smoothing.
- **Horizontal optimize** set streets in front of slopes to straight to prevent oscillations.

### Street editing

Once you activate the **Edit street/curves** tool , handles are displayed for selected streets or nodes. There are two types of handles: curve handles and street width handles. When a single node is selected, only the curve handle is shown. When a single street is selected, a combined curve and street width handle is displayed. When selecting multiple streets, only the street width handle is shown.



The left image shows the curve handle. The right image shows the curve and street width handle.



### Tip:

To lock the direction, press the **Shift** key before dragging a curve handle.

## Curve handles

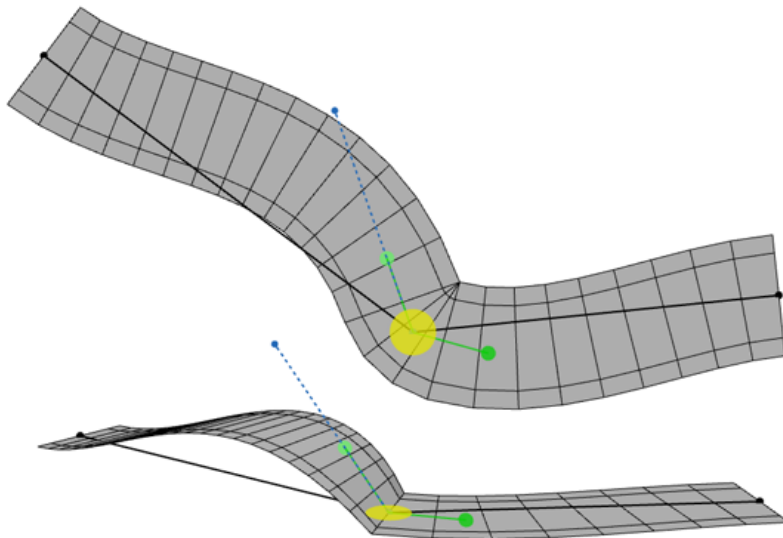
Each graph segment has two curve handles, one attached at the start and end nodes of the segment. The green handles drive the start and end direction of the street. The yellow circle allows moving the node. The dashed blue line (which is only visible when you hover over one of the curve handles) indicates the local weight of the curve.



### Note:

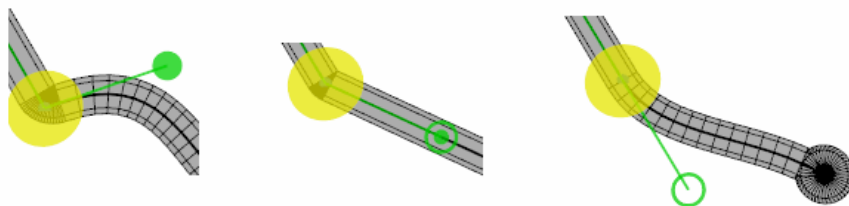
The spline is mathematically defined by the two end points of the dashed blue lines, plus the segment's start and end points.

To edit either the horizontal or vertical components of a curve handle, change the viewing angle relative to the segment node. A steep angle allows you to edit the horizontal component, with a glancing viewing angle of the vertical component.



Curve handles are shown.

The green circle of the curve handle indicates the smoothing type of the node (manual direction, straight, or smooth). Clicking a green circle while pressing the **Ctrl** key switches between the types.



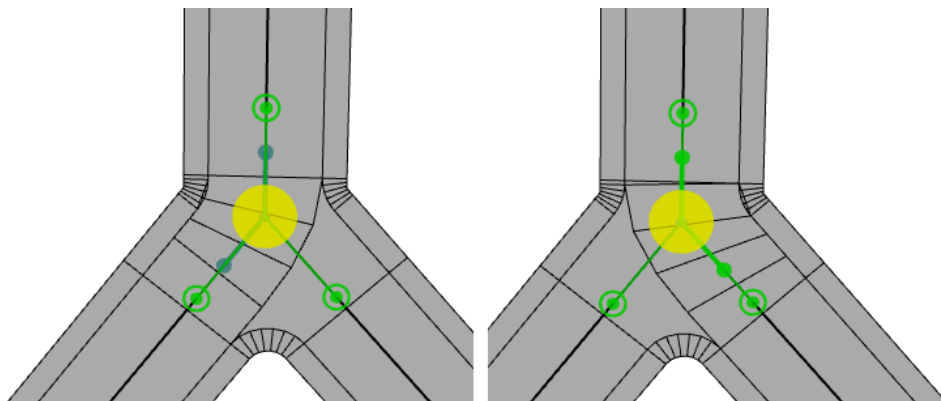
The images show the indication of node type in a green circle: (left) smooth with manual directions (middle) straight (right) smooth with automatic directions.

#### **Note:**

The segmentation of the street shapes defines the segmentation of the neighboring lot shapes. The overall number of segments per curve can help you avoid issues such as narrowly subdivided facades.

#### *Principle street handles*

When an intersection has the type Freeway or Junction and has three or more connecting streets, the curve handles are accompanied by principle street handles. These allow the principle street at an intersection to be edited, changing the intersection geometry.



Dark green principle street handles (above, left) indicate that the default principle street will be chosen. Once a principle street is specified manually, the handles are colored light green (right).

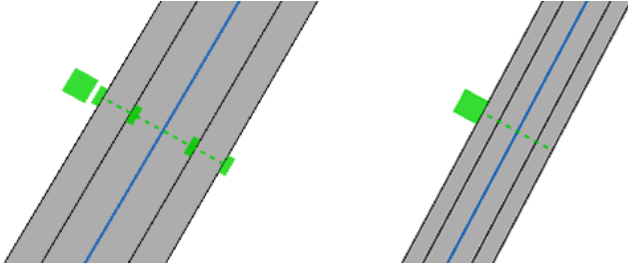
Specify the desired principle streets by dragging a principle street handle from one street to another.

#### **Note:**

Dragging any principle street handle into the yellow circle will clear the principle street at the intersection results in the default behavior.

#### *Street width handles*

You can edit the dimensions of a street with the **StreetWidth** handle. Drag the small green rectangles to individually set the width and offset of the street shape and the widths of the right and left sidewalks. Press the **Shift** key while dragging to make symmetrical adjustments. The big green rectangle is the combined width handle. Dragging it changes street width and sidewalk width simultaneously.




The **StreetWidth** handle is shown. Left: When zoomed in, sidewalk and street handles and a combined width handle are shown. Right: Zoomed out, only the combined width handle is shown.



**Note:**

You can set street width parameters numerically as well in the [street parameters](#).

# Cleanup graph

You can use the **Cleanup graph** tool  for fast cleanup of graph networks by merging nodes, merging segments, and creating nodes at intersecting segments.

Often imported, merged, or self-drawn graph networks may contain the following issues:

- Duplicate or close-by nodes
- Duplicate or close-by segments
- Intersecting segments that do not have nodes where segments intersect

These graph networks produce a number of problems when creating street shapes or extracting lots.

You can run the **Cleanup graph** tool  in the following ways:


- Click the **Cleanup graph** tool .
- Click **Graph > Cleanup Graph** from the main menu.


 **Note:**


This tool operates on a selection of graph segments. Unselected segments stay unchanged. When merging, the nodes of the selected segments are merged.

## Cleanup graph settings

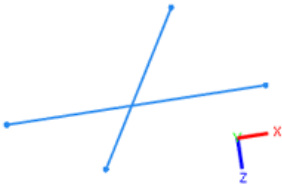
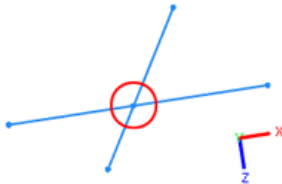

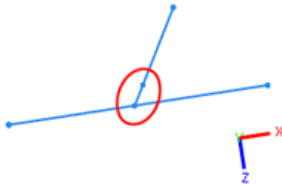


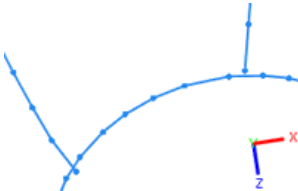
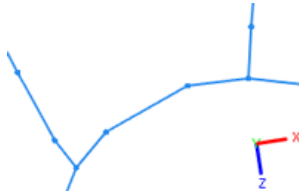
The checked operations (intersect, snap, merge, and resolve shape conflicts) are run one after another. The following options can be set:

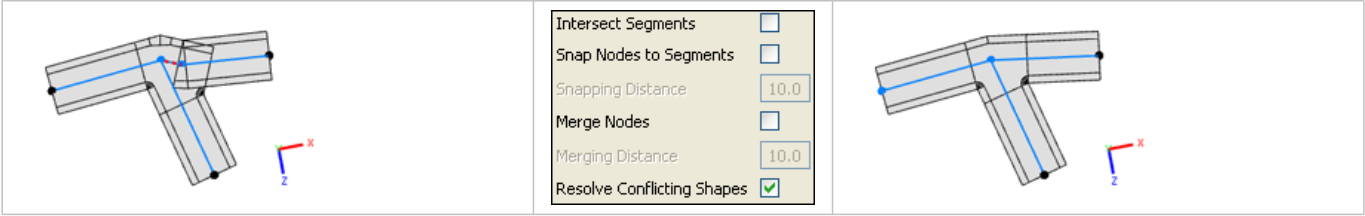
Option	Description
<b>Intersect Segments</b>	When checked, creates a new node at the intersection of segments.
<b>Snap Nodes to Segments</b>	When checked, nodes snap to segments.  <b>Note:</b> Nodes with smaller street widths always snap to segments with larger street widths. Node street width is defined as the maximal street width of the adjacent segments.
<b>Horizontal Snapping Distance</b>	The maximal horizontal distance between a node and a target segment. This option is only relevant if the <b>Snap Nodes to Segments</b> option is checked.
<b>Vertical Snapping Distance</b>	The maximal vertical distance between a node and a target segment. This option is only relevant if the <b>Snap Nodes to Segments</b> option is checked.
<b>Merge Nodes</b>	When checked, nodes that are close to each other are merged.
<b>Horizontal Merge Distance</b>	Nodes that are closer than this distance in a horizontal direction are merged into one. This option is only relevant if the <b>Merge Nodes</b> option is checked.
<b>Vertical Merge Distance</b>	Nodes that are closer than this distance in a vertical direction are merged into one. This option is only relevant if the <b>Merge Nodes</b> option is checked.

Option	Description
<b>Resolve Conflicting Shapes</b>	<p>When checked, the tool collapses all street segments that cause street shape conflicts. This is run iteratively until no conflicts exist.</p> <p> <b>Note:</b> Segments with the smallest minimal adjacent node valence are collapsed first; that is, this segment valence determines the order of the segment collapse iteration.</p>


 **Note:**  
This tool operates planar in the x-z plane. The y-coordinate is ignored. Running this tool on graph networks containing segments on different y-levels is not recommended.


Examples

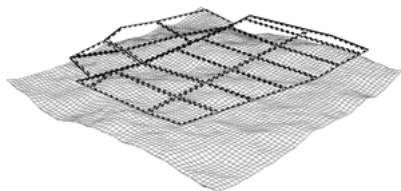
Simple intersections		
	<div><div>Intersect Segments:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Snap Nodes to Segments:</div><div><input type="checkbox"/></div></div> <div><div>Snapping Distance:</div><div>10.0</div></div> <div><div>Merge Nodes:</div><div><input type="checkbox"/></div></div> <div><div>Merging Distance:</div><div>10.0</div></div> <div><div>Resolve Conflicting Shapes</div><div><input type="checkbox"/></div></div>	
Snap nodes to segments		
	<div><div>Intersect Segments:</div><div><input type="checkbox"/></div></div> <div><div>Snap Nodes to Segments:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Snapping Distance:</div><div>10.0</div></div> <div><div>Merge Nodes:</div><div><input type="checkbox"/></div></div> <div><div>Merging Distance:</div><div>10.0</div></div> <div><div>Resolve Conflicting Shapes</div><div><input type="checkbox"/></div></div>	
Simple merge		
	<div><div>Intersect Segments:</div><div><input type="checkbox"/></div></div> <div><div>Snap Nodes to Segments:</div><div><input type="checkbox"/></div></div> <div><div>Snapping Distance:</div><div>10.0</div></div> <div><div>Merge Nodes:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Merging Distance:</div><div>10.0</div></div> <div><div>Resolve Conflicting Shapes</div><div><input type="checkbox"/></div></div>	
Full cleanup		
	<div><div>Intersect Segments:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Snap Nodes to Segments:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Snapping Distance:</div><div>10.0</div></div> <div><div>Merge Nodes:</div><div><input checked="" type="checkbox"/></div></div> <div><div>Merging Distance:</div><div>10.0</div></div> <div><div>Resolve Conflicting Shapes</div><div><input type="checkbox"/></div></div>	
Resolve conflicting shapes		



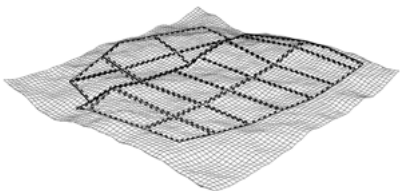
# Align streets to terrain

You can use the **Align streets to terrain** tool  to align graph networks to a terrain (map layers with the elevation attribute defined) or to the y=0 level. You can access the tool in the following ways:

- Click the **Align streets to terrain** tool  on the main toolbar.
- Click **Graph > Align streets to terrain** in the main menu.



Non-aligned graph network



Graph network aligned to a terrain.

## Settings

The following parameters control the alignment:

<b>Align function</b>	The alignment function to apply to the nodes of the graph. <ul style="list-style-type: none"><li>• <b>Project All</b>—Projects all nodes onto the terrain.</li><li>• <b>Project Below</b>—Projects the nodes located below the terrain only.</li></ul>
<b>Terrain</b>	The terrain to align the graph. All map layers with an elevation attribute plus the y=0 level are listed here.
<b>Offset</b>	The offset to add after alignment to the y-coordinate of the nodes.

# Generate street networks

The **Grow streets** tool can be used to generate typical street networks. Three street patterns (organic, raster, and radial) can be arbitrarily combined. The dialog box with a number of settings allows the user to generate street networks according to their needs.

The tool can be used in the following ways:

- Create a street network (deselect all and start the generator).
- Extend an existing street network by selecting an existing street layer before growing.
- Extend a part of an existing street network (select some streets of an existing street network and apply the tool).

You can access the **Grow streets** tool by clicking **Graph > Grow Streets** in the main menu.

## Note:

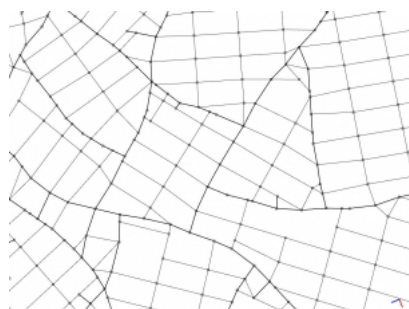
The algorithm distinguishes between major and minor streets. Basically, major streets are created until they enclose an area, called a quarter. Then the quarter is subdivided by minor streets. The algorithm continues creating major streets and so on.

The wizard creates a user-chosen number of streets. Each new street is added locally to the existing street network, depending on a number of settings (where the street pattern is probably the most important).

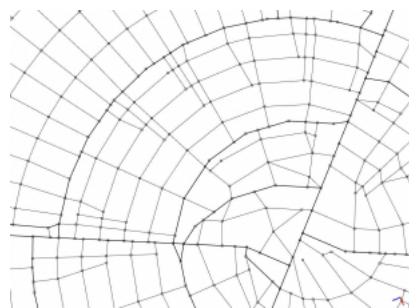
- Basic settings—Consist of the number of streets to generate and the street patterns.
- Pattern-specific settings—Define the street patterns more precisely.
- Advanced settings—Specify the algorithm behavior and the algorithm constraints.
- Environment settings—Include obstaclemaps to restrict the growth area and terrains to adapt the created streets to the elevation.
- Street settings—Define the street settings of the created streets.

## Basic settings

The basic settings consist of the number of streets, the street patterns, and the street lengths.



*Organic major street pattern and raster minor street pattern*



*Radial pattern for both major and minor streets*

Street patterns need two street lengths: The long and the short length. The organic pattern needs just one length (the short length is used).

Basic settings have the following parameters:

<b>Number of streets</b>	The number of streets to generate in total.
<b>Pattern of major streets</b>	The street pattern used for major streets: organic, raster, or radial.
<b>Pattern of minor streets</b>	The street pattern used for minor streets: organic, raster, or radial.
<b>Long length</b>	The average length of the long streets (used for the raster and radial pattern).
<b>Long length deviation</b>	Before the subdivision of a quarter, the length of the long streets is randomly set within the interval [Long Length - Long Length Deviation, Long Length + Long Length Deviation]. In the case of the organic pattern, this length is randomly set for each new street.
<b>Short length</b>	The average length of the short streets (used for all patterns). See Short Length Deviation.
<b>Short length deviation</b>	Before the subdivision of a quarter, the length of the short streets is randomly set within the interval [Short Length - Short Length Deviation, Short Length + Short Length Deviation]. In the case of the organic pattern, this length is randomly set for each new street.

## Advanced settings

The advanced settings specify the algorithm behavior and the algorithm constraints.

<b>Snapping distance</b>	If the distance between a new street node and an existing one is smaller than this snapping distance, the new node is snapped into the existing one. This way, you can control the minimal distance between any two nodes of the street network. Note that only half of this distance is applied if a minor street intersects with a major street (to model more realistic quarter subdivision).
<b>Minimal angle</b>	The minimal angle between any two neighbor streets of the street network. It is guaranteed that no smaller angle originates.
<b>Street to crossing ratio</b>	Using the street to crossing ratio, you can influence the average size of quarters. The algorithm tries to fulfill this ratio (only the major streets and major crossings count!). Quarters are large if this ratio is large and small if the ratio is small. $\text{Street to Crossing Ratio} = \frac{\text{\#Major street nodes}}{\text{\#Major crossing nodes}}$ where a street node is one with valence (valence = number of outgoing graph segments) equal to 2 and a crossing node is one with valence greater than 2.
<b>Development center preference</b>	If large, street nodes near the center are developed more likely than nodes outside. The center is defined as the center of mass of all selected nodes. If small, all nodes are equally likely to be developed.
<b>Angle offset of major streets</b>	Before major street creation, this offset angle is added to the proposed street angle.
<b>Angle offset of minor streets</b>	Before minor street creation, this offset angle is added to the proposed street angle.

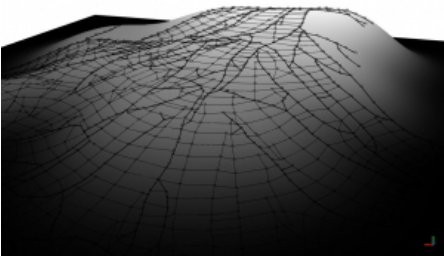
## Environment settings

Using environment maps, you can define boundary conditions such as terrains or obstaclemaps.

<b>Adapt to elevation</b>	Enables or disables adaption to elevation.
<b>Critical slope</b>	Only proposed streets with a slope greater than the critical slope are adapted.
<b>Maximal slope</b>	The maximal allowed street slope.
<b>Adaption angle</b>	The maximal angle a proposed street is adapted (rotation around the y-axis).
<b>Heightmap</b>	If a heightmap is selected, the new streets align to the heightmap and, if enabled (see below), the streets adapt to the elevation. In the combo box, all terrains of the scene are listed. A terrain is an attribute map that defines the float attribute elevation.
<b>Obstaclemap</b>	<div>If an obstaclemap is selected, the new street nodes avoid the obstacles. The street algorithm is able to avoid and circumnavigate obstacles. In the combo box, all obstaclemaps are listed. An obstaclemap is an attribute map that defines the Boolean attribute obstacle.</div> 

### Adaption to elevation

The adaption of new streets to elevation is active if a terrain is selected and the adaption is enabled. If the proposed street's length is close to Long Length, the proposed street is adapted to go along an elevation contour line, in other words, the goal is to create a street with slope 0.



Adaption to elevation

If its length is close to Short Length, the proposed street is adapted in order go maximally elevation up or downward.

### Pattern-specific settings

The pattern-specific settings specify the street patterns in more detail.

<b>Max. bend angle (organic)</b>	<p>The maximal bending angle of organic streets. The angle of a new organic street is randomly set within the interval [Proposed Angle - Max. Bend Angle, Proposed Angle + Max. Bend Angle]. It defines the legal area of street expansion (the green area in the figure below).</p>
<b>City center x (radial)</b>	<p>The city center used for the radial pattern along the x-axis. Streets go radial or centripetal around or outside of the center.</p>
<b>City center y (radial)</b>	<p>The city center used for the radial pattern along the y-axis. Streets go radial or centripetal around or outside of the center.</p>
<b>Max. bend angle (radial)</b>	<p>The maximal bending angle of radial streets. The algorithm tries to adapt the proposed street to either the radial or the centripetal direction. The maximal adaption angle is restricted by this parameter.</p>
<b>Street Alignment (radial)</b>	<p>There is a long and a short length for radial streets. This parameter decides whether the long streets are aligned radial or centripetal, or if the alignment is chosen randomly; (left) Radial street alignment (right) Centripetal street alignment.</p>

## Street settings

Street and sidewalk settings are assigned to the new streets. If an existing street is extended, its street and sidewalk settings are copied to the new street. Otherwise, street and sidewalk settings are randomly set according to the following parameters.

### General parameters

<b>Calculate width using street integration</b>	Uses the graph topology to <a href="#">calculate street and sidewalk widths</a> using graph connectivity (slower), otherwise randomly distributed street widths are used (faster).
---	--

### Street integration parameters

<b>Minimum number of street lanes</b>	The minimum number of lanes that each street may have.
<b>Maximum number of street lanes</b>	The maximum number of lanes that each street may have.
<b>Minimum sidewalk width</b>	The minimum width of each street's sidewalks.
<b>Maximum sidewalk width</b>	The maximum width of each street's sidewalks.

### Randomly distributed parameters and block subdivision

<b>Minimum number of major street lanes</b>	The minimum number of major lanes that each street may have.
<b>Maximum number of major street lanes</b>	The maximum number of major lanes that each street may have.
<b>Sidewalk width of major streets</b>	The average sidewalk width of a major street. See Sidewalk Width Deviation of Major Streets.
<b>Sidewalk width deviation of major streets</b>	The sidewalk width deviation for major streets. The sidewalk width is randomly set within the interval [Sidewalk Width of Major Streets - Sidewalk Width Deviation of Major Streets, Sidewalk Width of Major Streets + Sidewalk Width Deviation of Major Streets].
<b>Minimum number of minor street lanes</b>	The minimum number of minor lanes that each street may have.
<b>Maximum number of minor street lanes</b>	The maximum number of minor lanes that each street may have.
<b>Sidewalk width of minor streets</b>	The average sidewalk width of a minor street. See Sidewalk Width Deviation of Minor Streets.
<b>Sidewalk width deviation of minor streets</b>	The sidewalk width deviation for minor streets. The sidewalk width is randomly set within the interval [Sidewalk Width of Minor Streets - Sidewalk Width Deviation of Minor Streets, Sidewalk Width of Minor Streets + Sidewalk Width Deviation of Minor Streets].

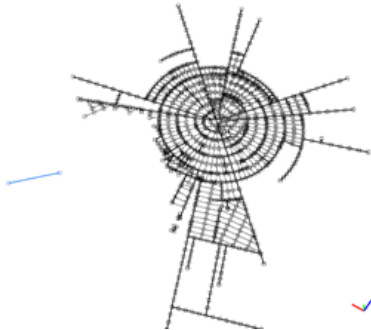
## Block Subdivision

Specifies the subdivision algorithm to use: Recursive Subdivision, Offset Subdivision, Skeleton Subdivision, No Subdivision, or Block Shape Creation Disabled. See [Block parameters](#) for more information.

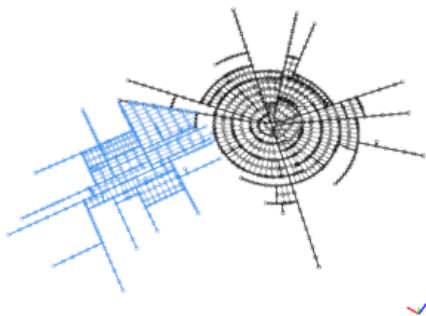
## Street pattern examples

### Workflow examples as impressions

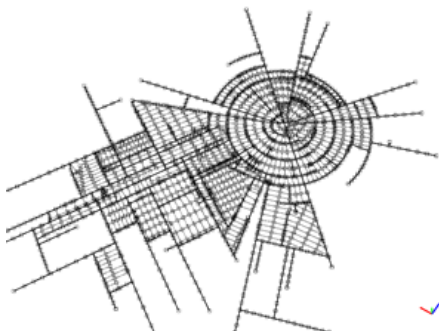
- Create a separated single street.



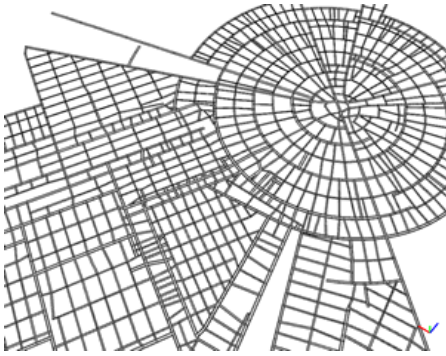
- First growth phase with raster pattern.



- Second growth phase with raster pattern.

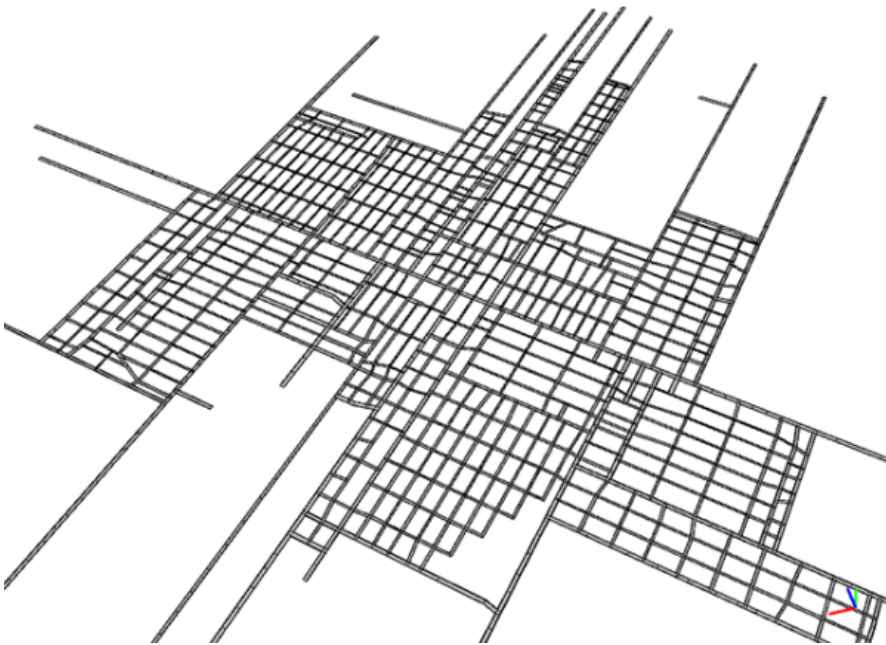


- The two networks are connected.



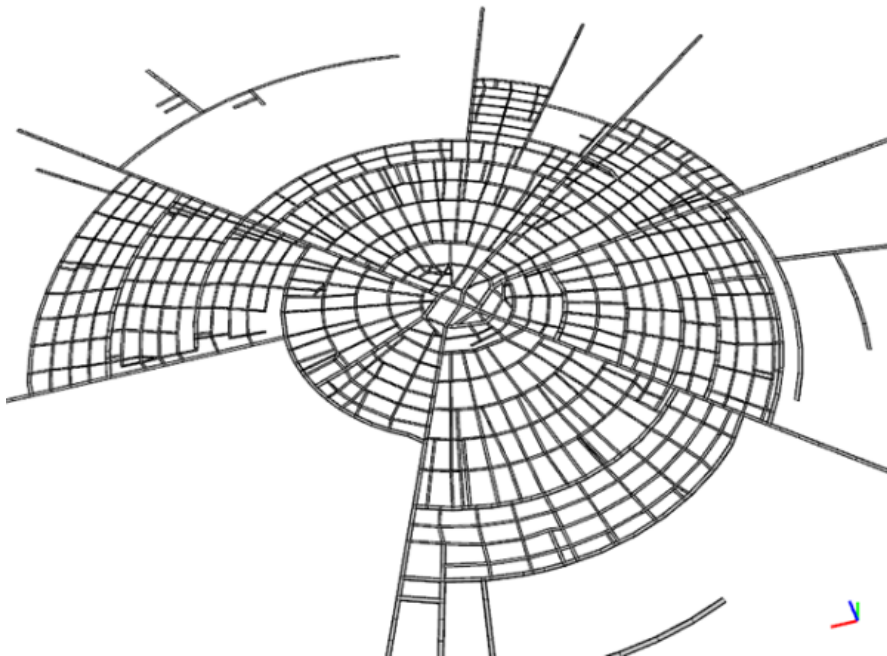
## Parameter sets

- Raster pattern.



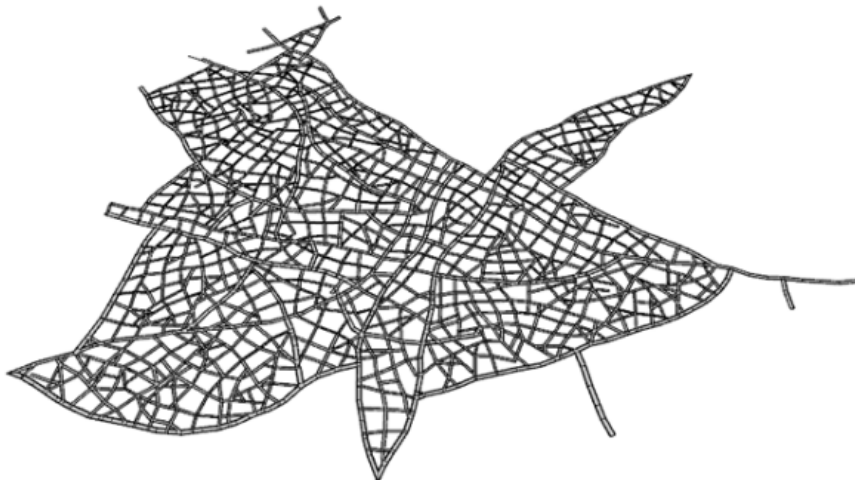
<b>Advanced Settings</b>	
Snapping distance	30.0
Minimal angle	22.5
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	0.0
Angle offset of minor streets	0.0
<b>Basic Settings</b>	
Number of streets	1500
Pattern of major streets	RASTER
Pattern of minor streets	RASTER
Long length	150.0
Long length deviation	50.0
Short length	80.0
Short length deviation	20.0
<b>Environment Settings</b>	
<b>Pattern Specific Settings</b>	
Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	RANDOM
<b>Street Width Settings</b>	

- Radial pattern.



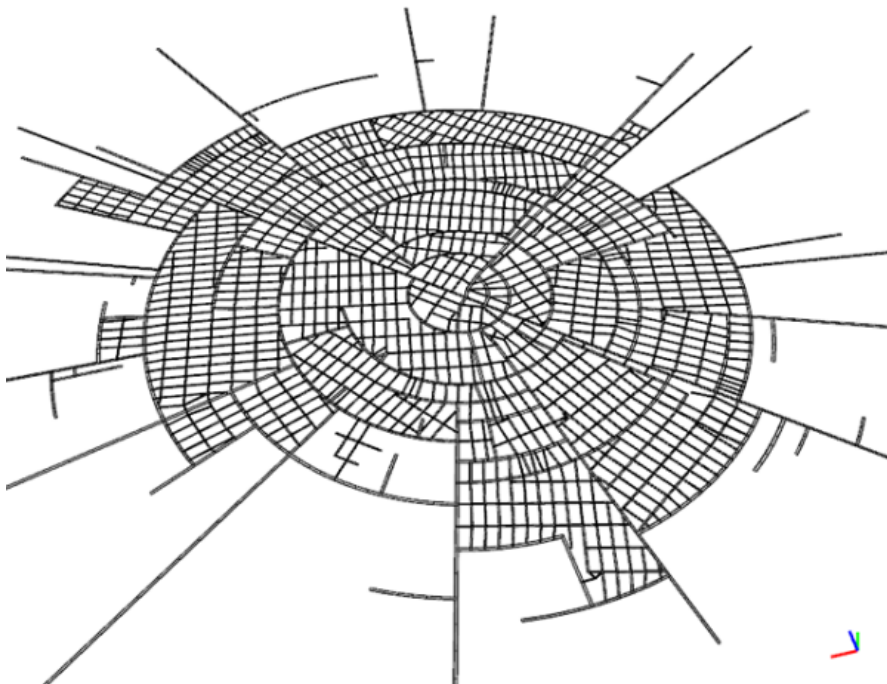
<b>Advanced Settings</b>	
Snapping distance	30.0
Minimal angle	22.5
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	0.0
Angle offset of minor streets	0.0
<b>Basic Settings</b>	
Number of streets	1500
Pattern of major streets	RADIAL
Pattern of minor streets	RADIAL
Long length	150.0
Long length deviation	50.0
Short length	80.0
Short length deviation	20.0
<b>Environment Settings</b>	
<b>Pattern Specific Settings</b>	
Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	RANDOM
<b>Street Width Settings</b>	

- Organic pattern.



<b>Advanced Settings</b>	
Snapping distance	30.0
Minimal angle	22.5
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	0.0
Angle offset of minor streets	0.0
<b>Basic Settings</b>	
Number of streets	2000
Pattern of major streets	ORGANIC
Pattern of minor streets	ORGANIC
Long length	150.0
Long length deviation	50.0
Short length	80.0
Short length deviation	20.0
Environment Settings	
<b>Pattern Specific Settings</b>	
Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	RANDOM
Street Width Settings	

- Radial major streets with raster pattern on minors.



**Advanced Settings**

Snapping distance	30.0
Minimal angle	22.5
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	0.0
Angle offset of minor streets	0.0

**Basic Settings**

Number of streets	1500
Pattern of major streets	RADIAL
Pattern of minor streets	RASTER
Long length	150.0
Long length deviation	50.0
Short length	80.0
Short length deviation	20.0

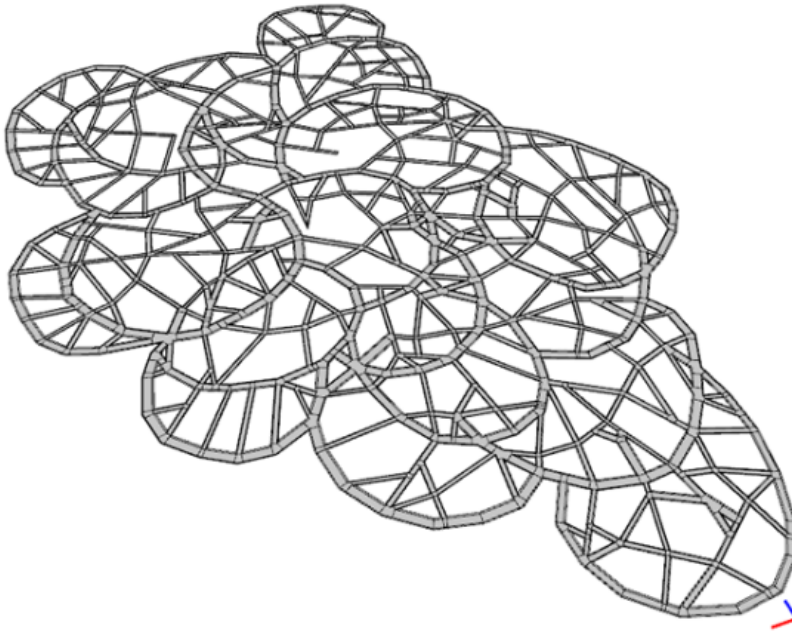
**Environment Settings**

**Pattern Specific Settings**

Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	CENTRIPETAL

**Street Width Settings**

- Organic circle pattern.



**Advanced Settings**

Snapping distance	30.0
Minimal angle	22.5
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	20.0
Angle offset of minor streets	20.0

**Basic Settings**

Number of streets	3000
Pattern of major streets	ORGANIC
Pattern of minor streets	RASTER
Long length	150.0
Long length deviation	50.0
Short length	80.0
Short length deviation	20.0

Environment Settings

**Pattern Specific Settings**

Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	CENTRIPETAL

Street Width Settings

- Honeycomb style.



Advanced Settings

Snapping distance

30.0

Minimal angle

22.5

Street to crossing ratio

4.0

Development center preference

2

Angle offset of major streets

60.0

Angle offset of minor streets

60.0

Basic Settings

Number of streets

3000

Pattern of major streets

ORGANIC

Pattern of minor streets

ORGANIC

Long length

150.0

Long length deviation

50.0

Short length

80.0

Short length deviation

20.0

Environment Settings

Pattern Specific Settings

Max. bend angle (organic)

15.0

City center x (radial)

0.0

City center z (radial)

0.0

Max. bend angle (radial)

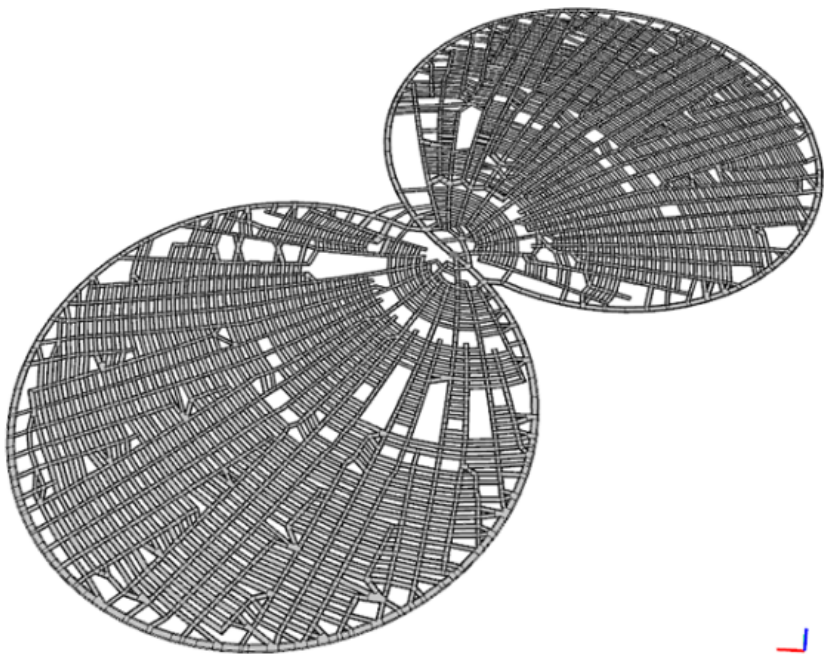
20.0

Street Alignment (radial)

CENTRIPETAL

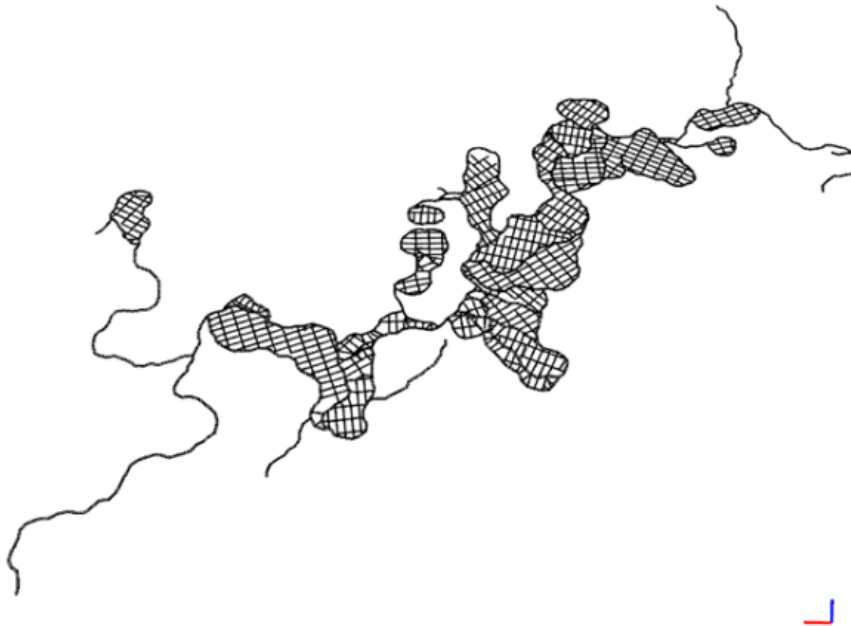
Street Width Settings

- The Glasses style.



<b>Advanced Settings</b>	
Snapping distance	30.0
Minimal angle	0.0
Street to crossing ratio	4.0
Development center preference	2
Angle offset of major streets	5.0
Angle offset of minor streets	10.0
<b>Basic Settings</b>	
Number of streets	5000
Pattern of major streets	RASTER
Pattern of minor streets	RADIAL
Long length	50.0
Long length deviation	20.0
Short length	20.0
Short length deviation	10.0
<b>Environment Settings</b>	
<b>Pattern Specific Settings</b>	
Max. bend angle (organic)	15.0
City center x (radial)	0.0
City center z (radial)	0.0
Max. bend angle (radial)	20.0
Street Alignment (radial)	RANDOM
<b>Street Width Settings</b>	

- Organic distribution of rasters.



^ Advanced Settings

Snapping distance

30.0

Minimal angle

22.5

Street to crossing ratio

10.0

Development center preference

2

Angle offset of major streets

0.0

Angle offset of minor streets

0.0

^ Basic Settings

Number of streets

500

Pattern of major streets

ORGANIC

Pattern of minor streets

RASTER

Long length

150.0

Long length deviation

50.0

Short length

80.0

Short length deviation

20.0

^ Environment Settings

^ Pattern Specific Settings

Max. bend angle (organic)

45.0

City center x (radial)

0.0

City center z (radial)

0.0

Max. bend angle (radial)

20.0

Street Alignment (radial)

RANDOM

^ Street Width Settings

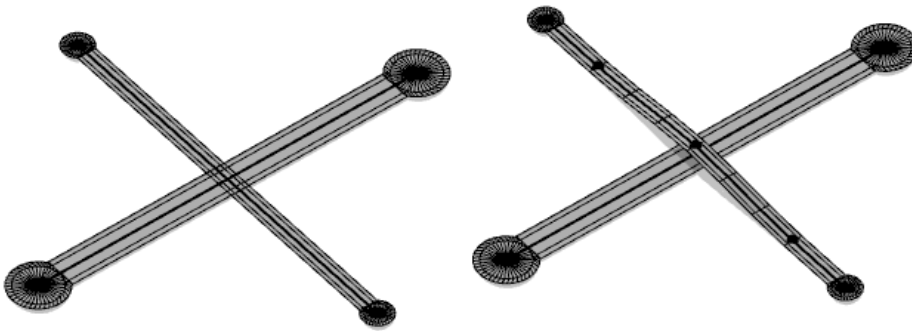
# Generate a bridge

Imported or manually created street networks often lack elevation data, which is necessary for the 3D display of crossing streets. The **Generate Bridges** tool can automatically create such data. This can be done by clicking **Graph** > **Generate Bridges** in the main menu. It operates either on the current street selection or on all streets when nothing is selected.

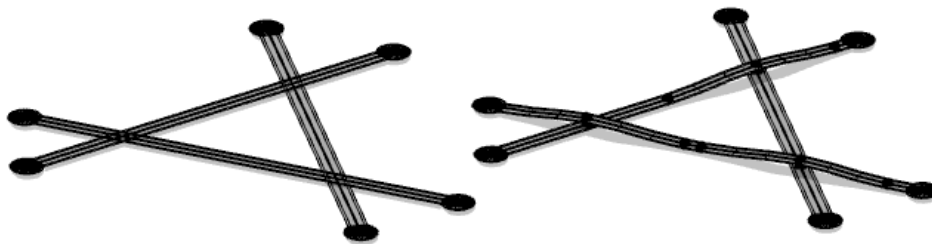
## Note:

The wider street is kept naturally at its original level, while the thinner street is raised. Also, note that new street nodes are inserted at the correct distances from the crossing point, defined by the maximal defined slope. The vertical tangent components are automatically adjusted.

The **Generate Bridges** tool adds elevation data to the streets:



(left) Original streets; (right) applying the *Generate Bridges* tool adds elevation data

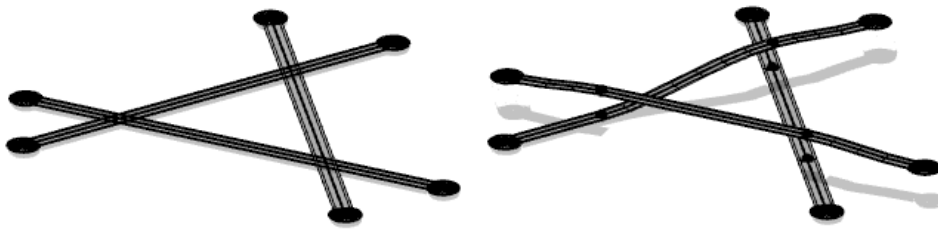


(left) Multiple Streets without elevation; (right) multiple streets with elevation data added after tool

## Generate bridges settings

### Level height

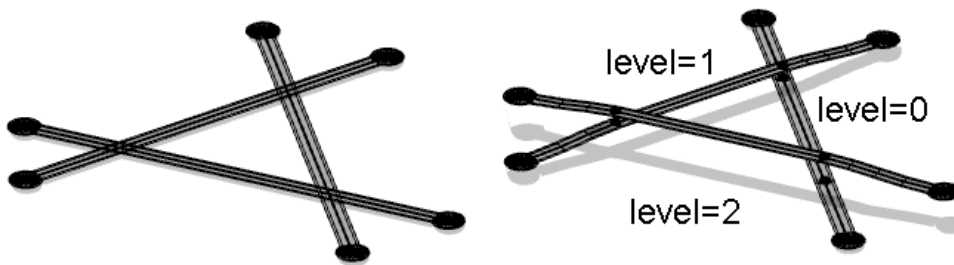
Vertical distance to be set between two crossing streets. Note that the Ramp maximum slope influences the resulting node elevations.



(left) Original; (right) level height = 20

## Object attribute for level (optional)

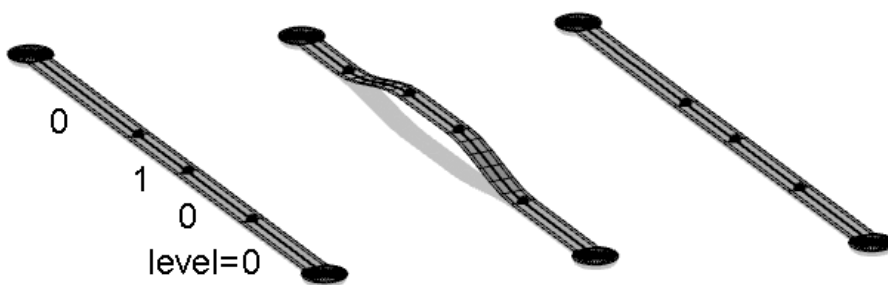
The height coordinate can be calculated from specific object attributes of streets. The height coordinate is set to Level height multiplied with the indicated attribute name. Attributes can either originate from imported data or be manually assigned, allowing full control of the vertical street layering.



(left) Original; (right) manually set level attributes enable full control of vertical street layering

## Only apply level when streets cross

Sometimes, imported GIS data, such as OSM data, may contain faulty attribute values that cause the creation of elevated parts of streets. This option activates or deactivates the vertical alignment in regions, where actually no other streets cross the street of interest.



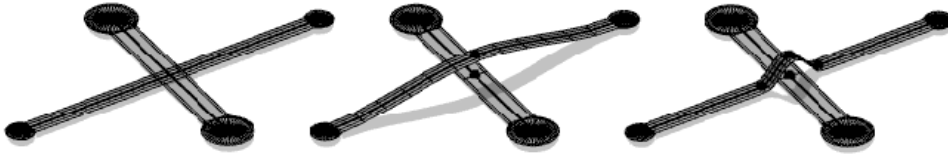
(left) Original with faulty level attribute value; (middle) Option unchecked; (right) Option checked

## Object attribute for absolute height (optional)

In contrast to the object attribute for level, this attribute allows direct specification of absolute heights. When a street has this attribute, the level height is ignored.

## Ramp maximum slope

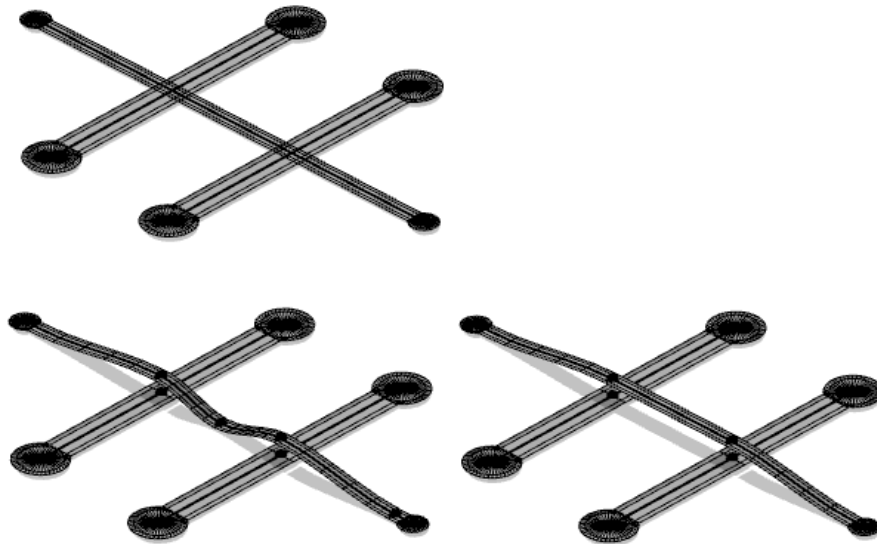
Maximum slope of ramps (vertical climb per horizontal unit).



(left) Original; (middle) maximum slope = 0.2; (right) maximum slope = 1.0

## Bridge join preference

If a street contains multiple bridges in a row, they are linked together according to this value. Low values: unlikely to join; high values: always join.



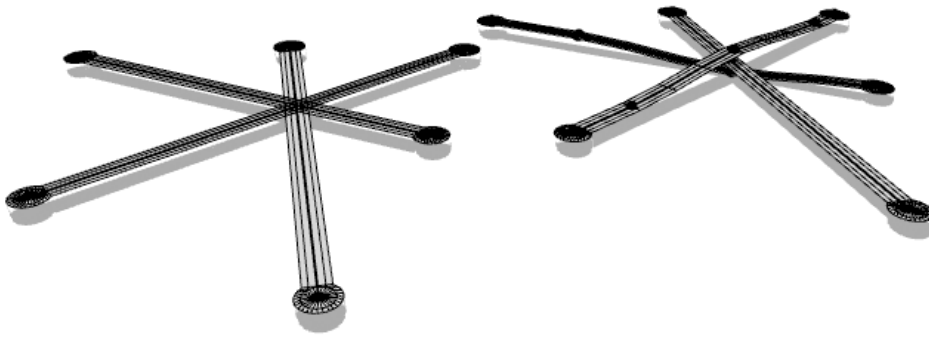
(top) Original; (bottom left) Bridge join = 0.1; (bottom right) Bridge join = 1.0

## Lock nonzero heights

Do not change height of nodes with nonzero position.

## Allow tunnels

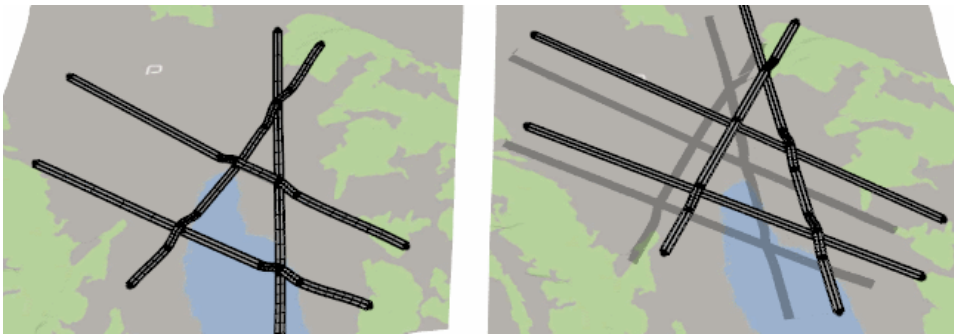
Allow tunnels (streets below zero height). Note that this is quite a rare state to construct. Note also that the Shadow Plane is rendered at the smallest (thus negative) elevation value.



*(left) Original; (right) solved with tunnels*

## Use visible terrain

Treat all heights as relative above terrain (if any). This causes bridges to follow the terrain.



*(left) Use visible terrain; (right) ignore visible terrain*

# Editing

# Editing


You can use a variety of editing tools in CityEngine that allow you to modify and adjust shapes, graphs, or terrain.

Shapes	Make edits to shapes using such tools as the <a href="#">transform tools</a> , <a href="#">set street edge</a> , <a href="#">reverse normals</a> , or <a href="#">combining shapes</a> .
Graphs	Make edits to graphs using such tools as the <a href="#">transform tools</a> , <a href="#">simply graphs</a> , or <a href="#">analyze graphs</a> .
Terrain	Make edits to terrain using such tools as <a href="#">interactive terrain editing</a> or <a href="#">align terrain to shapes</a> .

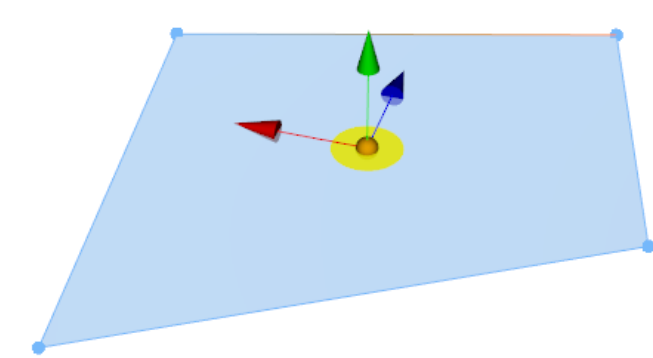
# Use transform tools

The transform tools allow you to manipulate and manually adjust your objects by moving, scaling, and rotating.

## Move

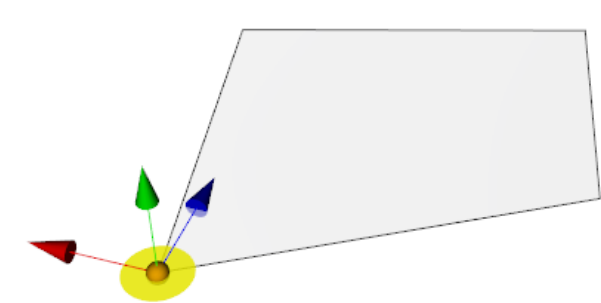
To move objects, click the **Transform Move** tool  (W). You can also click **Edit > Move Tool** in the main menu.

Drag the handles to move along individual axes. Use the yellow handle to move the selection on the xz or object planes. Use the orange sphere to drag the selection onto any plane and snap to faces, vertices, edges, or guides. You move selected objects and components along the x, y, and z axes. These axes are defined by the current [reference systems](#).




The Move Tool is shown with a shape.

Select vertices and edges to transform them separately from the whole shape.




The Move Tool is shown with a single vertex.

You can apply transform move options, such as reference system, position and orientation, translation, or snapping in the **Transform Move** tool options .


## Transform Move tool options

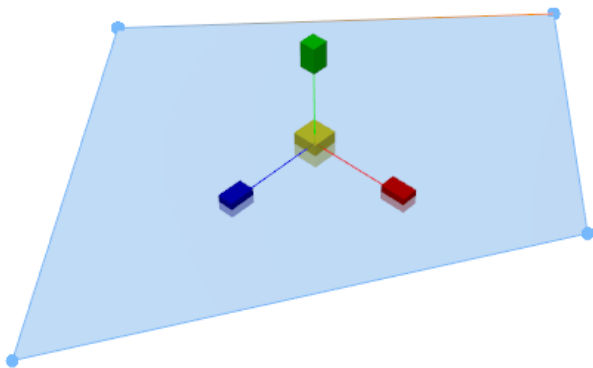
The **Transform Move** tool options  include the following:

 <b>Transform Move</b>	
<b>Reference System</b>	See <a href="#">Reference system</a> .
<b>Adjust the Position and Orientation</b>	See <a href="#">Adjust the Position and Orientation</a> .


<b>Translation (m)</b>	<p>Distance in meters to move objects along the axes of the reference system. The color of the boxes match the axes in the Viewport window.</p> <ul style="list-style-type: none"><li>• Press <b>Enter</b> to apply a value.</li><li>• Press <b>Tab</b> to move to the next input field.</li><li>• Apply only one value at a time.</li></ul>
<b>Snapping</b>	<p>Turn <b>snapping options</b> on and off. Select objects or components and do one of the following:</p> <ul style="list-style-type: none"><li>• Drag the orange sphere to snap to faces, vertices, edges, or guides on any plane.</li><li>• Drag the yellow disc to snap to vertices, edges, or guides on the xz plane.</li><li>• Drag the axes handles to snap to vertices, edges, or guides along the selected axis.</li><li>• Press <b>Shift</b> to temporarily turn on or off snapping.</li></ul>
<b>Copy on Move</b>	<p>Create a copy of the selection when moving objects. Press <b>Ctrl</b> to temporarily turn on or off this option.</p>

Scale

To scale objects, click the **Transform Scale** tool  ( **E** ). You can also click **Edit > Scale Tool** in the main menu. You can scale selected objects and components along the x, y, and z axes. Use the yellow handle to uniformly scale along all axes.




Scale tool with shape


You can apply transform scale options, such as reference system, position and orientation, scale, or snapping in the **Transform Scale** tool options .

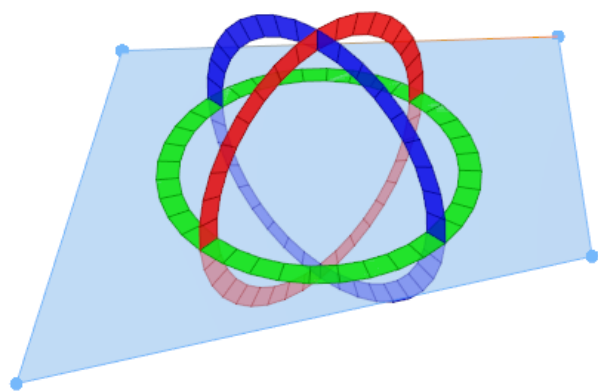
Transform Scale tool options

The **Transform Scale** tool options  include the following:

✂ Transform Scale	
Reference System	See <a href="#">Reference system</a> .
Adjust the Position and Orientation	See <a href="#">Adjust the Position and Orientation</a> .
Transform Individually	When multiple objects are selected, scale individual objects around their centroid with axes defined by the reference system.*
Scale (%)	<div>Scale objects along the axes of the reference system.</div> <ul style="list-style-type: none"><li>• Press <b>Enter</b> to apply a value.</li><li>• Press <b>Tab</b> to move to the next input field.</li><li>• Apply only one value at a time.</li><li>• —Enables uniform scaling along the axes.</li></ul>
Snapping	Turn <a href="#">snapping options</a> on and off. Press <b>Shift</b> to temporarily turn on or off snapping.

Rotate

To rotate objects, click the **Transform Rotate** tool  ( **R** ). You can also click **Edit > Rotate Tool** in the main menu. You can rotate selected objects and components along the x, y, and z axes.



Rotate tool with shape

You can apply transform rotate options, such as reference system, position and orientation, rotation, or snapping in the **Transform Rotate** tool options ✂.

Transform Rotate tool options

The **Transform Rotate** tool options ✂ include the following:


✂ Transform Rotate	
Reference System	See <a href="#">Reference system</a> .
Adjust the Position and Orientation	See <a href="#">Adjust the Position and Orientation</a> .

<b>Transform Individually</b>	When multiple objects are selected, rotate individual objects around their centroid in which axes are defined by the reference system.*
<b>Rotation (°)</b>	Rotate objects around the axes of the reference system. <ul style="list-style-type: none"> <li>• Press <b>Enter</b> to apply a value.</li> <li>• Press <b>Tab</b> to move to the next input field.</li> <li>• Apply only one value at a time.</li> </ul>
<b>Snapping</b>	Turn <a href="#">snapping options</a> on and off. Select objects or components, drag a rotate axis handle, and do one of the following: <ul style="list-style-type: none"> <li>• Rotate to snap to global axes.</li> <li>• Rotate to snap to planes.</li> <li>• Rotate to snap to parallels of guides.</li> <li>• Press <b>Shift</b> to temporarily turn on or off snapping.</li> </ul>
<b>Snap to discrete angles</b>	Rotate in discrete intervals.

\* When the **Transform Individually** option is turned on, the position of the transform tool handles is always at the center of the [lead selection](#) and is not affected by changes to the **Adjust the Position and Orientation** tool position.

## Reference system

Transformations operate by default along the principal x, y, and z axes of a scene in the world reference system. Additionally, for customization and precision, you can apply transformations using the object reference system or create a custom reference system to save and apply to other objects in the scene.

To choose a different reference system to apply to object transformations, click the drop-down menu next to **Reference System** in the transform **Tool Options** windows . Click any of the following:


- **World**—Transform tools are aligned to the axes defined by the [scene coordinate system](#).
- **Object**—Transform tools are aligned to the objects' features (edges and normals).
- **{Custom reference system}**—Transform tools are aligned to the axes of a custom reference system that you create and save. See [Custom Reference System](#).

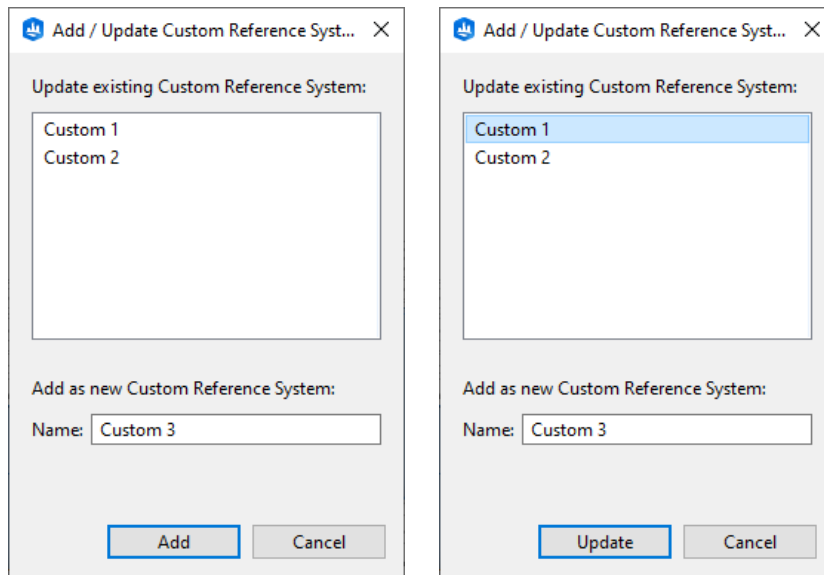
## Create a custom reference system

You can create a custom reference system (CRS) in which the orientation of the axes handles are saved for later to apply with the transform tools.

To add or update a CRS in a scene, do the following:

1. Click the **Adjust Position and Orientation** toggle key or press **O** to reposition the transform tool.  
The transform tool switches to the position and orientation axes handles.
2. Drag or rotate the axes handles to change orientation, or drag the orange sphere handles.  
As you rotate the axes, they snap similarly as the [Transform Rotate tool](#).

3. To add or update the current orientation of the axes as a CRS, click the **Save CRS As...** button  to open the **Add / Update Custom Reference System** dialog box.
4. Choose to add or update an existing CRS:
  - To add the CRS, name the new CRS and click **Add**.
  - To update an existing CRS, select a CRS in the list and click **Update**.



 **Note:**

The position of the CRS is not saved.

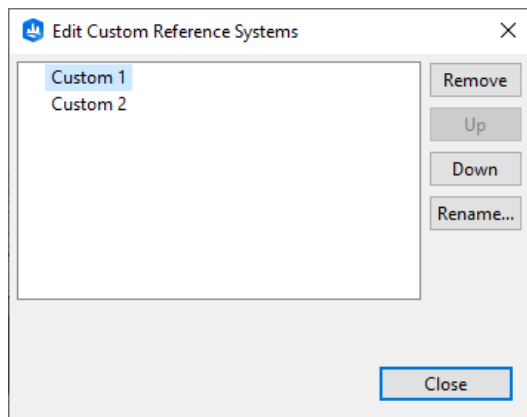
5. Click **Cancel** to close the window without adding or updating CRS.

## Edit CRS

You can rename, reorder, or remove an existing CRS by doing one of the following:

- Click **Edit CRSs...** in the **Reference System** drop-down menu.
- Click **Edit** > **Edit Custom Reference Systems...** in the main menu.

The **Edit Custom Reference Systems** dialog box appears.



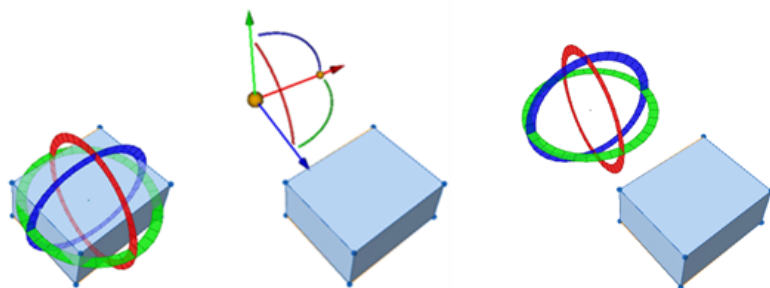
## Adjust Position and Orientation

The **Adjust Position and Orientation** tool in **☒ Tool Options** allows you to temporarily adjust both the position and the orientation of the transform tools handles.

1. Select the objects or components to transform.
2. Click the **Adjust Position and Orientation** toggle key or press **O** to reposition the transform tool.  
The transform tool switches to the position and orientation axes handles.
3. Move your mouse to a new location and click to reposition the tool to that place.  
You can also reposition the tool by dragging or rotating the axes handles or dragging the orange sphere handles. Rotate the reposition tool and it snaps similarly as the [Transform Rotate tool](#).

Once you move or rotate the reposition tool, the **Reference System** drop-down menu displays **Temporary**.

4. Click the **Adjust Position and Orientation** toggle key or press **O** to switch back to the transform tool.
5. Apply the new position and orientation with any of the transform tools.
6. To add or update the orientation of the axes as a CRS, click the **Save CRS As...** button **+** to open the **Add / Update Custom Reference System** dialog box.



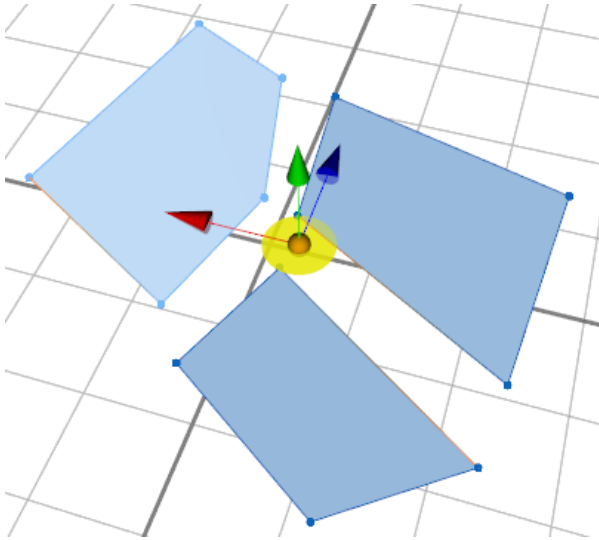
*The original rotation tool, moved outside of the shape, and the new tool position are shown.*

The tool automatically snaps to vertices, edges, faces, and guides as you reposition the tool.

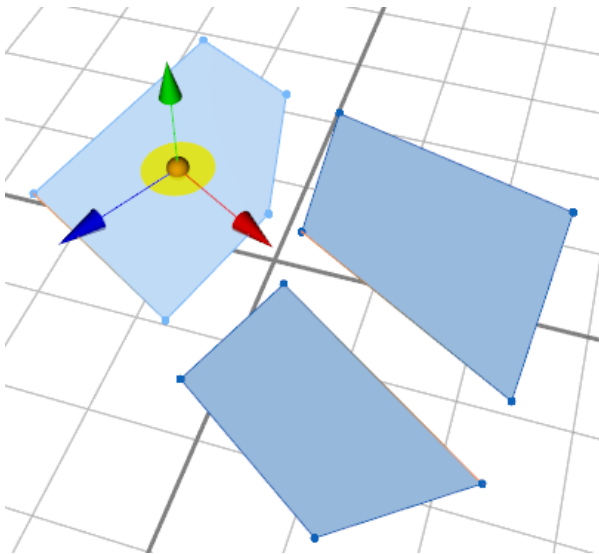
When you deselect, select another shape, or change reference systems in the drop-down menu, the reference system position is reset.

## Object selection

If object space is active and multiple objects are selected, the current reference system is determined by the currently selected object or lead object. The lead object is always the most recent individually selected object. You can change the lead object by pressing **Shift** and selecting the new lead object.



*A multiselection is shown with the lead selection in light blue. The Move Tool is positioned at the center of the multiselection.*



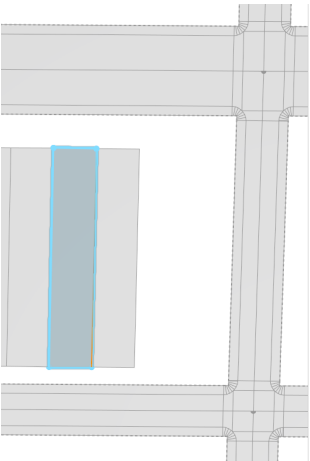
*The same multiselection is shown with the Move Tool and object reference system. Note the position of the Move tool (center of lead selection) and the x-axis aligned with the first edge (orange-blue border).*

# Edit shapes

# Compute first edges and street edges

The **Compute First/Street Edges** tool automatically calculates the first edges and street width attributes of the selected shapes. The tool finds the nearest street (within 100 meters) for every edge of a shape. The corresponding street width attribute is set to the width of the nearest street. The edge closest to a street is set as the first edge (edge 0). Click **Shapes > Compute First/Street Edges** in the main menu.

The selected shape displays the object attributes before the tool is applied in the **Object Attributes** section of the **Inspector** window. The first edge is the vertical edge highlighted in orange:



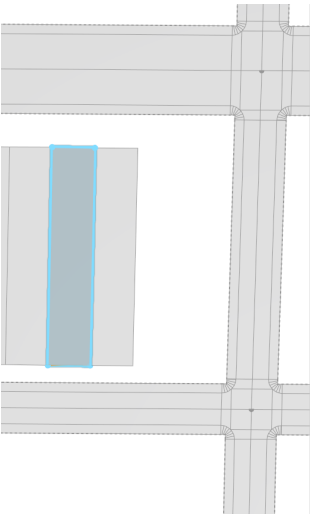
^ Object Attributes

FAR	0.7
Front	20
Height	35
OBJECTID	181
Rear	30
Side	7
address	4418 18TH ST
owner1	DETROIT LAND BANK AUTHORITY
parcelnum	10007320.
total_acre	0.106
zoning	R2

^ Materials

^ Vertices

After applying the tool, a `streetWidth` object array is created:



^ Object Attributes

FAR	0.7
Front	20
Height	35
OBJECTID	181
Rear	30
Side	7
address	4418 18TH ST
owner1	DETROIT LAND BANK AUTHORITY
parcelnum	10007320.
total_acre	0.106
zoning	R2
^ streetWidth[4]	[7, 7, 15, NaN]
0	7
1	7
2	15
3	NaN

Street width values are added to the `streetWidth` object array for edges that are set as street edges. The first edge is now set to the edge next to the bottom street. In the `streetWidth` object array, the corresponding Index `0` is assigned to the bottom street width, having a value of `7`. The top edge, Index `2`, is assigned to the top street width, having a value of `15`.

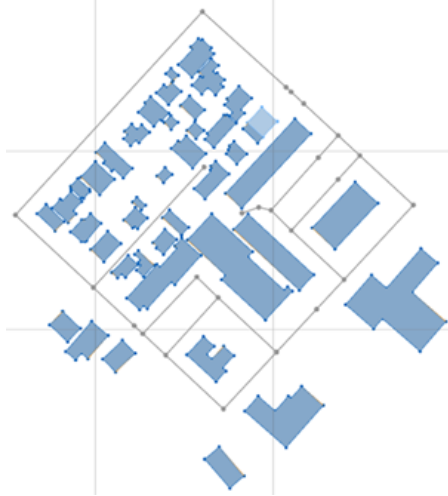
# Compute Edge Attributes tool

The **Compute Edge Attributes** tool computes orientation and street adjacency attributes for each edge of a shape. The computed edge attributes can then be used from CGA to drive model generation that is based on individual edge attributes.

## Use the tool

### Select the shapes

1. Select the shapes you want to apply the tool.



*Selected shapes are shown.*

The tool will compute the edge attributes for every selected shape with respect to the visible street segments, in other words, it is not necessary to select the street segments.

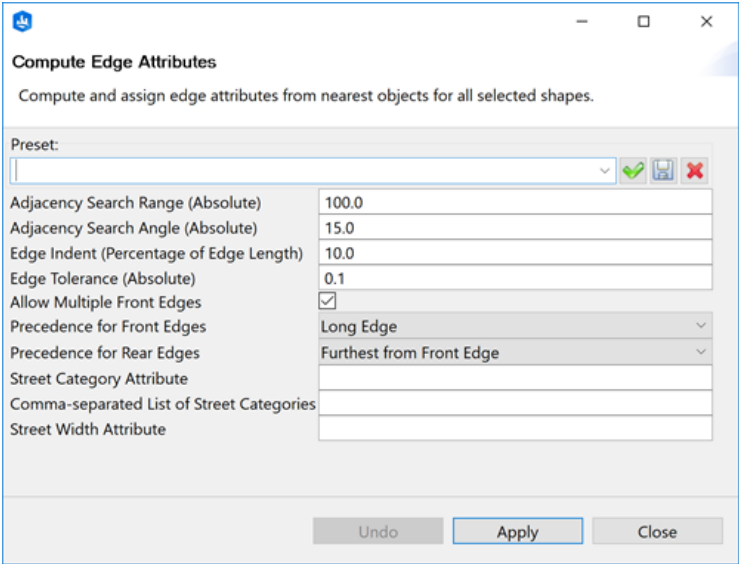


#### **Note:**

The algorithm currently only works for static shapes (see [Shapes](#)). To calculate edge attributes for dynamic shapes, you first need to convert them to static shapes using **Graph > Convert to Static Shapes**.

2. Click **Shapes > Compute Edge Attributes**.

The tool dialog box appears, providing numerous parameters to configure the algorithm. The algorithm and the parameters are explained in detail below. For general use cases, it is OK to use the default parameters and click **Apply** to run the algorithm.

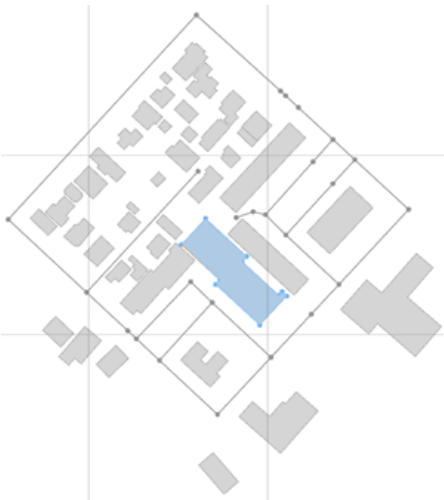


The Compute Edge Attributes dialog box is shown.

Inspect the computed edge attributes

After the tool is run, each selected shape is assigned the computed edge attributes. The following images illustrate how the shape attributes (in this case from an OSM data source) for a selected shape are enhanced with edge attributes after running the tool.

The following is the selected shape for which to compute the attributes:



The selected shape for computing attributes is shown.

The shape attributes before running the tool are as follows:

Object Attributes	
addr_postcode	8005
building	yes
building_levels	7
layer	1
name	Puls 5
opening_hours	Mo-Su 06:00-24:00
operator	Intershop Management
osm_id	10211605
shop	mall
website	http://www.puls5.ch
wheelchair	limited
wikidata	Q2117985
Add new object attribute...	

Attributes before computing edge attributes are shown.

The shape attributes after running the tool are as follows:

Object Attributes	
addr_postcode	8005
building	yes
building_levels	7
layer	1
name	Puls 5
opening_hours	Mo-Su 06:00-24:00
operator	Intershop Management
osm_id	10211605
shop	mall
website	http://www.puls5.ch
wheelchair	limited
wikidata	Q2117985
/edgeattr/orientations[30]	[front, front, side, side, s...]
/edgeattr/streetcategories[30]	[Major Edge, Major Edge, ρ, ...]
/edgeattr/streetwidths[30]	[4, 4, NaN, NaN, NaN, NaN, N...]
Add new object attribute...	

Attributes after computing edge attributes are shown.

As shown, the tool adds three shape attributes that contain a list of values for each edge, starting with the edge at index 0:

- /edgeattr/orientations—enumeration[]  
An array containing the orientation of each edge.
  - front: Assigned to edges oriented to the front, typically the main road.
  - rear: Assigned to edges on the opposite of front edges.
  - side: Assigned to edges between front and rear.
  - inner: Assigned to edges that are part of a shape’s hole.
- /edgeattr/streetcategories—string[]  
An array containing the street category for each edge, such as Major Edge or Highway.
  - <category>: Assigned to the edge if it is facing a street.
  - NULL: Assigned to the edge if it is not facing a street.

- `/edgeattr/streetwidths—float[]`

An array containing the street width for each edge.

- `<width>`: Assigned to the edge if it is facing a street.
- `NaN`: Assigned to the edge if it is not facing a street.

## Use computed edge attributes from CGA

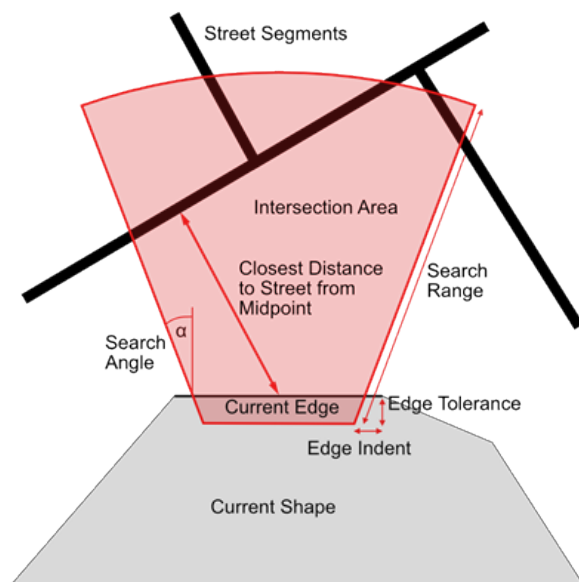
The computed edge attributes can be accessed from CGA using the [edge attribute functions](#).

## Description of algorithm and parameters

The algorithm to calculate the edge attributes runs in three phases: Phase 1 computes the adjacency information; Phase 2 determines the edge orientation; and Phase 3 creates the shape's edge attributes.

### Computation of adjacency information

Computation of adjacency information is based on an algorithm that looks for the closest street edges from the current shape edge within a given range. The range is controlled by a number of parameters, as illustrated in the following image:



Computation of adjacency information is shown.

- Adjacency Search Range—Absolute range to search for adjacent shapes and street segments (default value: 100.0).
- Adjacency Search Angle—Absolute angle to search for adjacent shapes and street segments (default value: 15.0).
- Edge Indent—Relative indent from both corners of an edge, in percentage of edge length (default value: 10.0).
- Edge Tolerance—Absolute offset toward inside of a shape for increased tolerance to correctly handle overlapping shapes (default value: 0.1).

### Computation of edge orientation

First, the front edge (or the front edges if **Allow Multiple Front Edges** is selected) is determined:

- The algorithm iterates over all edges and selects those facing a street.
- Of those facing a street, the ones with the lowest street category are selected.
- If only one front edge is allowed, and multiple edges are facing the streets of the same category, the longest or the shortest edge is selected, depending on the precedence setting for front edges (Precedence for Front Edges).
- Once the front edges are determined, the algorithm computes the rear edges. Depending on the precedence setting for rear edges, either the ones that are farthest from front edges or the ones that are most parallel to front edges are chosen (Precedence for Rear Edges).
- The remaining edges are assigned as side edges (or inner edges if they are part of a hole in a shape).

## Creation of edge attributes

As indicated above, the algorithm depends on adjacent streets' category and width attributes. It is possible to configure which street attributes are taken using the following algorithm parameters:

- **Street Category Attribute**—The name of the street network attribute to be used to obtain the street category from a street edge. The object attribute needs to be a `string` type. If left empty, the built-in Major or Minor attribute is used.
- **Comma-separated List of Street Categories**—This allows the user to define an ordered list of street categories that are used for sorting and determining the front edges. For example, assume that your street network contains an attribute `street_category` with the values `freeway`, `highway`, and `backroad`. Then you would set the street category attribute to `category` and the street category list to `freeway, highway, backroad` to ensure the precedence is properly used.
- **Street Width Attribute**—The name of the street network attribute to be used to obtain the street width from a street edge. The object attribute must be a `string` type. If left empty, the built-in street width is used.

**Compute Edge Attributes**  
Compute and assign edge attributes from nearest objects for all selected shapes.

Preset:  
<Previous Compute Edge Attributes Settings>

Adjacency Search Range (Absolute)	100.0
Adjacency Search Angle (Absolute)	15.0
Edge Indent (Percentage of Edge Length)	10.0
Edge Tolerance (Absolute)	0.1
Allow Multiple Front Edges	<input checked="" type="checkbox"/>
Precedence for Front Edges	Long Edge
Precedence for Rear Edges	Furthest from Front Edge
Street Category Attribute	street_category
Comma-separated List of Street Categories	freeway, highway, backroad
Street Width Attribute	street_width

Undo Close Apply


*Set the algorithm parameters.*

# Set first edge


The **Set First Edge** tool sets the first edge of a face to the currently selected edge. This step is often needed to orient a face's zero edge toward a street (for example, for placing a building's front correctly). If a face is selected, the highlighted gradient line indicates the first edge (with gradient from vertex 0 to vertex 1).

Click **Shapes > Set First Edge** in the main menu.

The first edge of a shape is highlighted in orange. The first edge of the shape is always defined as Index 0.




Inspector ×




Edge (Index 0)

x	
328128.0...	-0.080
328094.2...	-0.080

After the tool is applied, the first edge is now the bottom horizontal edge.



Inspector ×



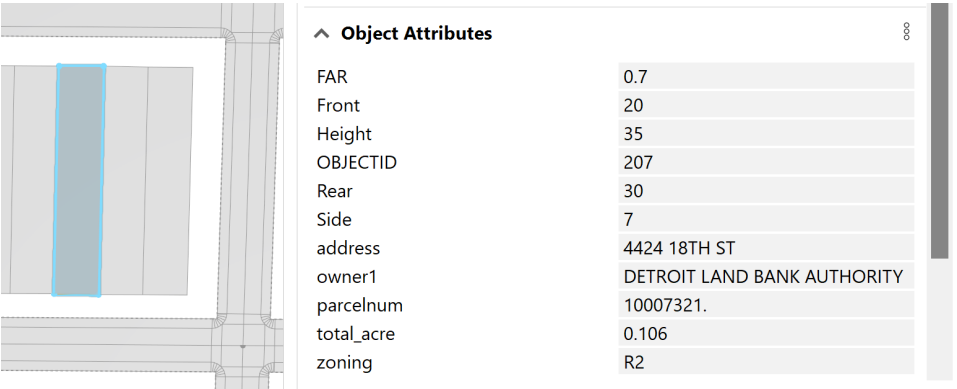
Edge (Index 0)

x	
328132.5...	-0.080
328128.0...	-0.080

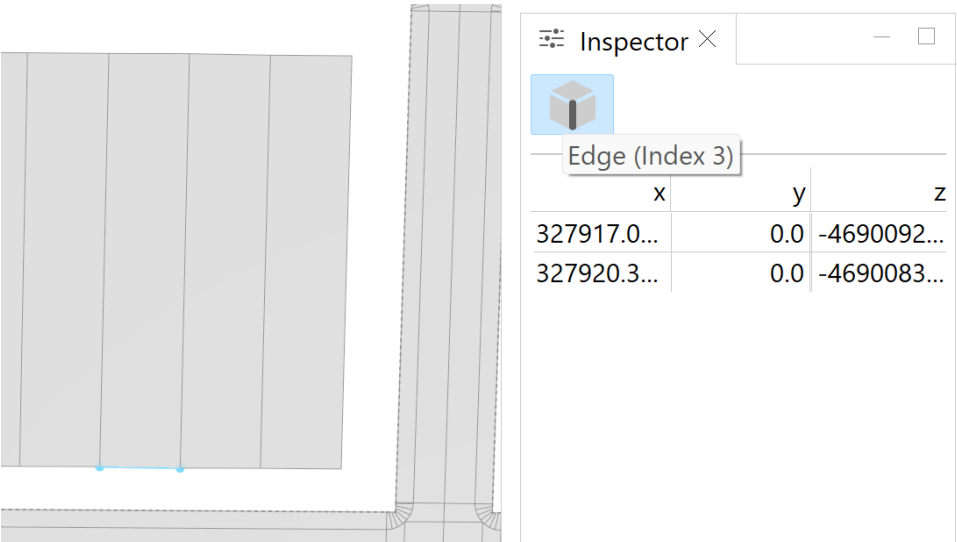
# Set street edges

The **Set Street Edges** tool marks the selected edges as street edges. More specifically, it sets the street width object attribute array to 1 for selected edge indices. When mapped to a CGA rule, the `streetWidth()` attribute can be used to identify edges or faces that are facing a street. (See also the [comp operation](#) in the CGA reference). Click **Shapes > Set Street Edges** in the main menu.

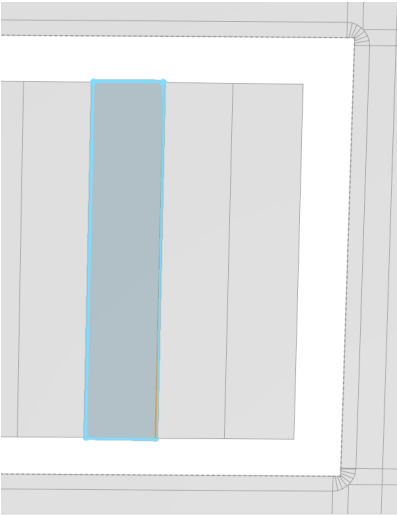
Before the tool is applied, you can see the object attributes of the selected shape in the **Inspector** window. The right vertical edge is the first edge, Index 0, and is highlighted in orange:



Next, the bottom edge is selected to be set as a street edge and displays as Index 3 when hovering over the shape icon in the **Inspector** window:



Finally, the street edge is set and a `streetWidth` object attribute array is created under the **Object Attributes** section. Index 3 is assigned a value of 1, indicating that it is a street edge:



^ Object Attributes

FAR

0.7

Front

20

Height

35

OBJECTID

207

Rear

30

Side

7

address

4424 18TH ST

owner1

DETROIT LAND BANK AUT

parcelnum

10007321.

^ streetWidth[4]

[NaN, NaN, NaN, 1]

^

0

NaN

1


NaN

2

NaN

3

1

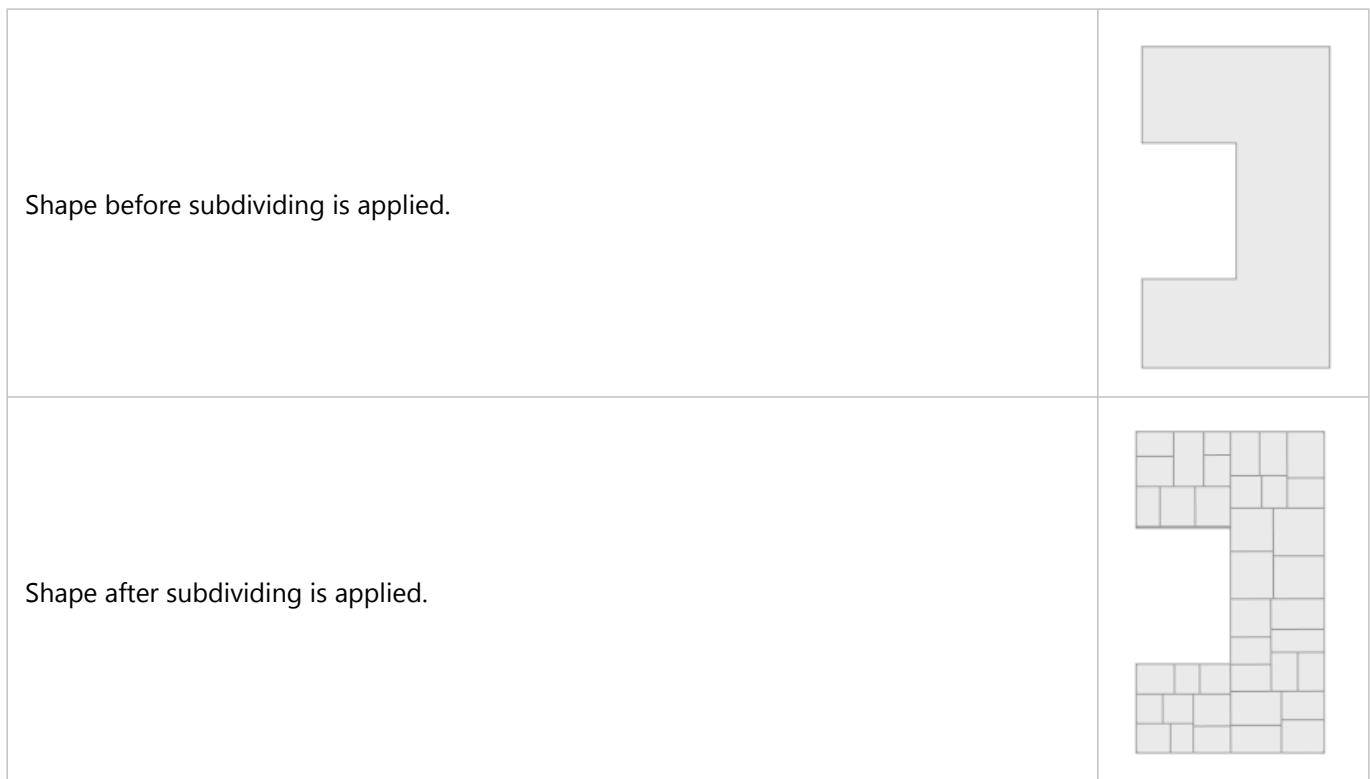
 **Tip:**  
To set multiple edges as street edges, press **Shift** and click when selecting the edges or make a rectangular selection.

# Shape geometry tools

In the **Shapes** main menu, you can edit shapes with the shape geometry tools such as **Reverse Normals**, **Separate Faces**, or **Subtract Shapes**.

## Subdivide

The **Subdivide** tool computes smaller shapes from selected shapes. Click **Shapes** > **Subdivide** in the main menu. A variety of parameters can be used to achieve different subdivision layouts.

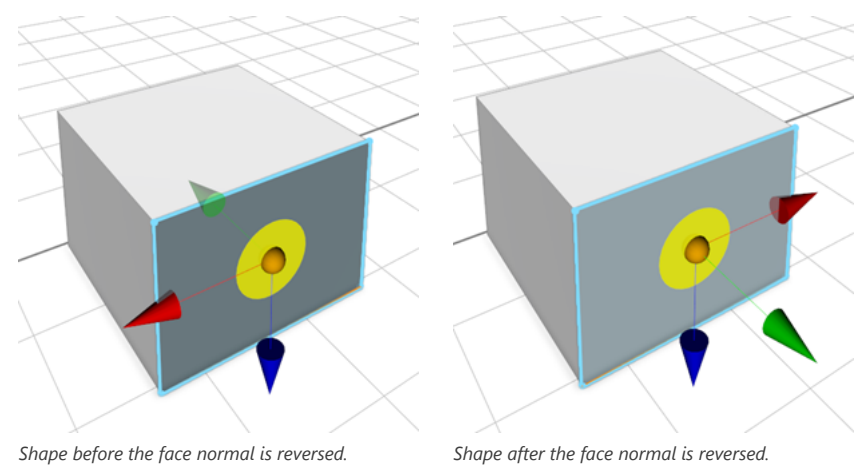


### Tip:

You can also subdivide shapes formed when [creating shapes from graph networks](#) using [Block Parameters](#) in the **Inspector**.

## Reverse Normals

The **Reverse Normals** tool reverses the face normals of the selected shapes. This step is often necessary after importing shapes with reversed orientation. Click **Shapes** > **Reverse Normals** in the main menu.



## Separate Faces

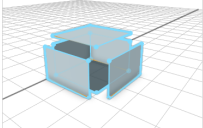
The **Separate Faces** tool allows you to select a shape or multiple shapes and separate individual shapes for every face. Click **Shapes > Separate Faces** in the main menu. All new shapes are created in the existing shape layer.

Before	<div>Shape object with six faces before separating.</div> <div><div>Search for layers, objects or attributes</div><div>New Shapes (1 object, 1 selected) <input type="checkbox"/> <input checked="" type="checkbox"/></div><div>Shape</div></div> <div></div>
After	<div>Six shape objects created after separating the faces and then moved apart.</div> <div><div>Search for layers, objects or attributes</div><div>New Shapes (6 objects, 6 selected) <input type="checkbox"/> <input checked="" type="checkbox"/></div><div>Shape_S_0 Shape_S_1 Shape_S_2 Shape_S_3 Shape_S_4 Shape_S_5</div></div> <div></div>

## Combine Shapes


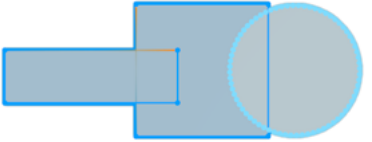

The **Combine Shapes** tool allows you to combine all the faces of the selected shapes into a single shape. Click **Shapes > Combine Shapes** in the main menu. After the shapes are combined, you can still apply tools on the individual shapes, such as the transform tools.

Before	<div>Six shape objects selected from above.</div> <div><div>Search for layers, objects or attributes</div><div>New Shapes (6 objects, 6 selected) <input type="checkbox"/> <input checked="" type="checkbox"/></div><div>Shape_S_0 Shape_S_1 Shape_S_2 Shape_S_3 Shape_S_4 Shape_S_5</div></div> <div></div>
--------	--

After	<p>The six shape objects are now combined into one object.</p> <div><div><div>Search for layers, objects or attributes</div><div>New Shapes (1 object, 1 selected) <input type="checkbox"/> <input checked="" type="checkbox"/></div><div>Shape_3.5</div></div></div>
-------	--

## Union Shapes

The **Union Shapes** tool performs a Boolean union of all selected shapes. Select the shapes to be united and click **Shapes > Union Shapes** in the main menu.


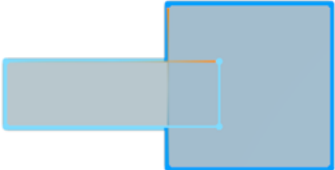


Overlapping shapes.	
Selected overlapping shapes.	
Multiple shapes united into one shape.	

 **Note:**

- The **Union Shapes** tool is different from the **Combine Shapes** tool because it creates one face from the outline of the selected overlapping shapes.
- If the selected shapes are not intersecting or coplanar, the **Combine Shapes** operation is performed.
- This tool works only with planar shapes.

## Subtract Shapes

The **Subtract Shapes** tool performs a Boolean subtraction of the [lead selection](#) shape from all other selected shapes. Click **Shapes > Subtract Shapes** in the main menu. Press **Shift** while selecting a different shape to change the lead selection.

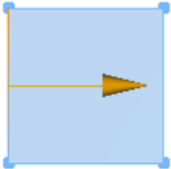

Overlapping shapes.	
Selected shapes with the rectangle as the lead selection.	
Shapes after subtracting the rectangle from the square.	
Shapes moved away from each other.	

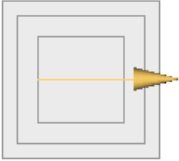


 **Note:**

This tool works only with planar shapes.


## Offset Shapes

The **Offset Shapes** tool creates a shape, rather than a new face, inside the selected shape or multiple shapes. Select a shape or an individual face and click **Shapes > Offset Shapes** in the main menu.

Selected shape to create an inside offset.	
Offset shape created.	

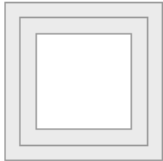
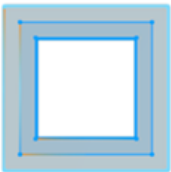

New offset shape created inside previous offset shape.	
Selected inner offset shape.	
Offset shape with a hole after deleting the inner shape.	

To manually enter an offset distance, open the **Tool Options** window  by clicking **Window > Tool Options** in the CityEngine main menu.

 <b>Offset Shapes</b>	
<b>Distance (m)</b>	The offset distance in meters. You can enter a value and press <b>Enter</b> to apply.

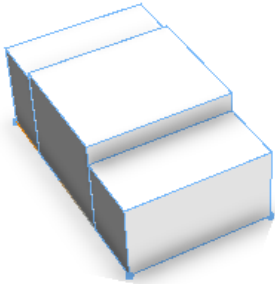
## Remove Holes

The **Remove Holes** tool removes all holes from the selected shapes. Click **Shapes > Remove Holes** in the main menu.

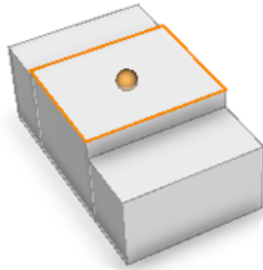
Shape with a hole inside it.	
Shape with the hole selected.	
The hole is removed from the selected shape.	

## Convert Models to Shapes

The **Convert Models to Shapes** tool converts CGA models to shapes for manual editing.



*A CGA model is shown.*



*A CGA model converted to shape.*



### **Note:**

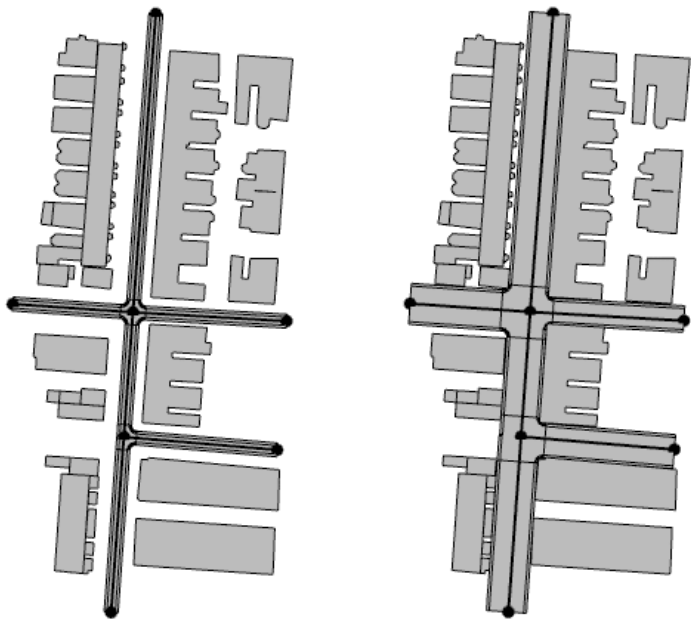
After this conversion, changes in attributes and CGA rules do not affect the shapes.

You can find additional shape tools in the **Shapes** main menu, such as the shape creation or the model generation tools.

# Edit graphs

# Fit widths to shapes

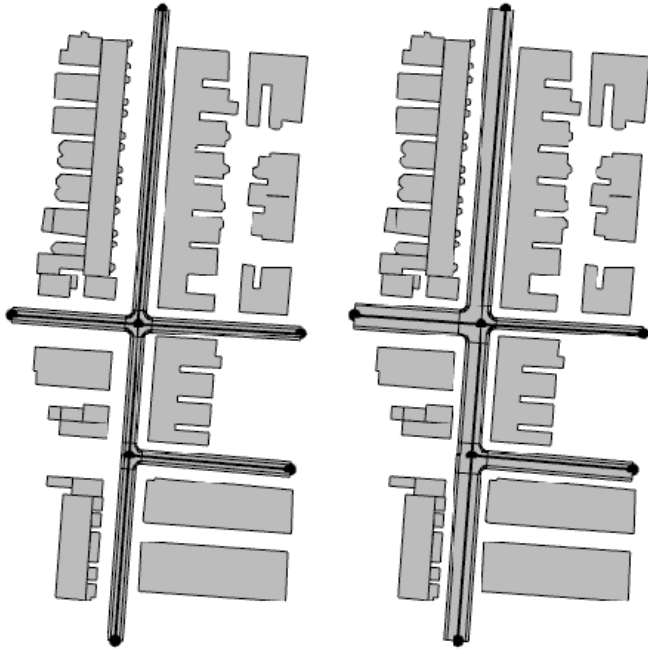
Often, datasets with street center lines do not have width attributes. Using the **Fit Widths to Shapes** tool allows the user to adjust them accordingly. It can be done by clicking **Graph > Fit Widths to Shapes** in the main menu.



Original streets do not line up with the footprint shapes (right). Widths and offsets are adjusted to touch the footprints (left).

## Parameters

<b>Max Street Width</b>	Maximum width the streets will be fitted to. When the calculated width is larger than this parameter, the street is left unchanged.
<b>Min Street Width</b>	Minimum width the streets will be fitted to. When the calculated width is smaller than this parameter, the street is left unchanged.
<b>Sidewalk Scale</b>	Determines whether sidewalk widths are retained (Do Not Change) or scaled proportionally to the street (Scale Proportionally).
<b>Adjust Street Offsets</b>	Determines whether the street offsets should be adjusted to better fit the static shapes.
<b>Additional Margin</b>	This allows you to increase the margin between streets and static shapes.



*(left) Original (right) Additional Margin = 3*

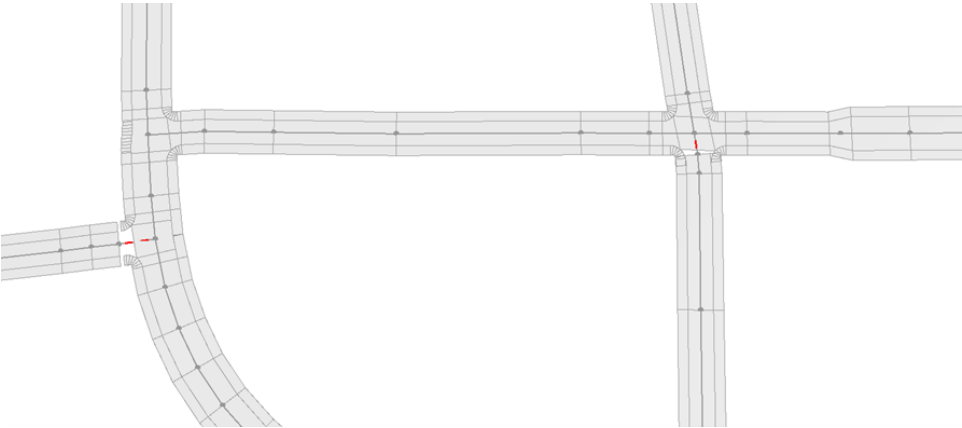
# Simplify graphs

You can use the Simplify Graph tool to remove unnecessary nodes from straight sections and curves of the input streets. It operates on the current street selection or on all the streets when nothing is selected.

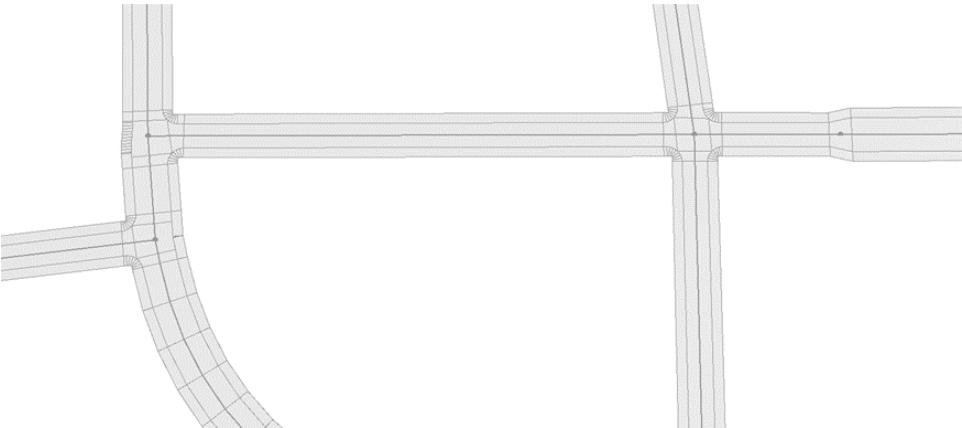
Segments are combined to longer straight or curved segments based on their angle in-between, except in the following cases:

- The node in-between is an intersection (valency > 2).
- Their street width is not equal.
- One of the segments to combine exceeds the set thresholds.
- Their combined length would be larger than the set maximum.

You can click **Graph > Simplify Graph** in the main menu to simplify streets as shown in the following images:



Streets with unnecessary nodes are shown.



Streets simplified with nodes removed are shown.

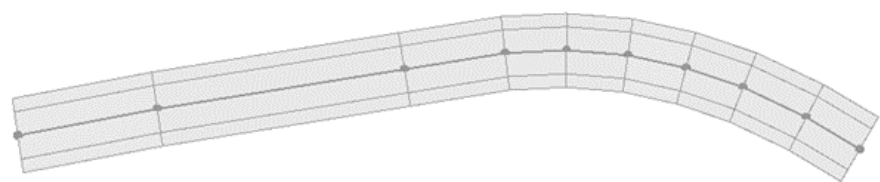
## Simplify settings

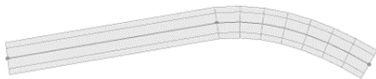
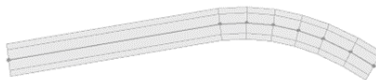
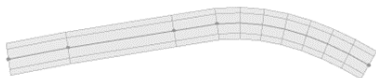
<b>Simplify straight sections</b>	Straight sections are simplified.
-----------------------------------	-----------------------------------


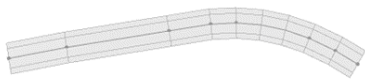

<b>Straight threshold angle</b>	Angle in degrees. Streets with a lower angle than this are combined to straight segments. Meaningful for both simplify straight sections and simplify curves.
<b>Simplify curves</b>	Curves are simplified.
<b>Curve threshold angle</b>	Angle in degrees. Streets with a higher angle than this form boundaries between fitted curves. Only meaningful if the <b>Simplify curves</b> option is checked.
<b>Curve threshold segment length</b>	Maximum length of a single curve input segment. Only meaningful if the <b>Simplify curves</b> option is checked.
<b>Maximum simplified segment length</b>	Maximum length of the output straight or curve segment.

Examples

Below are examples of applying different simplify graph settings to the following input street:



<b>Straight sections and curves</b> 	<div>Simplify straight sections <input checked="" type="checkbox"/></div> <div>Straight threshold angle 5.0</div> <div>Simplify curves <input checked="" type="checkbox"/></div> <div>Curve threshold angle 45.0</div> <div>Curve threshold segment length 25.0</div> <div>Maximum simplified segment length 150.0</div>
<b>Straight sections only</b> 	<div>Simplify straight sections <input checked="" type="checkbox"/></div> <div>Straight threshold angle 5.0</div> <div>Simplify curves <input type="checkbox"/></div> <div>Curve threshold angle 45.0</div> <div>Curve threshold segment length 25.0</div> <div>Maximum simplified segment length 150.0</div>
<b>Curves only</b> 	<div>Simplify straight sections <input type="checkbox"/></div> <div>Straight threshold angle 5.0</div> <div>Simplify curves <input checked="" type="checkbox"/></div> <div>Curve threshold angle 45.0</div> <div>Curve threshold segment length 25.0</div> <div>Maximum simplified segment length 150.0</div>

<div>Curves without limits</div> <div></div> <div>Valid when they are shorter than the set threshold.</div>	<div><div>Simplify straight sections<input type="checkbox"/></div><div>Straight threshold angle0</div><div>Simplify curves<input checked="" type="checkbox"/></div><div>Curve threshold angle180</div><div>Curve threshold segment length1000</div><div>Maximum simplified segment length1000</div></div>
<div>Combined curves</div> <div></div>	<div><div>Simplify straight sections<input type="checkbox"/></div><div>Straight threshold angle0</div><div>Simplify curves<input checked="" type="checkbox"/></div><div>Curve threshold angle180</div><div>Curve threshold segment length9</div><div>Maximum simplified segment length1000</div></div>
<div>Combined segments with length limited</div> <div></div>	<div><div>Simplify straight sections<input checked="" type="checkbox"/></div><div>Straight threshold angle5</div><div>Simplify curves<input checked="" type="checkbox"/></div><div>Curve threshold angle180</div><div>Curve threshold segment length1000</div><div>Maximum simplified segment length60</div></div>

# Analyze graphs

Graph networks can be analyzed by computing global integration, local integration, and in-between centrality. For each street, these values are computed and stored as object attribute `integrationGlobal`, `integrationLocal`, and `inbetweenCentrality`. The values can be visualized or used to approximate street widths.

To open the dialog box, select a set of graph segments or a graph layer and click **Graph > Analyze Graph** in the main menu.

## Settings

Mode	<p>Three modes are available:</p> <ul style="list-style-type: none"><li>• <b>Calculate analysis only (as object attribute)</b>—For each street, the three analysis values, global integration, local integration and inbetween centrality, are computed. These values are stored as object attributes.</li><li>• <b>Visualize analysis (assign rule)</b>—Computes the three analysis values and assigns a visualization rule file to the street shapes. Model generation is automatically triggered. The visualization can be configured by selecting street shapes and using the <b>Inspector</b>.</li><li>• <b>Set street width (based on integration)</b>—Computes the three analysis values and maps the local integration to the range specified by Street Width Min and Street Width Max. After running the tool, select the street layer to see or change the mapping code in the layer attributes in the inspector.</li></ul>
Depth of local integration	The number of 90-degree turns to take into account to compute the local integration value.
Street Width Min	Street width lower bound; it is only used in Set street width mode.
Street Width Max	Street width upper bound; it is only used in Set street width mode.

## Definitions

All shortest paths between all selected street segments are computed. The shortest path cost function is the sum of all angles between the segments along the path.

### Global integration

For each street segment, the shortest paths to all other segments are summed up. Each sum is then divided by the square of the number of segments. Next, each value is inverted. Finally, the values are normalized so that each value is in the range zero to one.

### Local integration

For each street segment, the shortest paths to all other segments which are closer than (Depth of local integration) 90-degree turns are summed up. Each sum is then divided by the square of the number of visited segments. Next, each value is inverted. Finally, the values are normalized so that each value is in the range zero to one.

### *Inbetween centrality*

For each street segment, the number of shortest paths that pass this segment is computed. Then, the values are normalized so that each value is in the range zero to one.

### Example



*The global integration of a typical street network is shown.*


# Edit terrain

# Interactive terrain editing

The **Terrain Edit Brush** , **Terrain Smooth Brush** , and **Terrain Reset Brush**  tools allow you to interactively change the elevation of the terrain in a scene.


## Set terrain height

To edit the height of the terrain, click the **Terrain Edit Brush**  tool or click **Terrains > Terrain Edit Brush** in the main menu.

In the scene, click the terrain to adjust the height. Click and hold to continuously brush the terrain to the new height. You can change brush options, such as height, brush size, or smoothing terrain, in the **Terrain Edit Brush** tool options .


## Terrain Edit Brush tool options

The **Terrain Edit Brush** tool options  include the following:


 <b>Terrain Edit Brush</b>	
<b>Brush Size (m)</b>	The brush size in meters defining the radius in which the terrain height is adjusted.
<b>Elevation Picker</b>	Allows you to pick the terrain height at the location of your mouse by clicking a terrain in the Viewport. Press <b>Y</b> to turn the elevation picker toggle button on and off.
<b>Height (m)</b>	The height in meters in which the terrain is adjusted.
<b>Smooth Borders</b>	<div>Applies a smooth transition method at the brush border between the edited and the original terrain.</div> <ul style="list-style-type: none"><li>• <b>None</b>—No smoothing is applied.</li><li>• <b>Smooth Range</b>—Applies smoothing to the terrain in which the relative height difference at the brush border is interpolated within the specified range defined in meters (see <b>Range</b>).</li><li>• <b>Constant Gradient</b>—Applies a slope with a constant up or down gradient defined in degrees (see <b>Gradient</b>).</li></ul>
<b>Range (m)</b>	<ul style="list-style-type: none"><li>• Radial range in meters in which to apply a smooth transition.</li><li>• Available for the <b>Smooth Range</b> and the <b>Constant Gradient</b> methods.</li></ul>
<b>Gradient (°)</b>	<ul style="list-style-type: none"><li>• The gradient in degrees of the border slope.</li><li>• Available for the <b>Constant Gradient</b> method.</li></ul>
<b>Easing</b>	<ul style="list-style-type: none"><li>• Applies a nonlinear smoothing to avoid sharp edges at the inner and outer fronts of the border region.</li><li>• When the toggle button is off, the smoothing is applied linearly.</li><li>• Available for the <b>Smooth Range</b> method.</li></ul>


## Smooth terrain height

To smooth the height of the terrain, click the **Terrain Smooth Brush**  tool or click **Terrains > Terrain Smooth Brush** in the main menu.

In the scene, click the terrain to smooth the terrain height. Click and hold to continuously smooth the terrain. You can change brush options such as brush size or strength in the **Terrain Smooth Brush** tool options .


### Terrain Smooth Brush tool options

The **Terrain Smooth Brush** tool options  include the following:


 <b>Terrain Smooth Brush</b>	
<b>Brush Size (m)</b>	The brush size in meters defining the radius in which the terrain height is adjusted.
<b>Strength (%)</b>	Determines the amount the terrain surface is smoothed, ranging from no smoothing (0%) to strong smoothing (100%).
<b>Continuous Smoothing</b>	When you turn on the toggle button, the terrain is smoothed continuously as you click and hold. Otherwise, the terrain is smoothed only once.


## Reset the terrain

To reset the terrain to the original height, click the **Terrain Reset Brush** tool  or click **Terrains > Terrain Reset Brush** in the main menu.

In the scene, click the terrain to reset it to its original height. Click and hold to continuously reset the terrain. You can set the brush size in the **Terrain Reset Brush** tool options .

### Terrain Reset Brush tool options

The **Terrain Reset Brush** tool options  include the following:

 <b>Terrain Reset Brush</b>	
<b>Brush Size</b>	The brush size in meters defining the radius within which the terrain is reset.
<b>Smooth Borders</b>	<p>Applies a smooth transition method at the brush border between the edited and the original terrain.</p> <ul style="list-style-type: none"><li>• <b>None</b>—No smoothing is applied.</li><li>• <b>Smooth Range</b>—Applies smoothing to the terrain in which the relative height difference at the brush border is interpolated within the specified range defined in meters (see <b>Range</b>).</li><li>• <b>Constant Gradient</b>—Applies a slope with a constant up or down gradient defined in degrees (see <b>Gradient</b>).</li></ul>
<b>Range (m)</b>	<ul style="list-style-type: none"><li>• Radial range in meters in which to apply a smooth transition.</li><li>• Available for the <b>Smooth Range</b> and the <b>Constant Gradient</b> methods.</li></ul>

<b>Gradient (°)</b>	<ul style="list-style-type: none"><li>• The gradient in degrees of the border slope.</li><li>• Available for the <b>Constant Gradient</b> method.</li></ul>
<b>Easing</b>	<ul style="list-style-type: none"><li>• Applies a nonlinear smoothing to avoid sharp edges at the inner and outer fronts of the border region.</li><li>• When the toggle button is off, the smoothing is applied linearly.</li><li>• Available for the <b>Smooth Range</b> method.</li></ul>

 **Note:**

Reset means restoring the original height defined by the `elevation` attribute of the terrain layer.

## Globally reset terrain

You can reset a single terrain or all terrains simultaneously using the **Reset Terrain** tool. Click **Terrains > Reset Terrain**.

### Reset settings


<b>Terrain</b>	Choose which terrain to reset or select <b>All</b> .
<b>Constraint</b>	<ul style="list-style-type: none"><li>• <b>Everywhere</b>—All terrains are completely reset.</li><li>• <b>Inside selected shapes only</b>—Only the terrain vertices intersecting the currently selected shapes are reset.</li></ul>
<b>Add border</b>	If enabled, a small border region around the shapes is reset, too. Only meaningful if <b>Constraint</b> is set to <b>Inside selected shapes only</b> .

# Align terrain to shapes

The **Align terrain to shapes** tool  aligns terrains to selected shapes.

To align terrain to shapes, click the **Align terrain to shapes** tool  or click **Terrains > Align terrain to shapes** in the main menu.

 **Note:**

The **Align terrain to shapes** tool  aligns one or all terrains to the shapes currently selected.

## Tool settings

The **Align terrain to shapes** tool  has the following options:


<b>Terrain</b>	The terrain that should be aligned or All.
<b>Raise terrain</b>	If enabled, terrain vertices below selected shapes are aligned.
<b>Maximal raise distance</b>	If the distance between the terrain and shape is smaller, terrain vertices below the shape are raised.
<b>Lower terrain</b>	If enabled, terrain vertices above selected shapes are aligned.
<b>Maximal lower distance</b>	If the distance between the terrain and shape is smaller, terrain vertices above the shape are lowered.
<b>Add border</b>	If enabled, a small border region around the shapes is aligned, too.
<b>Write cut/fill volumes to attributes</b>	If enabled, for each selected shape cut and fill volumes in cubic meters are approximately calculated. The values are written into the object attributes (fields cutVolume and fillVolume).
<b>Smooth borders</b>	<p>Applies a smooth transition method at the border between the edited and the original terrain.</p> <ul style="list-style-type: none"><li>• <b>None</b>—No smoothing is applied.</li><li>• <b>Smooth range</b>—Applies smoothing to the terrain in which the relative height difference at the border is interpolated within the specified range defined in meters (see <b>Border range</b>).</li><li>• <b>Constant gradient</b>—Applies a slope with a constant up or down gradient defined in degrees (see <b>Border gradient</b>).</li></ul>
<b>Border range (m)</b>	<ul style="list-style-type: none"><li>• Radial range in meters in which to apply a smooth transition.</li><li>• Available for the <b>Smooth range</b> and the <b>Constant gradient</b> methods.</li></ul>
<b>Border gradient (°)</b>	<ul style="list-style-type: none"><li>• The gradient in degrees of the border slope.</li><li>• Available for the <b>Constant gradient</b> method.</li></ul>

<b>Border easing</b>	<ul style="list-style-type: none"><li>• Applies a nonlinear smoothing to avoid sharp edges at the inner and outer fronts of the border region.</li><li>• When the toggle button is off, the smoothing is applied linearly.</li><li>• Available for the <b>Smooth range</b> method.</li></ul>
----------------------	--

## Elevation delta maps

When aligning terrains, the original heightmap of the terrain is not modified. The elevation data is stored as a separate image file in a subfolder (named after the CityEngine scene) in the project's data folder. These files and folders should not be renamed or deleted. If required, however, you can modify the elevation delta image file with an image processing tool (for example, apply blurring) and store it under the same name.

 **Tip:**

Use the **Align terrain to shapes** tool  in conjunction with the [interactive editing](#) tools.

# CGA Modeling


# CGA modeling

Computer Generated Architecture (CGA) is a programming language that is used to instruct CityEngine how to automatically generate 3D geometry, and particularly architecture.

The following are the basic elements of CGA:

**Rule**—The building instructions are called rules. Each rule corresponds to one instruction or transformation step. When run, each rule takes geometry as input and alters or replaces it to generate an output. This output becomes the input of a following rule. The first rule that is run is the start rule, and it takes an initial shape as input geometry. A rule file (\*.cga) contains the start rule and other rules as well as attributes and functions.

**Initial Shape**—The input geometry is typically a polygon that represents a lot or a building footprint. It can be imported, modeled manually, or automatically generated from street graphs. To connect a start rule to an initial shape, drag a \*.cga file onto the shape in the **Viewport** window.

**Generation**—To begin model creation, click the **Generate** button  on the toolbar, or press **Ctrl+G**. The start rule runs and the resulting 3D geometry (model) is shown in the **Viewport** window.

## Note:

To better understand the basic elements in CGA modeling, see the [Generate models with rules](#) section of Tutorial 1: Essential skills.

## CGA and manual modeling

Procedural techniques are applied in many areas of 3D modeling. These approaches are efficient and time-saving when large numbers of models with slight variations and similar typology are required, while full artistic control over each individual instance is secondary. The creation of city models has many use cases for procedural modeling.

In urban design and planning, procedural modeling offers another use case. By instructing a CGA rule to calculate sizes and dimensions of a model during generation, key performance indicators (KPIs) such as gross floor area or total window area can be calculated on the fly. This information can influence the design process in early stages and therefore lead to more complex design decisions.

Manual modeling, however, is fast when a one-of-a-kind model is needed and only a visual representation is required.

Manual and CGA modeling can be combined: CityEngine offers [shape creation tools](#) to define a rough building hull. CGA modeling can then be used to detail facades and roofs. Furthermore, [local edits](#) provide an intuitive tool to manually alter CGA generation by interactive tools.

# Work with rules

The tutorials and example projects that you can download from the CityEngine main menu (**Help > Download Tutorials and Examples**) as well as **ESRI.lib** contain versatile rules that you can use in your projects as is, or as starting points for your own customized rules. To start working with rules, you can create a simple rule file, apply it to initial shapes, and generate models.

## Create a rule file

1. Select the rules folder of your project in the **Navigator** window.
2. Click **New > CityEngine > CGA Rule File**.  
Alternatively, right-click the rule folder in the **Navigator** window and choose **New > CGA Rule File**.
3. Define the name of the rule file.
4. Click **Finish**.  
A new CGA file is created and opened in the **CGA Editor** window. It is empty except for some header information.

## Metadata

```
/**
 * File:      rule.cga
 * Created:   4 May 2008 23:27:29 GMT
 * Author:    xxxx
 */

version "2019.0"
```

The file starts with autogenerated metadata information for the user. It is marked as comments and is ignored by the compiler. The version tag specifies the CityEngine version for which this rule file was originally written.

## Attributes

```
attr minheight      = 10
attr maxheight      = 300
```

You can start by defining two building parameters: minimum and maximum height. It is good practice to choose descriptive names. Additionally, the keyword **attr** is added to the front of each parameter definition. This way, these values become editable attributes in the **Inspector** window. The values set in the CGA file are default values.

## Start rule

```
@StartRule
Lot --> extrude(rand(minheight,maxheight))
```

CityEngine recognizes the annotation **@StartRule** as the start rule, so it doesn't need to be specified manually when assigning the rule file to an initial shape.

The **Lot** rule extrudes the initial shape to a height that is defined by a randomly selected value between the values of the two attributes you defined before.





**Note:**

- Press **Ctrl+S** and click **File > Save** or right-click in the **CGA Editor** window and select **Save** to save your CGA file. You must save your changes for them to take effect for the model generation.
- To open CGA files, double-click the file in the **Navigator** window or click **File > Open** in the main menu.

Learn more about the [CGA Editor](#) window.

## Assign rules and generate models


1. Create some shapes using the **Polygonal Shape Creation** tool (S) .
2. Select the shapes.
3. Click **Shapes > Assign Rule Files** and select the rule file you just created.
4. Click **Open**.
  - In the **Inspector** window, the **Rule File** and **Start Rule** fields are filled in now.
  - The **maxHeight** and **minHeight** attributes appear in the **Inspector** window.
5. Click the **Generate models** tool  or press **Ctrl+G**.  
In the **Viewport** window, you now see extrusions to different heights.

Select some of the shapes and change one of the attributes in the **Inspector** window. This automatically triggers a regeneration of the selected shapes.



**Note:**

The changes in the **Inspector** window don't affect the values defined in the **.cga** file.

Repeatedly regenerating the rule without changing the attributes always yields to the same building heights. This behavior is intended because although you are using random values, you want the scene to look exactly the same as when closing and reopening the scene. To get new random values, you need to set a new seed on the shapes. You can do this by clicking the **Update Seed** tool  (or pressing **Ctrl+Shift+G**) on the toolbar.

Learn more about CGA in [Tutorial 6: Basic shape grammar](#).

## Models from CGA without shapes

Some CGA rules, such as vegetation or furniture, can generate 3D models without assigning them to shapes. For example, you can customly place trees into a scene, by dragging the **Plant\_Loader.cga** rule from the **ESRI.lib** library directly onto the terrain in the **Viewport**. A small shape (0.1m x 0.1m) is created at the position where you dragged the rule.

After, you can use the **Inspector** window to modify the rule attributes.

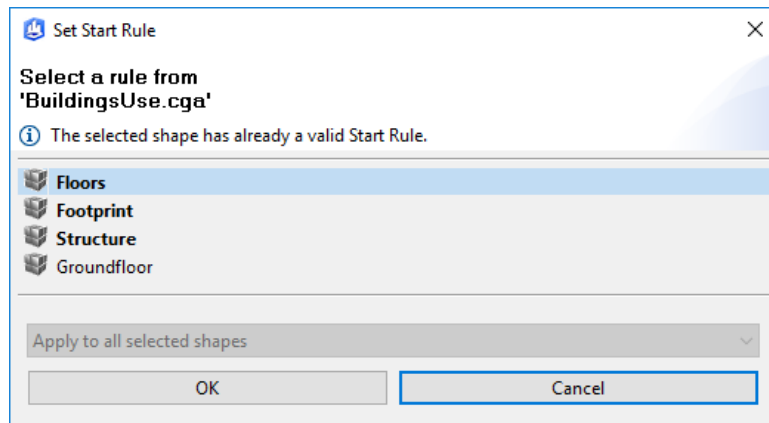
## Set the start rule

In addition to the rule file, a shape requires a valid start rule to trigger model generation. If no valid start rule is found while a rule file is assigned, the **Set Start Rule** dialog box appears.

You can also type the start rule in the **Inspector** window, or set it on the **Set Start Rule** dialog box by clicking **Select** in the **Inspector**.

### Set Start Rule dialog box

The **Set Start Rule** dialog box displays all rules that can be applied as start rules. Rules that are marked as start rules are displayed in bold in the following image (see also [CGA annotations](#)):



*Start rules are selected on the Set Start Rule dialog box.*

Select the rule that you want to assign as start rule. Additionally, you can choose the following:

- **Apply to all selected shapes**—Set the start rule to all selected shapes.
- **Apply only to shapes with invalid Start Rule**—Set the start rule to all shapes with an invalid or empty start rule. This option is only available if applicable.

#### **Note:**

CityEngine attempts to automatically detect and suggest start rules from a rule file. Use the annotation `@StartRule` to explicitly mark a rule as a start rule. See [CGA annotations](#) for more details.

### Default start rules

- Shapes that are generated in CityEngine from a street network (by block subdivision or street shape creation) have their start rule set to a default value during creation (Lot, LotInner, Street, Sidewalk, and so on). You can reset these start rules to their initial value by clicking **Graph** > **Reset Shape Attributes**.
- Shapes that are imported from an `.obj` file (arbitrary geometries) have their start rule set to the obj group name by default.

#### **Note:**

See [Rules](#) to learn more about working with rules in **Inspector**.

# CGA essentials

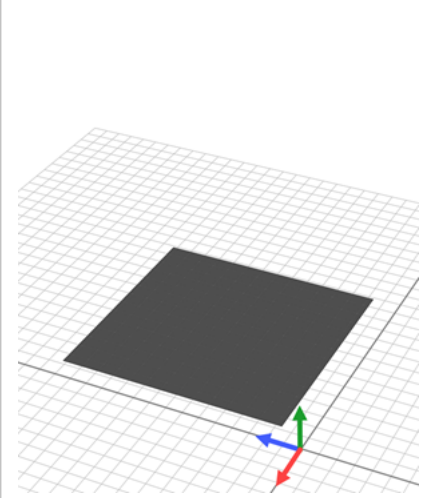
## Coordinate systems

3D coordinate systems are used to describe the position and orientation of objects in space. While CityEngine can work with many types of (georeferenced) coordinate systems, CGA modeling works only with Cartesian coordinate systems. A coordinate system is defined by its origin and the orientation of the three orthogonal axes. While one global coordinate system is sufficient to define each vertex of a geometry, it is more practical to use a hierarchy of coordinate systems that are defined relative to each other.

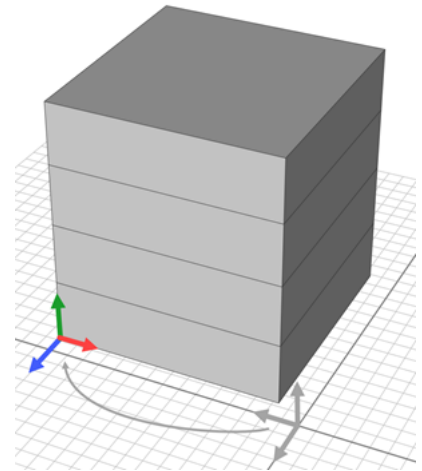
For example, a car model is placed on a street segment. It makes sense to define the position of the wheels relative to the car and the position of the car relative to the street. The origin and the orientation of the car system are defined with coordinates of the street system. This way, the two coordinate systems are relative to each other and the coordinates of one system can be translated into coordinates of the other.

Several coordinate systems are involved when working with shapes. All transformations described above operate in the system defined by the current shape's scope, the scope system. There is also a pivot system associated to each shape, and every shape defines an object coordinate system.

Multiple coordinate systems are relative to each other in CGA:

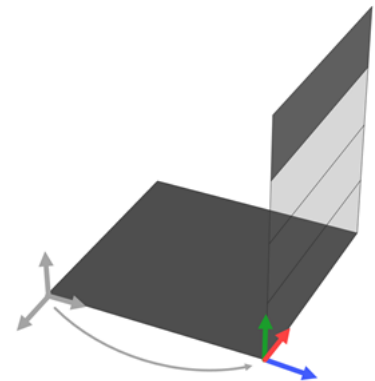
World Coordinate System	
<ul style="list-style-type: none"><li>• Defined per scene.</li><li>• Global—there can be only one per scene.</li><li>• Can be georeferenced, but doesn't have to be.</li><li>• The position of initial shapes is defined in world coordinates.</li></ul>	
Object Coordinate System	

- Local coordinate system is defined for each initial shape.
- The origin is placed at the first point of the initial shape's first edge.
- The axes are oriented so that the x-axis is directed along the first edge, the y-axis is directed along the first face's normal, and the z-axis is perpendicular to the first two.



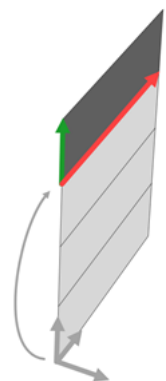
### Pivot Coordinate System

- The pivot system is described in object coordinates.
- In many cases, the pivot origin and orientation are identical with the object coordinates. Still, it becomes a factor, for example, with [component splits](#).



### Scope Coordinate System

- The scope is like a bounding box that is placed and oriented in the pivot system.
- In addition to its characteristics of a coordinate system, the scope has also a size. The size is defined by a width, height, and depth.
- Using [transformations](#), you can define the position, orientation, and size of the scope.

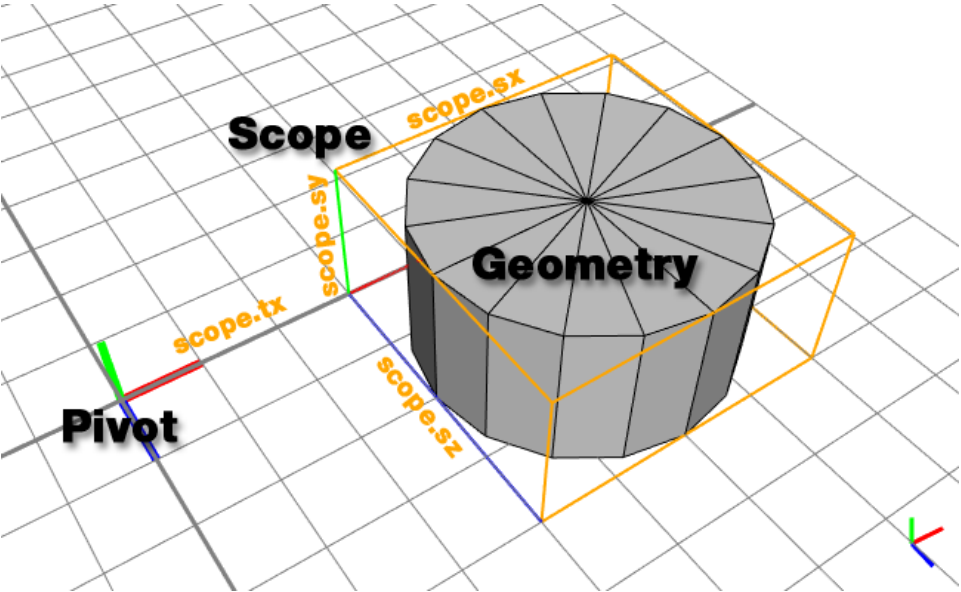


 **Note:**

- When developing CGA rules, you can visualize the coordinate systems and shapes of a model using the [Model Hierarchy](#) window.
- You can query the [pivot](#) and [scope attribute](#) attributes.

CGA shapes

CGA shapes are the central component of the CGA shape grammar. A shape consists of a geometry in an oriented bounding box, the scope. The scope is placed relative to the pivot.



The translation and size of the scope applied in the pivot coordinate system defines the position, size, and orientation of the geometry.

 **Note:**

Initial shapes serve as a starting CGA shape for CGA rules. See [Shapes](#).

A shape has the following components:

<b>Shape Symbol</b>	The name of the shape. Used to find the matching rule that is used to generate the successive shapes.
<b>Pivot</b>	<p>The pivot describes the shape's coordinate system and is defined by the following:</p> <ul style="list-style-type: none"><li>• A position vector <math>p</math> (<code>pivot.px</code>, <code>pivot.py</code>, <code>pivot.pz</code>)</li><li>• An orientation vector <math>o</math> (<code>pivot.ox</code>, <code>pivot.oy</code>, and <code>pivot.oz</code>).</li></ul> <p>The pivot is given in object coordinates, relative to the initial shape's origin; see <a href="#">Coordinate systems</a>.</p>

<b>Scope</b>	<p>The scope represents the oriented bounding box for the shape in space relative to the pivot and is defined by three vectors:</p> <ul style="list-style-type: none"><li>• Translation vector <code>t</code> (<code>scope.tx</code>, <code>scope.ty</code>, and <code>scope.tz</code>)</li><li>• Rotation vector <code>r</code> (<code>scope.rx</code>, <code>scope.ry</code>, and <code>scope.rz</code>)</li><li>• Size vector <code>s</code> (<code>scope.sx</code>, <code>scope.sy</code>, and <code>scope.sz</code>)</li></ul>
<b>Geometry</b>	<p>The geometry contains the information about the corners, edges, and faces that make up the form of the shape. The geometry can be any type of polygonal mesh. In addition, information about color, material, and textures (shader attributes) is stored in the geometry.</p>
<b>Parameters</b>	<p>Each shape can have an associated parameter list. The ordered parameter list is implicitly defined in the rule that creates the shape. Three parameter types are supported:</p> <ul style="list-style-type: none"><li>• Boolean</li><li>• Numeric (internally represented with double-precision float)</li><li>• String</li></ul>

### Rule application

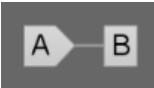
The basic purpose of a rule is to replace a shape with a specific shape symbol with a number of new shapes, for example:

PredecessorShape --> Successor

The following is a simple example:

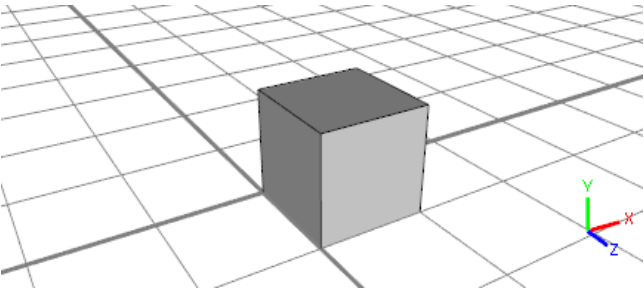
A --> B

On application on a specific shape with symbol A, the rule above creates a copy of the shape and sets its shape symbol to B. The A shape is now considered completed and not processed further. If there is no rule matching symbol B, the generation process is finished. The resulting structure is called the shape tree and looks like the following example:



Shape tree

In the example shape tree, A is the root shape and B is a leaf shape. Leaves are important because the sum of all leaves represents the generated model. Inner nodes are not visible in the final model.



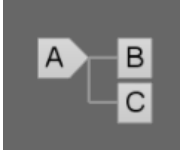
Generated model

In this simple example, shape A's geometry is assumed, and scope and pivot are set up so that the shape represents a unit cube in the origin. Because B is a copy of A, B looks exactly the same (see the image above).

A rule can have more complex successors, for example, the right side of the rule can consist of multiple shape symbols and shape operations:

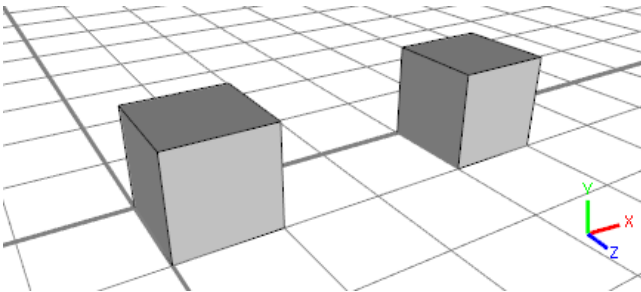
$A \rightarrow B \ t(3, 0, 0) \ C$

This successor is now run from the left to the right. Again, B is an identical copy of A. The current shape is then translated by 3 units in x-direction (in other words, `scope.t` is manipulated) and a new shape C is created. The shape tree now has two leaves:



*Shape C created in tree*

The two leaves B and C make up the final 3D model:

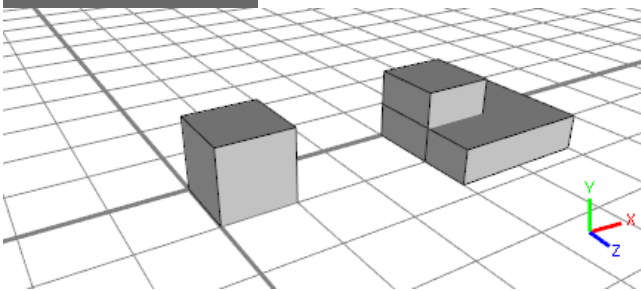
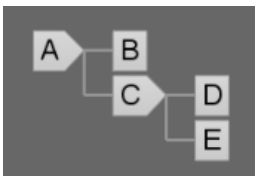


*Leaves B and C make up final model*

Add this rule for C:

$C \rightarrow D \ s(2, 0.5, 1.75) \ E$

The generation process adds two children, D and E, to shape C. Shape D is an exact copy of shape C, but shape E will have a different-sized scope (because of the `s()` shape operation). The shape tree and the associated model look like this:



*Shape tree and model with children D and E added*

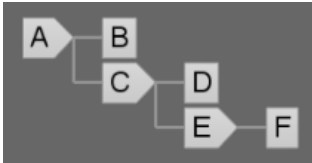
**Note:**

Now, the leaves (B, D, E) are not on the same level (in other words, they have different distances to the root shape), but they are all part of the model.

Another shape operation is the insert operation `i()`:

```
E --> i("cylinder.obj") F
```

After starting the generation again, shape E is no longer a leaf but now has a child shape, F.

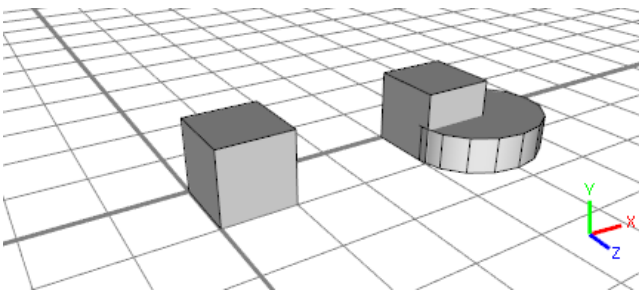


Shape E with child shape F

The geometry of shape F is no longer a cube but is replaced with the mesh read from file "cylinder.obj".

**Note:**

The size (in other words, the `scope.s` vector) of shape F is still the same as that of shape E.



Shape F with same size as shape E

## Terminal shapes

In the E rule above, F is a so-called terminal shape: because no F rule is defined, the generation is stopped at this point. However, the CGA editor will issue an undefined rule warning. This can be suppressed by adding a period after F, which marks F as a terminal shape:

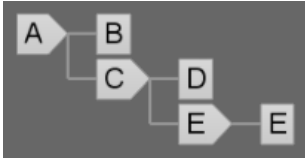
```
E --> i("cylinder.obj") F.
```

## Anonymous leaf shapes

For convenience, rules such as the E rule above can be truncated:

```
E --> i("cylinder.obj")
```

In this case (in other words, when E has no children), the rule interpreter silently inserts an anonymous leaf with the same name as the rule itself. The shape tree after applying the E rule above looks like this:



*The shape tree after applying the E rule*

# CGA Features

# CGA features

The following table of CGA features includes information about typical constructs, CGA syntax, and helpful features. CGA operations are described in [Essential shape operations](#).

<a href="#">Rule with parameters</a>	Pass multiple strings, numbers, or Boolean values from a rule to its successors.
<a href="#">Conditional rule</a>	Call different successor rules based on conditions.
<a href="#">Stochastic rule</a>	Call different successor rules at random. The likelihood of each successor can be controlled by percentages.
<a href="#">CGA attributes</a>	Global variables used to store values in multiple rules. You can modify them in the rule file or the <b>Inspector</b> window.
<a href="#">CGA styles</a>	A style is a specific combination of values for a set of attributes. You can switch between styles in the <b>Inspector</b> window.
<a href="#">CGA functions</a>	Encapsulate code so that it can be used by multiple rules. This helps to avoid duplicated code.
<a href="#">Local variables</a>	Store intermediate values in local variables.
<a href="#">Rule file import</a>	Import rule files to get access to all rules, functions, and attributes.
<a href="#">Comments</a>	Make your code easier to understand for your peer programmers (or yourself in the future).

# Rule with parameters

Rules can be parameterized; in other words, a signature with parameters can be defined, and the matching signature is chosen during generation.

## **Note:**

No explicit parameter type is required. The CGA compiler automatically finds the type of the parameter. There are three types in the CGA grammar: float, Boolean, and string. The float type is used for all numbers (also integers). For each type there is also an array variant: float array, Boolean array, and string array.

### Example 1

```
Lot                --> s('0.8','1','0.8)
                    center(xz)
                    Footprint(20)

Footprint(height) --> extrude(height*0.8)
```

When the Lot rule is run, a new shape is generated with a shape symbol of Footprint and a float parameter of 20. Also, the height parameter has a value of 20.

## **Note:**

You can build arbitrary mathematical expressions with float parameters.

### Example 2

```
Lot                --> s('0.8','1','0.8)
                    center(xz)
                    Footprint(20,geometry.area)

Footprint(height, area) --> t(0,0,1)
                           extrude(height)
                           Envelope(area)

Envelope(area)      --> split(y) { ~4 : Floor(area) }*
```

The Footprint rule takes two parameters, and the Envelope and Floor rules take one parameter.

## **Note:**

Notice how area is passed from rule to rule.

### Example 3

```
Lot                --> Footprint("just an example")

Footprint(height,area) --> t(0,0,1)
                           extrude(height)
```

```
Envelope(area)
Footprint(text)    --> print(text)
```

Rule overloading is shown in example 3. There are two `Footprint` rules, one with two float parameters and one with one string parameter. The compiler automatically ensures that only the matching one is used during shape creation (in other words, when the `Lot` rule above is run, a `Footprint` shape with a string parameter is created).

**Note:**

If no matching rule exists, a new leaf is generated.

# Conditional rule

It is possible to generate different successors for different conditions. Conditions can be expressed by rule parameters, shape attributes, or geometry functions.

```
PredecessorShape -->
    case condition1 : Successor1
    case condition2 : Successor2
    ...
    else: SuccessorN
```

## Example 1

```
Footprint(type) -->
    case type == "residential" : extrude(10)
    case geometry.area/2 < 200 : extrude(30)
    else : NIL
```

In this example, the Footprint rule takes one parameter, `type`, of type string. If the string is equal to "residential", the first successor is taken (in other words, the current shape is extruded by 10 units).

- If the string is not equal to "residential", and the area of the current shape's geometry is smaller than 400, the second successor is taken (in other words, the current shape is extruded by 30 units).
- If none of the two conditions above is true, the third successor is taken and a NIL shape is generated (NIL is a special shape symbol and means that no shape is generated).
- Conditions can arbitrarily be combined with operators `&&` and `||` (Boolean and/or operations), and mathematical expressions can be used.
- It is also possible to nest conditions. There is no limit on the nesting level.

## Example 2

```
Footprint(type) -->
    case type == "residential" || type == "park":
        case geometry.area/2 < 200 && geometry.area > 10 : extrude(10)
        else: extrude(15)
    case type == "industrial" : extrude(100)
    else : NIL
```

Example 2 demonstrates nested conditions and Boolean operations.

### Note:

The `case` and `else` statements must build a consecutive block and cannot be interrupted with successors (like a block, and they are different from `if` statements in well-known programming languages).

The `case` and the `else` statements must build a consecutive block and cannot be interrupted with successors. They are similar to `switch-case` blocks and different from `if` statements in well-known programming languages.

# Stochastic rule

Similarly to conditional rules, the CGA shape grammar permits stochastic rules, in other words, creating variation using randomness.

```
PredecessorShape -->
    percentage% : Successor1
    percentage% : Successor2
    ...
    else : SuccessorN
```

The sum of all percentages must not be greater than 100.

## Example 1

```
Lot -->
    30% : Lot("residential")
    20% : Lot("retail")
    else : Lot("industrial")
```

In this example, there is a 30 percent chance that the first successor is chosen and a new Lot shape with parameter "residential" is generated, a 20 percent chance for the second successor, and a 50 percent chance for the last successor to be chosen.

All random numbers, and also the choice of the percentages above, depend on the current shape's seed (the seedian shape attribute).

## Example 2

Again, condition blocks can be nested:

```
Lot -->
    30% :
        50% : Lot("residential")
        else : NIL
    20% : Lot("retail")
    else : Lot("industrial")
```



### Note:

A condition block must always be finished with an `else:` statement and percentages and successors must not be mixed up.

# CGA attributes

Attributes are a set of global variables defined in the rule file with the following properties:

- They are global relative to the scope of the rule file.
- Each attribute is initialized to a specific value in the rule file.
- The attribute values can be changed individually on a per shape basis.

You can do this in the **Rules** pane in the **Inspector** window.

## Note:

In contrast to [functions](#), attributes are evaluated only once at the beginning of the generation process.

## Example

```
attr height = 150
attr landuse = "residential"

Lot --> extrude(height) Envelope(landuse)
```

In this example, two attributes are defined using the `attr` keyword: `height`, which is of type float, and `landuse`, which is a string. The attributes are used in the `Lot` rule.

Attributes can also be conditional or stochastic:

```
attr landuse = 50% : "residential"
               else : "industrial"
```

In this example, there is a 50 percent chance that `landuse` evaluates to `"residential"` and a 50 percent chance that it evaluates to `"industrial"`. For each shape, the conditional or stochastic attributes are evaluated once and stay constant during the generation process.

Similarly, you can use the `rand()` function:

```
attr height = rand(30,50)

Lot --> extrude(height) Envelope

Envelope -->
  case height < 40 : SmallBuilding
  else: LargeBuilding
```

For each shape that has the `Lot` start rule, `height` evaluates to a value between 30 and 50 units. This height is constant and can be used everywhere in the rule file.

 **Note:**

- You can click and edit the value or use the slider to modify the value. See [Map attributes with the Connection Editor](#) for more information about using different sources to control input of rule attributes.
- The display of attributes in the **Inspector** window can be controlled by [CGA annotations](#).
- You can use interactive [handles](#) to edit attributes in the 3D view.

## Change attributes in the Inspector window

You can set the values of functions marked with `attr` individually for each initial shape in the **Inspector** window. For instance, the definition below yields an entry in the **Rule Parameters** section.

```
@Range(min=10, max=40)
attr height = 20
```

## Working with material attributes


As CGA materials consist of several [attributes](#), you can manage CGA materials by using [.cgamat](#) files. These files are typically created with the CGA Material Encoder that is accessible using the model exporters.

```
version "2024.0"

@MaterialFile
attr myMaterial = "/ESRI.lib/assets/Materials/Architectural/Walls/Brick/
Brick_StretcherBond_10x10_Black.cgamat"

Init -->
    primitiveCube()
    setMaterial(readMaterial(myMaterial))
```

A typical workflow for assigning materials to CGA models involves exposing string attributes that represent paths to [.cgamat](#) files. For example, the above rule assigns the path of one of the [materials](#) from `ESRI.lib` to the `myMaterial` attribute. The material file is read by the [readMaterial](#) function which turns it into an array of key and value pairs that can be passed to the [setMaterial](#) operation.

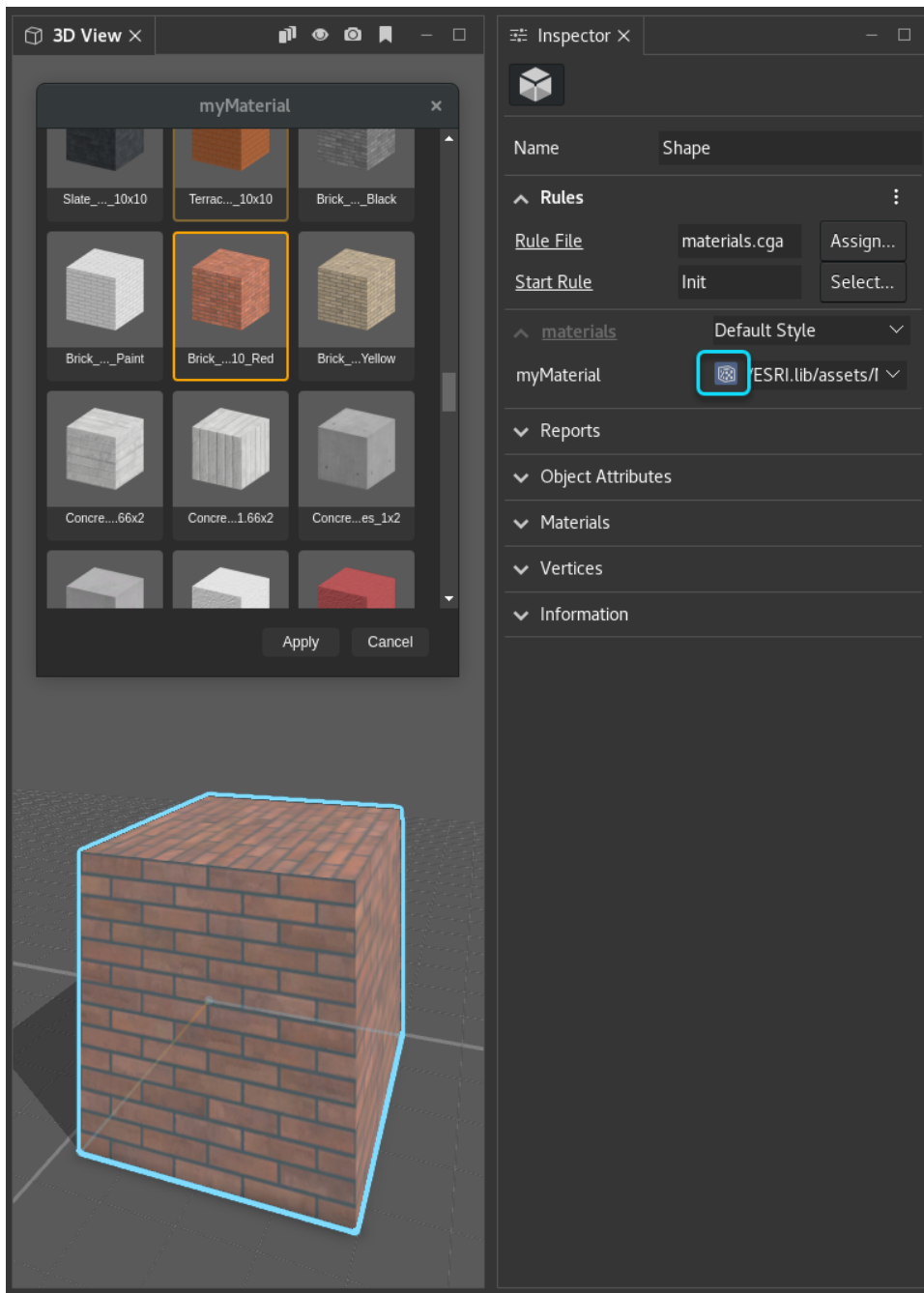
A string attribute can be annotated with the [@MaterialFile](#) annotation that enables the **Material Browser** window for the attribute. To access the **Material Browser** window, click the **Material** button  or click **Browser** from the drop-down menu in the **Inspector** window.

 **Note:**

The material browser is populated with all [materials from ESRI.lib](#).

Also, to preview a [.cgamat](#) file in the **Navigator** window, right-click the file and select **File Preview**.

The **Material Browser** window provides a grid-like gallery with previews of all [.cgamat](#) material files within `ESRI.lib`.



The **Material Browser** window displays the name of the active attribute in the title bar. Each time you select a material, it is applied to the selected models in the CityEngine scene.

#### **Note:**

The **Material Browser** window is non-blocking which means you can modify the camera and inspect the current selected material on the models in the **Viewport** window.

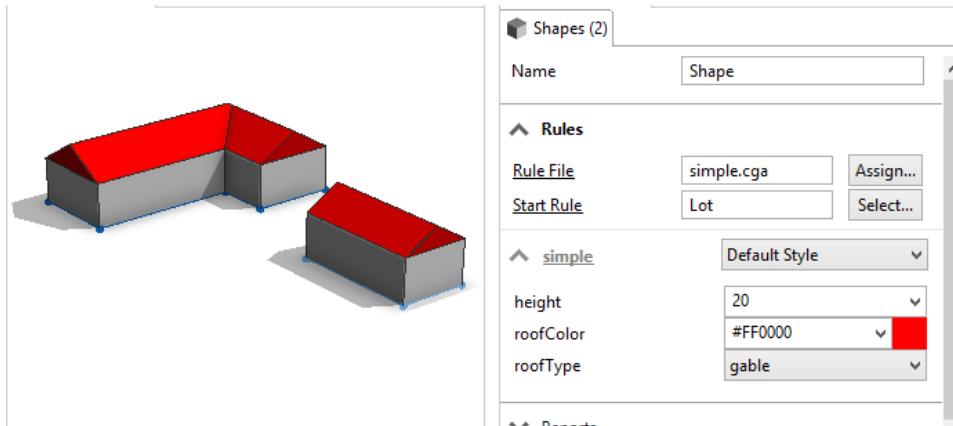
Click **Apply** to apply the material, or click **Cancel** to exit the **Material Browser** window without applying.

Also, changing the scene selection closes the **Material Browser** window without applying the material, similar to the **Cancel** button.

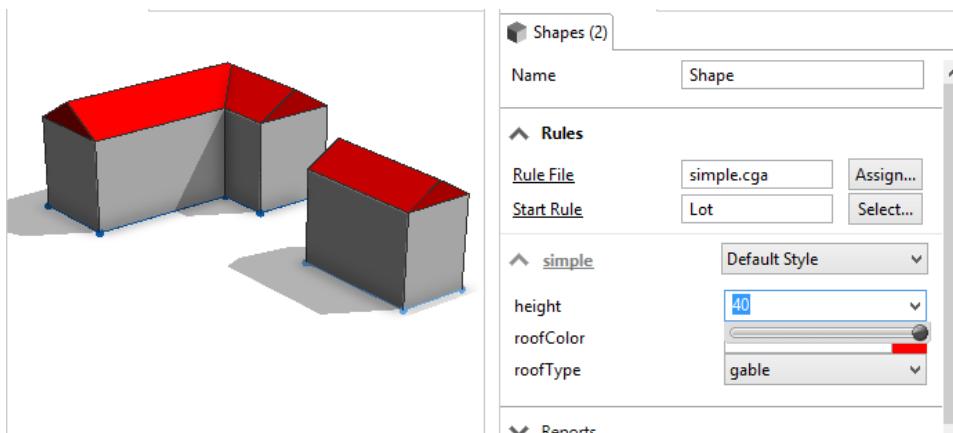
## Assign values to multiple shapes

You can manually assign a value to multiple initial shapes quickly. Complete the following steps to assign attributes to multiple shapes:

1. Select two initial shapes that have a rule file assigned with the `height`, `roofColor`, and `roofType` attributes.



2. Change the height value to 40.  
The height values of both selected shapes are changed.



3. Select only one shape (or model), and change the height value back to the rule default of 20.
4. Select both models.  
The height value is displayed as a question mark (?), indicating that the two selected models do not share the same value for this attribute.

# CGA styles

You can save a set of attribute values using styles. This allows you to save, manage, and retrieve favorite settings for a rule.

## Create a style

To create a style, complete the following steps:

1. Generate the following rule and select the model:

```
@Color
attr col = "#FF0000"
attr height = 1

Lot -->
  color(col)
  extrude(height)
```

2. Change the attribute values in the **Inspector** window.
3. Open the **Default Style** drop-down menu and click **Add new Style**.  
The **Create and apply new style** dialog box appears with the following options:

Style name	Provide a unique name for the style. An existing style with the same name will be overwritten.
Based on	Specify the reference style on which the new style is based. This can be either the default style or the current style (if one is applied).
Description	Optionally, provide a description for the style.

4. Click **OK**.
  - A new style is created and applied to the selected shape.
  - The new style is added to the .cga file.

## Use the Style Manager

To apply a CGA style to shapes, do the following:

1. Select a shape with a rule file assigned.
2. Open the **Style** drop-down menu on the **Rule** tab of the **Inspector** window.
3. Select **Preview & select styles**.  
The **Style Manager** window displays all available styles. A rule file that has no styles defined shows only one style (the default style).
4. Click the style you want to assign to the selected shapes.
5. Click **OK** (or double-click the style).

Alternatively, click the red x to delete a style.

## Toolbar options

The following options are available on the toolbar:

- Tiles view or Scroll view—Display CGA styles as tiles or a list.
- **Without user attributes** or **With user attributes**—Optionally, preserve user-set attributes on the shapes, overwriting the default style attributes.

## Styles in CGA rule file

When a new style is created using the create style wizard, the active CGA rule file is modified, and a new style section is created at the bottom of the rule file. Depending on the creation options, a different set of attributes is added to the new style.

- A new style is created based on the default style—All user-set attributes are added as attributes to the new style, with their user value set as initial value (Default Rule value).

```
style MyStyle
attr col = "#006600"
attr height = 2
```

- A new style is created based on the current style—The new style extends the current one. Only attributes that are different from the base style are added as attributes to the new style; attributes from the base style are inherited.

```
style MyStyle1 extends MyStyle
attr col = "#99FF99"
```

### Note:

- When a new style is created based on an existing style, the new style extends the parent style.
- Instead of using the wizard, you can also add and modify styles in the **CGA Editor** window. For more information, see [styles in the CGA reference](#).

## style keyword

```
# -- facade.cga
...
attr Window_Width = 1.2
attr Door_Height = 2.5

...
// all your Facade CGA rules
...

style Facade_Wide
attr Window_Width = 2.2
attr Door_Height = 2.8
```

By adding a new style using the style keyword and a style name, a new namespace is defined. All definitions below the style keyword are valid for this style only. In the example above, the `attr Window_Width` value, which was set to 1.2 in the default style, is overwritten to 2.2 in the `Facade_Wide` style.

### extends keyword

```
# -- facade.cga
...
attr Window_Width = 1.2
attr Door_Height = 2.5
...
// all your Facade CGA rules
style Facade_Wide
attr Window_Width = 2.2
attr Door_Height = 2.8

style Facade_Wide_2 extends Facade_Wide
attr Window_Width = 2.4

style Facade_Wide_3
attr Window_Width = 2.4
```

You can extend a style from an existing style. An extended style will inherit all definitions from its parent style. In the example above, the following are true:

- Style `Facade_Wide_2` inherits `Door_Height` from its parent style `Facade_Wide` (resulting in a value of 2.8).
- Style `Facade_Wide_3`, which does not extend a parent style, will inherit its value for `Door_Height` from the default style (resulting in a value of 2.5).

#### **Note:**

All styles, whether extended or not, implicitly extend the default style.

## CGA functions

You can use functions to encapsulate evaluations that are used several times in the rules. Unlike rules, functions are typed (in other words, they return a value) and do not change the current shape. Functions can be parameterized, conditional, and stochastic.

### Example

```
getHeight(area) =  
  case area > 1000 : 300  
  case area > 500 :  
    20% : 200  
    50% : 150  
    else : 100  
  else : rand(20,50)
```

The `getHeight` function takes one float parameter (`area`) and returns a height depending on the parameter. If `area` is larger than 1000, 300 is returned. If `area` is larger than 500 (but smaller than, or equal to, 1000), the return value is either 200, 150, or 100 with probabilities 0.2, 0.5, and 0.3. If `area` is smaller than or equal to 500, a random value between 20 and 50 is returned. A rule that uses `getHeight` may look like the following:

```
Lot --> extrude(getHeight(geometry.area)) Envelope
```

**Note:**

In contrast to attributes, functions are evaluated in every call. This means that a function such as the following makes sense only for dedicated purposes, because it returns a different value every time it is used:

```
height = rand(30, 50)
```

### Constant functions

You can make functions constant with the `const` keyword. These functions behave the same as attributes; the only difference is that `const` functions are internal to the rule file and cannot be mapped in the [Inspector](#) window.

# Local variables

Local variables efficiently store values of any type. Within the scope of a rule or a function, you can use them many times via an identifier in the same way as parameters. Local variables eliminate the need to pass intermediate results as a parameter and help you to better structure your code.

## Example 1

In the following example, the Lot rule that was introduced in the previous section, [CGA functions](#), is extended:

```
getHeight(area) =
  case area > 1000 : 300
  case area > 500 :
    20% : 200
    50% : 150
    else : 100
  else : rand(20,50)

Lot -->
  Lot(getHeight(geometry.area))

Lot(height) -->
  extrude(height)
  comp(f) { top : roofHip(case height < 35 : 45 else : 10) Roof
           | all = Envelope }
```

A roof is constructed on the top face of the extruded geometry (see [Component split](#) for more information about the comp operation). If the height value is less than 35, the roof angle is set to 45 degrees; otherwise, it is set to 10 degrees. You want to call the function `getHeight()` only once and use the evaluated value many times. This is achieved by passing the computed height value as a parameter to an overloaded Lot rule (see [Rule with parameters](#)).

With local variables, you can write this using only one rule, Lot:

```
Lot with ( height := getHeight(geometry.area) ) -->
  extrude(height)
  comp(f) { top : roofHip(case height < 30 : 45 else : 10) Roof
           | all = Envelope }
```

The local variable `height` is evaluated and stored as soon as the rule is invoked. It is not reevaluated when it is referenced within the definition of the rule.

## Example 2

In addition to rules, you can also define local variables for functions, const functions, and attributes. In the following example, the `getHeight()` function has no parameter `area`. Instead, the area is computed and stored in a local variable as soon as the function is called.

```
getHeight with ( area := geometry.area ) =
  case area > 1000 : 300
  case area > 500 :
    20% : 200
    50% : 150
```

```

else : 100
else : rand(20,50)

```

## Example 3

You can define a sequence of local variables. An expression may reference a local variable that was defined previously in the list. This makes local variables a tool to split complex calculations into a few smaller steps.

```

getHeight with (
  area          := geometry.area
  minHeight     := sqrt(area) * 0.2
  heightDowntown := 80%: rand(15,35) else: rand(5,15)
  heightHighrise := rand(50,150)
  height        := 10%: heightHighrise else: heightDowntown
  finalHeight   := max(minHeight, height)
) = finalHeight

```

## Rule file import

Importing a rule file makes all rules, attributes, and functions of the imported rule file available under the specified prefix. For more information, see [CGA reference](#).

The following rule files both define a Facade rule. Each textures a building facade in a different way:

```
# -- facade1.cga

actualFloorHeight = scope.sy/rint(scope.sy/4)
actualTileWidth   = scope.sx/rint(scope.sx/4)

Facade --> setupProjection(0, scope.xy, 2, 2)
          projectUV(0)
          texture("facade1.jpg")
```

```
# -- facade2.cga

actualFloorHeight = scope.sy/rint(scope.sy/6)
actualTileWidth   = scope.sx/rint(scope.sx/4)

Facade --> setupProjection(0, scope.xy, 3, 2)
          projectUV(0)
          texture("facade2.jpg")
```

The next rule file contains a Lot rule that creates a building mass and splits it into its side faces and a roof face:

```
# -- structure.cga

// the attribute height will be overridden by the
// attribute height from "main.cga" if this rule
// file is included. Thus if this rule file is
// used standalone, the buildings will be of height
// 100, if this rule file is included by "main.cga",
// the buildings will be of height 200.

attr height = 5

Lot --> extrude(height)
      comp(f) { side : Facade | top : Roof. }
```

Finally, the main rule file with the Lot start rule imports both facade rules and the structure rule.

- The Lot rule calls the Lot rule in `structure.cga` using the `st.` namespace.
- The height attribute in `structure.cga` is overwritten by the height attribute in `main.cga`, so a building of height 10 is created.

### Note:

If `structure.cga` is used as a stand-alone rule file, a building of height 5 will be created.

- The Facade rule for `structure.cga` is defined in `main.cga`. It calls the Facade rule of either `facade1.cga` or `facade2.cga`.

```
# -- main.cga

// define an attribute "height", potentially
// overriding the same attribute in imported
// rule files.

import f1 : "facade1.cga"
import f2 : "facade2.cga"
import st : "structure.cga"

attr height = 10

Lot --> st.Lot

st.Facade --> 50% : f1.Facade else : f2.Facade
```

# Code comments

You can add comments to CGA source code in one of the following ways. You can use line comments with the `//` or `#` characters:

```
// a comment  
# another comment
```

You can use block comments with `/* ..... */`:

```
/* block comments  
   can be used to write  
   multi-line comments  
*/
```

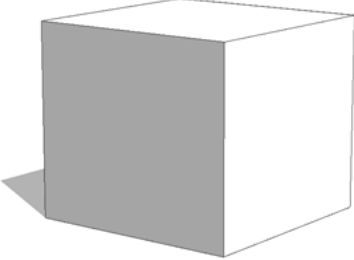
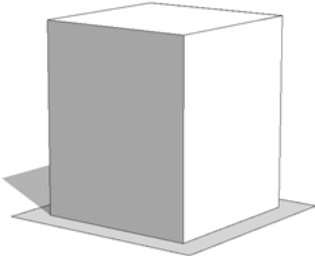
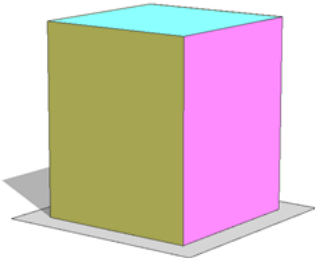
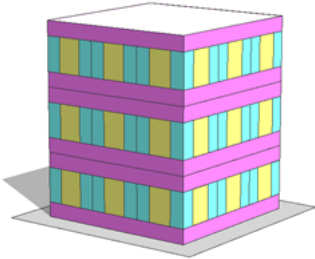
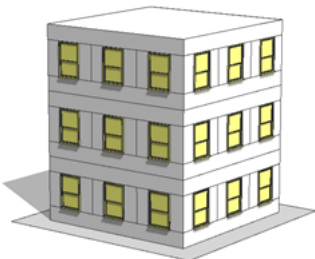
Or, you can use inline comments:

```
Lot -->  
  Garden House /*Garage*/ Fence  
  ...  
  comp(f){ front : F | /* side : S | */ top : T }
```

# Essential shape operations

# Essential shape operations

Shape operations are the actions that you can apply to shapes. You can find a list of all available operations in [CGA reference](#). The following is a list of essential operations with descriptions:

Extrusion	<div>Extrudes the input shape to a given height. <code>Lot --&gt; extrude(13) Mass</code></div>	
Transformation	<div>Used to place, orient, and size the scope and geometry. <code>Mass --&gt; s('0.75, '1, '0.75)           center(xz)           Block</code></div>	
Component split	<div>Splits a shape into its subcomponents, for example, a volume into its faces. <code>Block --&gt; comp(f) { side :                   Facade   top : Roof. }</code></div>	
Subdivision split	<div>Splits a geometry along scope axes. <code>Facade --&gt; split(y) { 1 :                   Facade.   ~3 : Floor   1 :                   Facade. }* Floor --&gt; split(x) { 1 :                   Floor.   ~2 : Window   1 :                   Floor. }*</code></div>	
Insert assets	<div>Loads an asset and inserts it in the current scope. <code>Window --&gt; i("window.obj")</code></div>	

# Extrusion

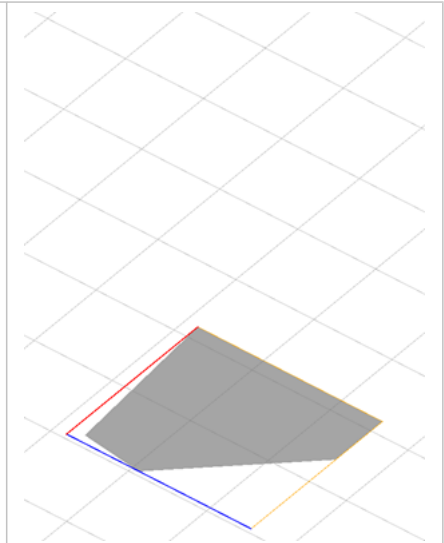
Extrusion is typically the first step to generate a 3D building from a 2D footprint. This operation increases the dimension; for instance, a two-dimensional building footprint can be extruded to a three-dimensional mass model. See [extrude operation](#) in CGA reference.

## Example

See the following examples for more information about extrusion:

`Lot --> Lot.`

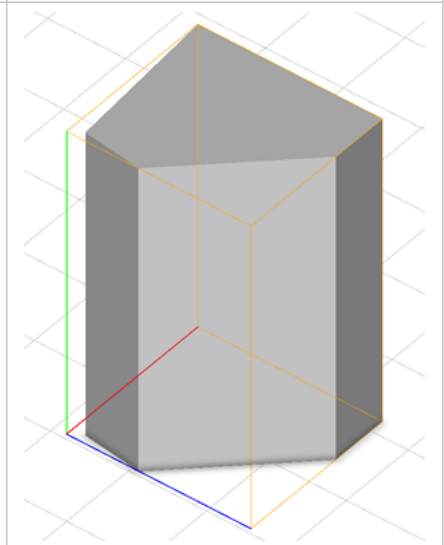
A 2D building footprint in its initial scope (yellow). The scaled x- and z-axes of the scope are illustrated in red and blue, respectively. The y-dimension of the scope is zero.



`Lot --> extrude(4)`

The 2D building footprint is extruded to a 3D mass model. The extrusion direction is orthogonal to the shape (along the normal of the footprint polygon). The scope has changed. In this example, the x- and z-dimensions are the same but the y-dimension of the scope has changed to the extrusion height.

If a footprint lies on a hill, it may be preferable to extrude along a world coordinate axis rather than the polygon normal. This and other extrusion variants are explained in the [extrude operation](#) topic.



# Transformation

The following transformations are available to modify the scope of the current shape:

- `t(tx, ty, tz)`—Translates the scope's position along the scope axes.
- `r(rx, ry, rz)`—Rotates the scope around its origin by adding `rx`, `ry`, and `rz` to the scope's rotation vector `scope.r`. You can also rotate around the scope center by writing `r(scopeCenter, rx, ry, rz)`.
- `s(sx, sy, sz)`—Sets the scope's size to the values of `sx`, `sy`, and `sz`. In contrast to the translate and rotate operations, the parameter values are not added but overwritten. Note that the size operation sets the size in absolute values (for example, meters or yards) and does not perform a relative scaling.
- `center(axes)`—Translates the scope of the current shape so that its center corresponds to the center of the scope of the previous shape on the shape stack, according to the `axes`. The latter determines in which axis direction (of the previous shape on the shape stack) the translation is performed.

## Relative operator

For the `t()` and `s()` operations, you can transform the absolute values `tx`, `ty`, `tz` or `sx`, `sy`, `sz` to values relative to the scope size using the `'` operator.

```
s('0.5, '1, '1)
t('2, 0, '3)
```

This is equal to the following:

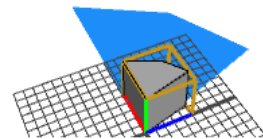
```
s(0.5*scope.sx, 1*scope.sy, 1*scope.sz)
t(2*scope.sx, 0*scope.sy, 3*scope.sz)
```

## Examples

### Setting the size

The extruded Lot is set to an absolute size of 5 units in all three dimensions.

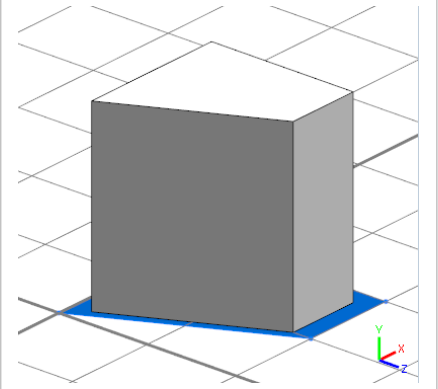
```
Lot --> extrude(10)
      s(5, 5, 5)
```



### Relative resizing and center

The scope is first sized down by using the `s()` operation in conjunction with the relative operator `'`, and then centered (relative to the scope of the Lot shape) and finally extruded to a 3D geometry.

```
Lot --> s('0.8, '1, '0.8)
        center(xz)
        extrude(20)
```



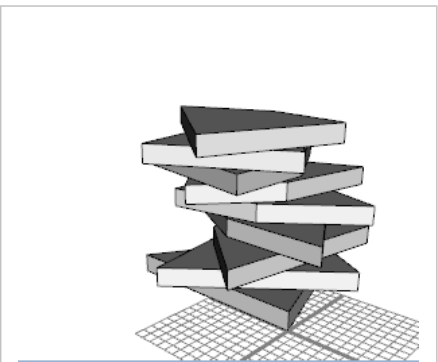
## Rotation and center

Each split shape is first rotated around its scope origin and then centered.

```
Lot -->
  extrude(18)
  split(y) {
    2 : r(0, 360*split.index/split.total, 0)
        center(xyz) X.
  }*
```

Using

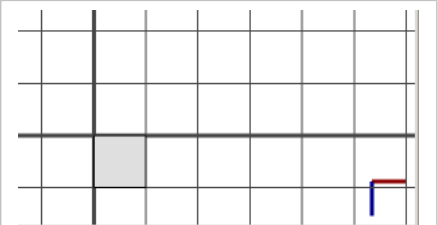
`r(scopeCenter, 0, 360*split.index/split.total, 0)`  
instead of the `r()` `center()` sequence gives the same result.



## Translate—Rotation concatenation

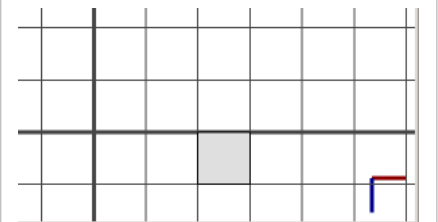
This is the shape you start with.

```
A --> primitiveCube()
```



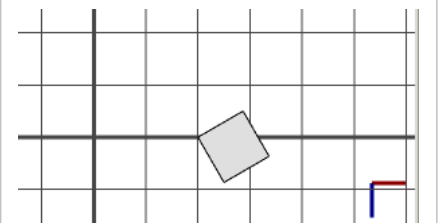
First a translation of two units along the x-axis.

```
A --> primitiveCube()
      t(2, 0, 0)
```



Then a rotation of 30 degrees around the y-axis.

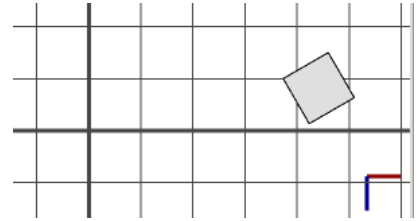
```
A --> primitiveCube()
      t(2, 0, 0)
      r(0, 30, 0)
```



And another translation of 2 units along the x-axis:

- Translations are along the scope's x-axis; in other words, the rotation changes the global translation direction.
- The relative operator ' ' is used—in this example, it does not make a difference because `scope.sx` is 1.

```
A --> primitiveCube()  
      t(2, 0, 0)  
      r(0, 30, 0)  
      t('2, 0, 0)
```



See the [t operation](#), [s operation](#), [r operation](#), and [center operation](#) topics in the CGA reference for more details.

# Component split

A typical operation is to deconstruct an architectural design into geometric components. In the CGA shape grammar, the component split allows you to divide shapes into shapes of smaller dimensions.

The operation below splits a predecessor shape, based on its geometry, into its components and runs a set of operations on each component.

```
comp(component){selector : operations | selector : operations ...}
```

The component parameter identifies the type of the component to split; for example, it can be set to **f** for faces, **e** for edges, or **v** for vertices. The selector parameters define the selection of components.

As a basic example, the following rule creates a shape B for all faces of shape A's geometry:

```
A --> comp(f) { all : B }
```

Similarly, use the following two rules to split into edges and vertices, respectively:

```
comp(e) { all : B }
```

```
comp(v) { all : B }
```

## Selectors

To access only selected components, use operation calls such as the following to create a shape that consists of the original shape's third face:

```
comp(f) { 3 : B }
```

Such calls are not generic and require the user to be aware of the topology of the predecessor shape's geometry. Therefore, as an alternative, use selectors:

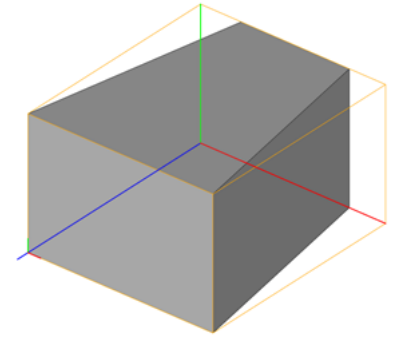
```
Building --> comp(f) { side : Facade }
```

This CGA grammar selects only the vertical **side** faces of the **Building** geometry and creates the new facade shapes accordingly. To do this, the rule interpreter analyzes the orientation of the geometry components (the direction of the face normal relative to the orientation of the scope).

## Example

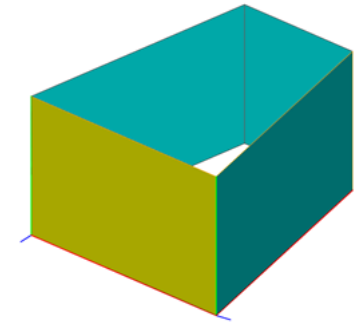
```
Building --> extrude(10)
```

A building footprint is extruded to a 3D mass model. The resulting scope is illustrated in yellow. The x-, y-, and z-dimensions of the scope are illustrated in red, green, and blue, respectively. The pivot axes are shown in the same colors.



```
Building --> extrude(10)
comp(f) { front : color("#FFFF00") Main. |
         side  : color("#00FFFF") Side. }
```

The mass model is split into one Main facade (front) and a number of Side facades (side). Each face is now the geometry of a new shape (Main and Side shapes). The new shapes' scopes and pivots depend on the faces' orientation. The x-axis points along the first edge and the z-axis points along the face normal. The scope's z-dimension is zero.



Typically, the facades are then subdivided further into floors. Each of the new Main and Side shapes has its pivot and scope positioned and oriented so that the facade rules can be written conveniently.

See the CGA [component split operation](#) for more information.

# Subdivision split

You can use the split operation to model shapes and to set up geometry by splitting a larger geometry object into smaller ones. The split operation is central to creating designs with the CGA shape rules. The basic definition for split is as follows:

```
split(axis){size : operations}
```

See the CGA [split operation](#) topic for detailed information.

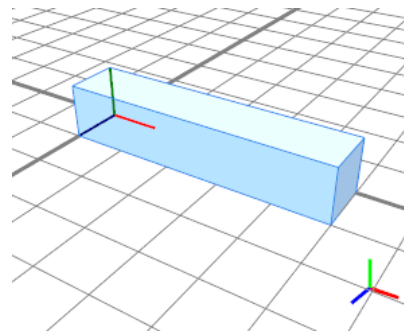
The following are split operation types:

- Absolute split
- Relative split
- Floating split
- Repeat split

The following examples start with an introductory rule and continue with examples for each split type:

```
Lot --> s(5, 1, 1)
        primitiveCube()
        Blue(1)
Blue(height) --> s('1, 'height, '1)
                color("#84C0fC")
```

In this introductory rule, the initial scope is resized and a cube is inserted in that scope. The scope's x-dimension (the length of the cube) is set to 5. The y- and z-dimensions of the scope are set to 1. The Blue rule resizes the scope's y-dimension (the height of the cube) relative to scope.sy using the relative operator '. Because the rule parameter height is set to 1, the height of the cube remains unchanged.

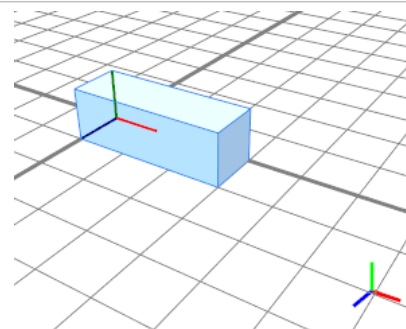


## Absolute split

The absolute split cuts the geometry at absolute values.

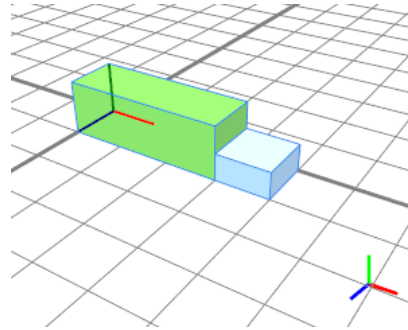
The operation `split(x) { 3 : Blue }` splits the geometry along the x-dimension of the scope. The cube of length 5 is cut at an absolute value of 3 units and replaced with the successive cube shape Blue.

```
Lot --> s(5, 1, 1)
        primitiveCube()
        split(x) { 3 : Blue(1) }
```



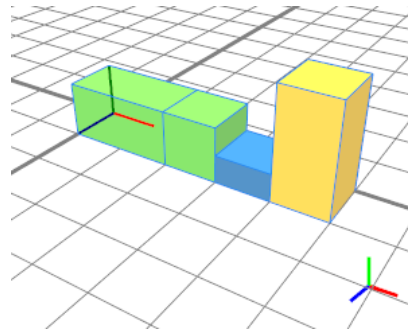
This example shows what happens if two absolute split values are used (3 units in green and 1 unit in blue). After the split, the total resulting length is 3+1=4.

```
Lot --> s(5, 1, 1)
        primitiveCube()
        split(x) { 3 : Green(1)
                  | 1 : Blue(0.5) }
```



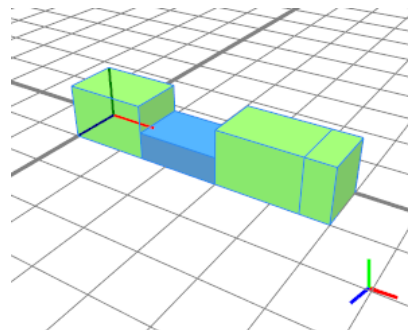
This example shows the effect of absolute values within a (limited) scope. The resulting shape Yellow has a length of 1 instead of 2 since there is a total length of 5. The rightmost shape Blue will not be generated.

```
split(x) { 2 : Green(1) |
          1 : Green(1) |
          1 : Blue(0.5) |
          2 : Yellow(2) |
          1 : Blue(1)   }
```



No splits will be produced with negative or zero-sized values. Both Yellow shapes are not generated. Blue starts at 1.5. The last Green shape stops at the total length of 5.

```
split(x) { 1.5 : Green(1) |
          -2 : Yellow(1) |
          1.5 : Blue(0.5) |
          1.5 : Green(1) |
          0 : Yellow(1) |
          1.5 : Green(1) }
```



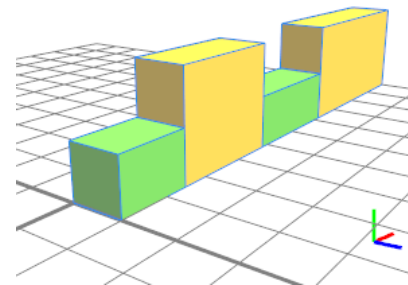
## Relative split

The relative split uses scales relative to the scope size instead of absolute values. Relative values are denoted by the operator ' followed by a value between 0 and 1.

This example cuts the scope using the Golden Ratio.

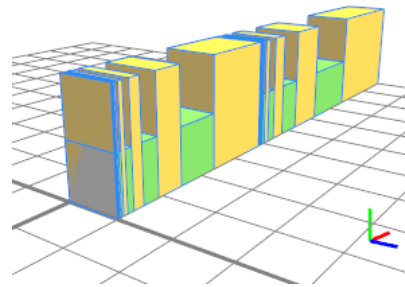
```
split(x) { '0.382 : A | '0.618 : B }

A --> split(x) { '0.382 : Green(1) |
                '0.618 : Yellow(2) }
B --> split(x) { '0.382 : Green(1) |
                '0.618 : Yellow(2) }
```



This example shows a recursive split using the Golden Ratio. The recursion stops when the geometry gets too small for a further split.

```
split(x) { '0.382 : A | '0.618 : B }
A --> split(x) { '0.382 : A      |
                  '0.618 : C      }
B --> split(x) { '0.382 : B      |
                  '0.618 : D      }
C --> split(x) { '0.382 : Green(1)
                  '0.618 : Yellow(2) }
D --> split(x) { '0.382 : Green(1)
                  '0.618 : Yellow(2) }
```

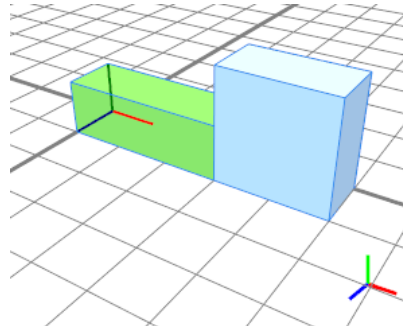


## Floating split

A floating split adapts values so that the entire space is filled. Floating splits are denoted by the operator `~`.

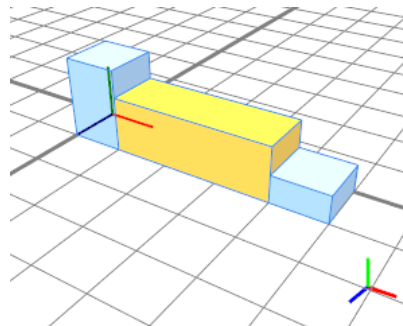
In this example, the `split` operation splits the cube of length 5 at 3 units (green). The remainder geometry is then cut at 1 unit. This time, the floating operator `~` is used, which expands the 1 unit over the remaining space of the first cut, for example,  $5-3=2$  units (blue).

```
split(x) { 3 : Green(1) |
          ~1 : Blue(2) }
```



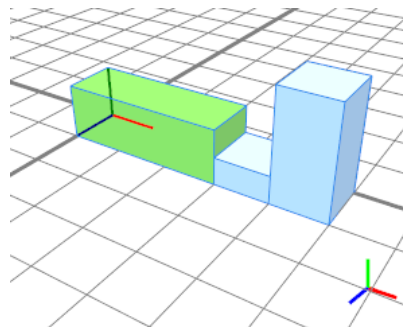
The floating value 2 fills the space (3 units in yellow) between two absolute values (1 unit in blue).

```
split(x) { 1 : Blue(1.5) |
          ~2 : Yellow(1) |
          1 : Blue(0.5) }
```



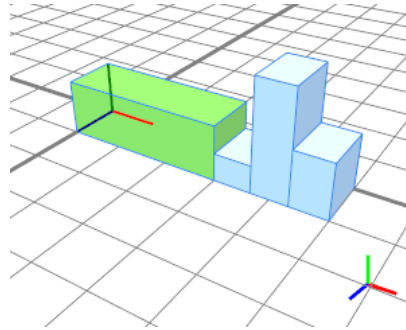
Multiple floating splits with same unit (each 1.5) are distributed evenly over the remaining scope space (2 units).

```
split(x) { 3 : Green(1) |
          ~1.5 : Blue(0.5) |
          ~1.5 : Blue(2) }
```



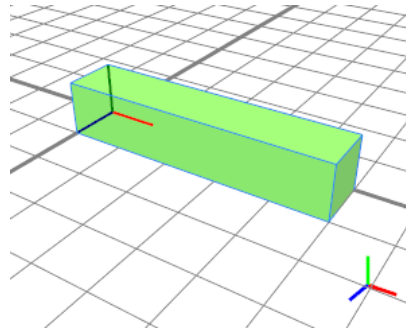
Three floating splits are distributed evenly.

```
split(x) { 3 : Green(1) |
          ~2 : Blue(0.5) |
          ~2 : Blue(2)   |
          ~2 : Blue(1)   }
```



If there is no space left for a floating split, the corresponding shape is not generated (Blue in this case).

```
split(x) { 5 : Green(1) |
          ~1 : Blue(1)   }
```



## Repeat split

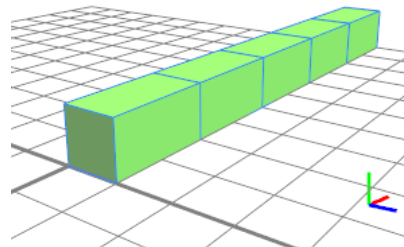
The repeat split is used to repeat geometry within a given scope. With the repetitive use of a group of geometric objects, it establishes a recognizable pattern. A typical scenario is the alternating arrangement of windows and columns in the facades of common high-rise office buildings. An asterisk (\*) after the bracket denotes the repetitive split.

In the first example, a scope of length 10 is filled repetitively with a green shape with absolute length 2. Exactly 5 cubes of length 2 fit in the scope.

```
Lot --> s(10, 1, 1)
        primitiveCube()
        split(x) { 2 : Green(1) }*
```

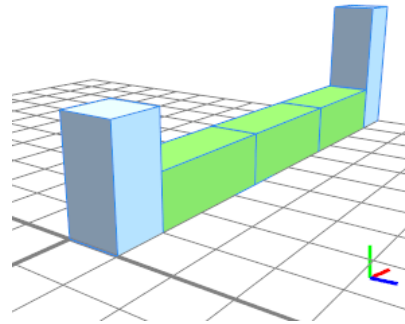
A floating value of 2.1 generates the same result because a split to 5 successor shapes (with a length of 2 each) still approximates the target length of 2.1 best.

```
split(x) { ~2.1 : Green(1) }*
```



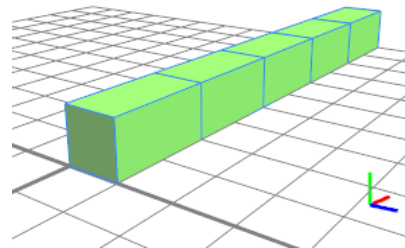
A floating repeat split (green) is bordered by two absolute splits (blue). Note that the resulting length of the floating splits is  $(10-2)/3 \approx 2.66$ , which is the best approximation for the target value 2.5.

```
split(x) {
  1 : Blue(2)
  { ~2.5 : Green(1) }*
  1 : Blue(3)
}
```



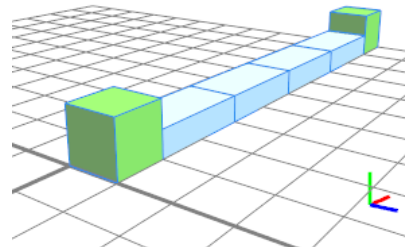
The repetitive absolute split (green) will be evaluated first independently on whichever side of the split the repetition is denoted. The floating splits do not get any space to be generated.

```
split(x) {
  { 2 : Green(1) }*
  ~1 : Blue(2)
}
split(x) {
  ~1 : Blue(2)
  { 2 : Green(1) }*
}
```



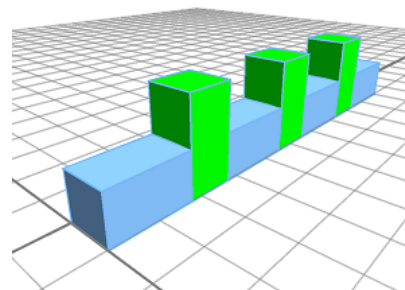
This example shows that a repeat split with absolute (or relative) values can be bordered by two neighboring floating (or absolute) splits.

```
split(x) {
  ~1 : Green(1)
  { 2 : Blue(0.5) }*
  ~1 : Green(1)
}
```



The size of the first floating split and the size of the repetitive floating split is the same. The 2 units are equally distributed 4 times (blue) to the resulting length of  $(10-3)/4 = 1.75$ .

```
split(x) {
  ~2 : Blue(1)
  { 1 : Green(2)
    ~2 : Blue(1) }*
}
```

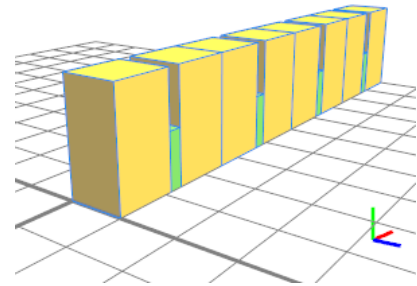


## Repeat split—Patterns

You can use repeat operations to create patterns. The following example shows that geometric objects can be grouped into a bracket and repeated:

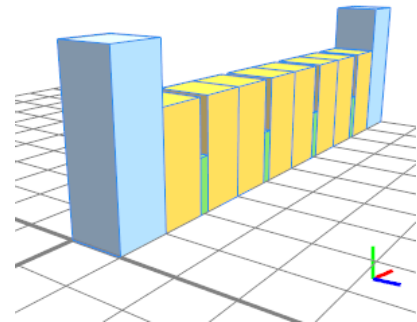
A pattern is created and repeated.

```
split(x) { ~1 : Yellow(2) |
          0.25 : Green(1) |
          ~1 : Yellow(2) }*
```



Repeating patterns can be inserted within other geometries.

```
split(x) { 1 : Blue(3) |
          { ~1 : Yellow(2)
            0.2 : Green(1)
            ~1 : Yellow(2) }* |
          1 : Blue(3) }
```

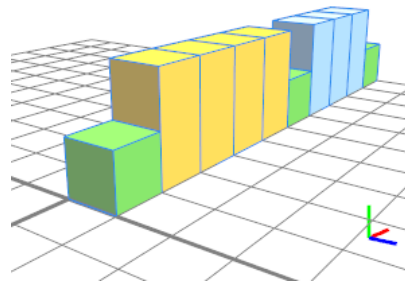


## Repeat split—Parallel

You can perform repeat splits in parallel.

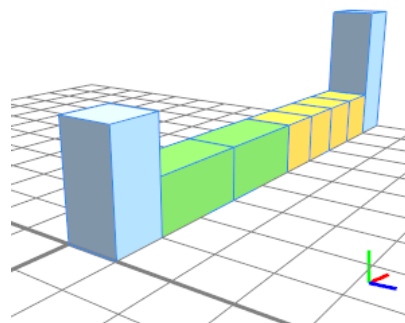
Two repeat splits (yellow and blue) are bordered by floating splits (green).

```
split(x) { ~1 : Green(1) |
          { ~1 : Yellow(2) }* |
          ~1 : Green(1) |
          { ~1 : Blue(2) }* |
          ~1 : Green(1) }
```



Two successive repeat splits (green and yellow) are bordered by absolute splits (blue).

```
split(x) { 1 : Blue(2) |
          { 2 : Green(1) }* |
          { 1 : Yellow(1) }* |
          1 : Blue(3) }
```



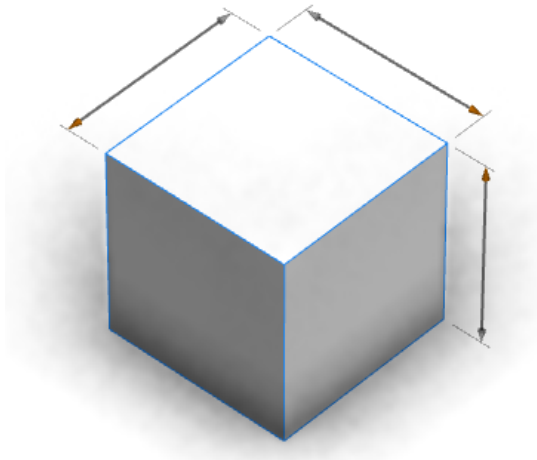
# Asset insertion

A typical last step is the insertion of assets, for example, polygon meshes such as windows, doors, and so on. This is described in detail in the CGA [insert operation](#) topic.

# Handles

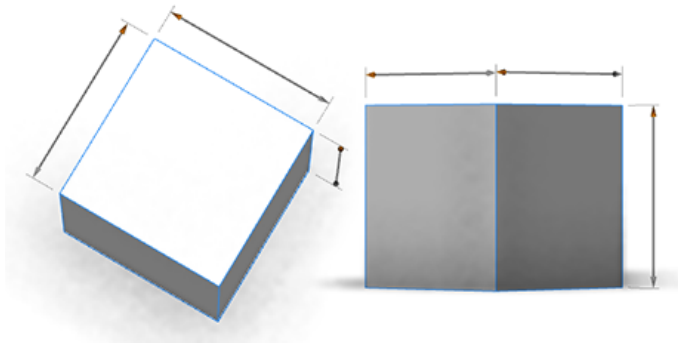
# Handles

Handles allow you to edit value attributes directly in the **Viewport** window.



*Handles with a model*

The handles are positioned so that they are always visible, whichever camera angle is used.



*Handles in multiple camera angles*

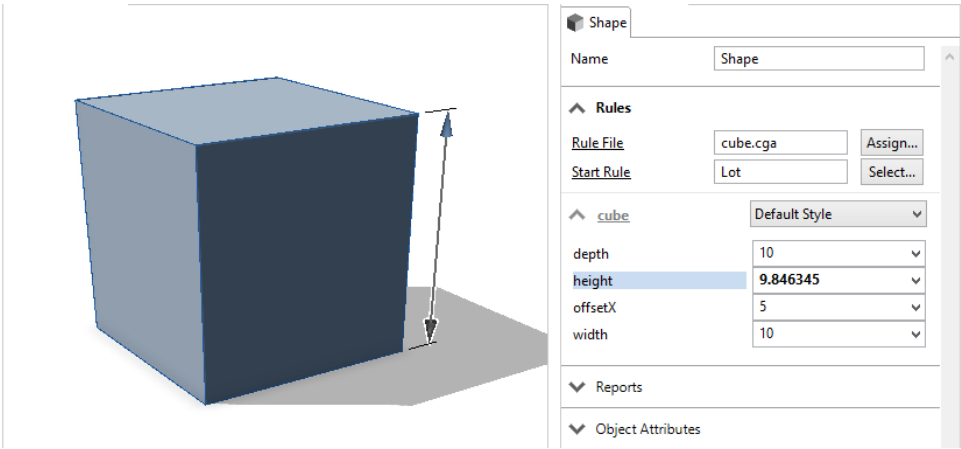
You can enable or disable the handles on the **3D View** toolbar under **View settings**



. You can also click **Edit > Preferences > Scene**. By default, handles are enabled when present.

## Using handles

When you hover the mouse pointer over a handle, CityEngine highlights both the attribute in the **Inspector** window and the associated scope in the view.



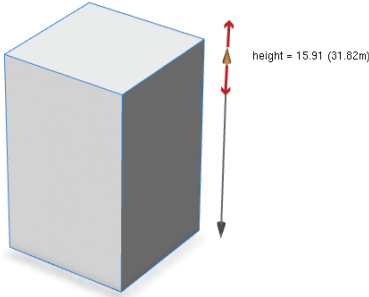
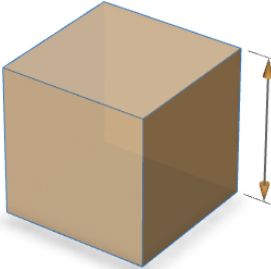
Handles in the Inspector window

 **Tip:**

Small handles or handles attached to small parts of the model are sometimes not shown. To show these handles, press the **Ctrl1** key.

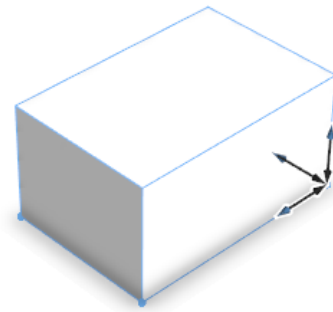
## Handle types

There are several types of handles for editing different types of attributes, such as linear distances, angles, Boolean switches, multiple choice, or colors. You can use them by dragging, or clicking, the colored (orange) elements.

Linear	
Linear handles adjust distance attributes. To edit them, drag the colored markers. When you drag the handle, the value of the attribute is shown. If the value of the attribute is different from the length of the linear handle, the length of the handle is shown in parentheses.	
Some linear handles can be dragged from either end. These handles have two colored arrows.	

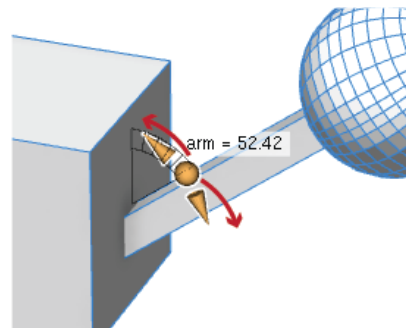
## Move

Move handles adjust position or offset attributes. They are similar to linear handles; however, the length of the move handles is always constant, and dragging one end also moves the opposite end by the same amount.



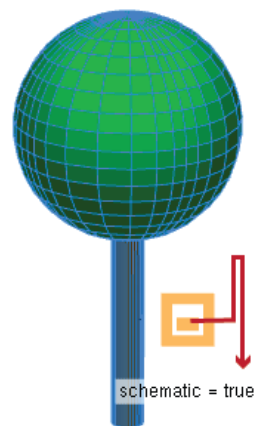
### Rotational

Rotational handles adjust angle attributes. To edit them, drag the colored sphere.



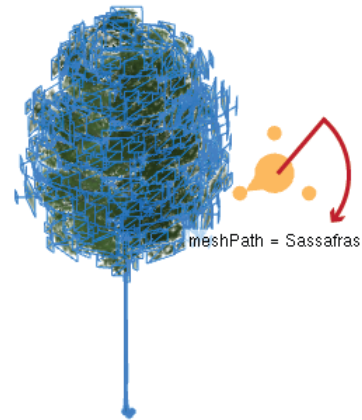
### Toggle

Toggle handles adjust Boolean (on or off) attributes. To edit them, click the colored square, or drag the square up and down.



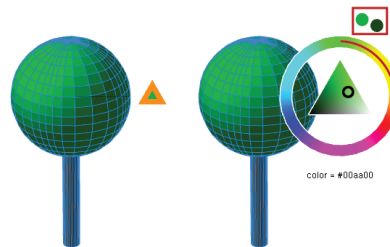
### Selector

Selector handles edit multiple choice values. To edit them, click the colored circle, or drag in a clockwise or counterclockwise direction around the circle.



## Color

Color handles edit color attributes. To edit them, click the colored triangle, and drag in (or outside) the hue circle to adjust the hue. Drag inside the triangle to adjust the saturation and brightness. The selected color is shown by a circle within the triangle. Alternatively, click the triangle and drag to a color swatch to select that color.

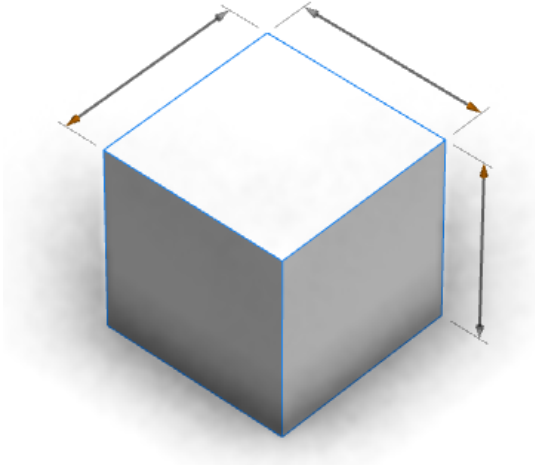


You can limit the range of handles by the rule (using [@Range](#) and [@Enum](#) annotations). In this case, the values of the handles are constrained when editing.

# Create handles

## Create handles

To create handles, add aa `@Handle` annotation to an attribute.



Model with `@Handle` annotation added

The handles in the image above were created using the following attribute annotations:

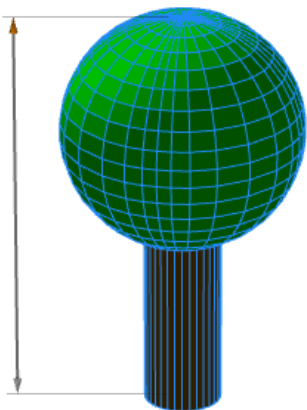
```
@Handle(shape=Cube, axis=y)
attr height = 30

@Handle(shape=Cube, axis=x)
attr width = 30

@Handle(shape=Cube, axis=z)
attr depth = 30

Lot -->
    s(width, 0, depth)
    extrude(height)
    Cube
```

The `shape=Cube` option attaches the handle to the shape Cube. The `axis=x|y|z` option specifies a scope direction for the handle. Thin gray extension lines connect the handle to the shape that it is measuring. You can apply handles to a wide range of objects.



Tree model with handle

The handle in the image above was created using the following annotation:

```
@Handle(shape=TreeCenter, axis=y, reference=center, slip=screen)
attr height = 30
```

You can also create handles that do not move with the camera, for example:

```
@Handle(shape=TreeCenter, axis=y, reference=origin, slip=inside, occlusion=false)
attr height = 30
```

The advanced options `reference` and `slip` specify the handle's position and movement as the camera moves. In this case, the height handle rotates smoothly around the tree with the camera.

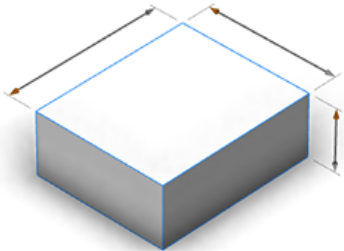
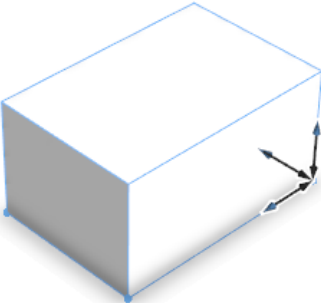
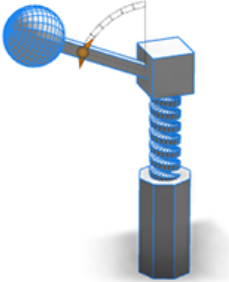
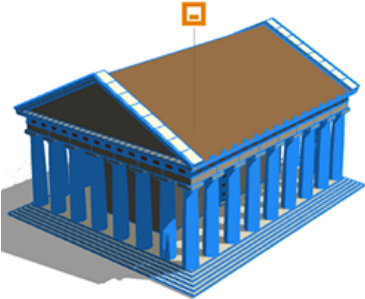
 **Note:**


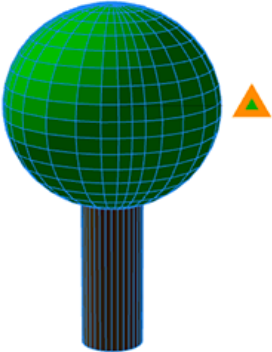
Adding many handles or adding handles to complex models may lead to low frame rates on low-end hardware.

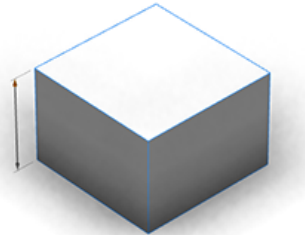
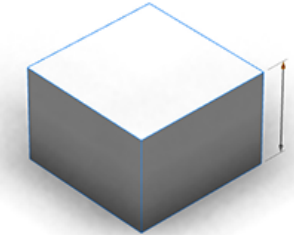
## General options

The default options in the following table are marked with an asterisk. If any option is omitted, the default value is taken. The only option that must be specified is `shape`.

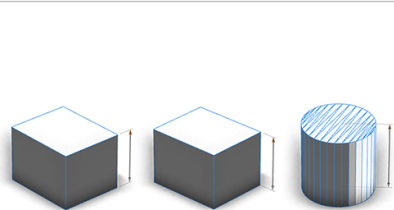
shape
shape=shape_name^argument_count
Selects the shape of the rule with name <code>shape_name</code> . The parameter <code>argument_count</code> specifies the number of arguments that the rule takes.
If <code>^argument_count</code> is omitted, <code>^0</code> is assumed. An asterisk (*) allows referencing <a href="#">anonymous leaf shapes</a> , for example, <code>shape=Box*^3</code> .

<b>type</b>	
type=linear* move angular toggle selector color	
Selects the type of handle to create and color for editing color string attributes.	
Additional options specific to each type are as follows:	
<p><b>type=linear</b></p> <p>Handles with a linear type are used for editing float values that represent distances.</p>	
<p><b>type=move</b></p> <p>The move type is used for editing float values that represent positions; in contrast to the linear type, the handles are always shown at a constant length.</p>	
<p><b>type=angular</b></p> <p>The angular type is for float attributes that represent angles.</p>	
<p><b>type=toggle</b></p> <p>The toggle type is for Boolean attributes.</p>	

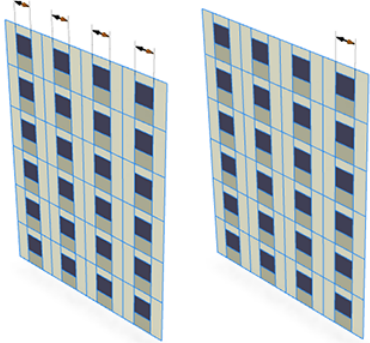
<p><b>type=selector</b></p> <p>The selector type is for attributes with the <a href="#">@Enum annotation</a>.</p>	
<p><b>type=color</b></p> <p>The color type is for editing color string attributes.</p>	

<p><b>align</b></p> <p><code>align=topLeft   left   bottomLeft   bottom   bottomRight   right   topRight   top   default*</code></p> <p>Selects a screen direction as the offset direction preference for a reference handle. <code>default</code> selects the nearest location outside of the model's silhouette.</p>	
<p>Linear example of <code>align=left</code></p>	
<p>Linear example of <code>align=right</code></p>	

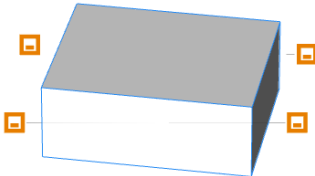
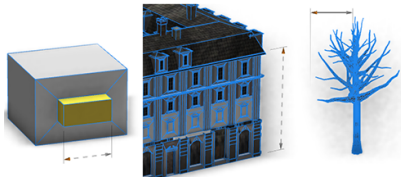
<p>Toggle example of <code>align=topLeft</code></p>	
<p>Toggle example of <code>align=right</code></p>	

<p><b>slip</b></p>	
<p><code>slip=scope* screen inside</code></p>	
<p>This option specifies the way in which reference handles may be moved outside the model's silhouette. <code>slip=scope</code> moves the handle in directions given by the scope axes, <code>slip=screen</code> moves the handle parallel to the camera, and <code>slip=inside</code> disables the offsetting behavior.</p>	
<p>Different <code>slip</code> options are appropriate for different shapes. For example, the length of cylindrical objects is best specified using <code>slip=screen</code>, while the dimensions of a cuboid should use <code>slip=scope</code>.</p>	
<p><code>slip=scope</code> causes the handle to align itself to the edges of the scope as the camera moves. In contrast, <code>slip=screen</code> maintains a constant offset direction, independent of camera location.</p> <p>It is recommended that cuboid objects use <code>slip=scope</code>, while cylindrical or spherical objects will use <code>slip=screen</code>. While it is possible to use either of these options in any case, following these recommendations presents a consistent and intuitive interface to users.</p>	

<p><b>repeat</b></p>	
<p><code>repeat=chain* none</code></p>	
<p>Chained handles will be clustered into a single continuous chain. Linear handles of several different attributes with the same orientation can be grouped onto one chain if there is sufficient space.</p>	

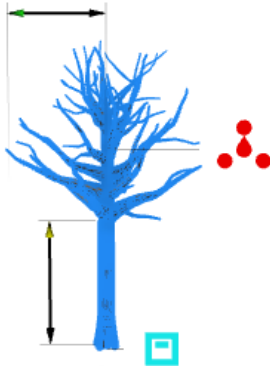
<p>Left: repeat=chain, right: repeat=none</p> <p>If repeat=none is set, CityEngine determines an appropriate location for a single handle, preferring locations with short extension lines and long linear handles.</p>	
---	---

<b>minDisplaySize</b>
<code>minDisplaySize=pixels</code>
If the size of a handle is below <code>pixels</code> , it is not shown. Users can override this behavior by pressing the <code>Ctrl</code> key.
The size of a handle with linear type is the screen length. The size of an angular handle is the distance between zero degrees and the handle. The size of toggle, selector, or color handles is the screen length of the shape's scope's shortest edge.

<b>extensionLines</b>	
extensionLines=scope* silhouette fade off	
Specifies the style of the extension lines associated with a handle.	
<p>Clockwise from lower right: extensionLines=scope, extensionLines=fade, extensionLines=off and extensionLines=silhouette.</p>	
<p>The image shows typical use cases for the different extension line types: scope (left) is used to highlight embedded features, silhouette (middle) with obvious features is used to avoid cluttering the geometry, and fade (right) is used for irregular shapes.</p>	


<b>translate</b>
<code>translate={translate_x,translate_y,translate_z}</code>
Specifies a scope-relative translation of the handle. For example, <code>translate={3,0,0}</code> translates the handle by 3 * <code>scope.sx</code> in the direction of the scope's x-axis.

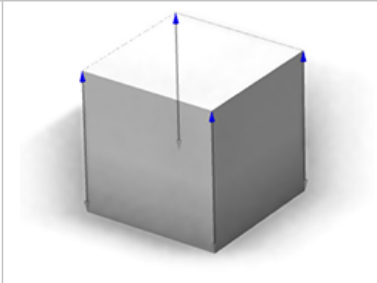
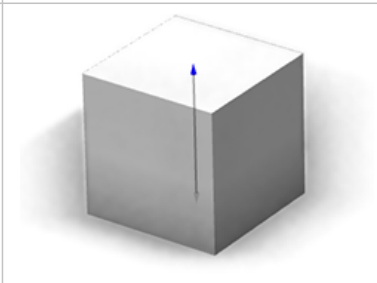
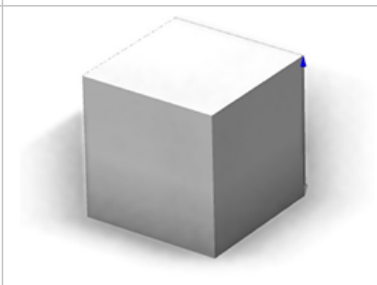
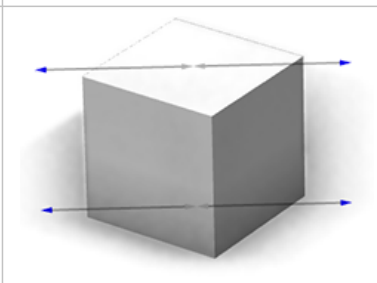
occlusion
occlusion=true* false
Enables or disables occlusion handling for the handle. When enabled, handles are only shown when their reference positions are visible. However, for complex models this may cause the handle to flicker. Using occlusion=false resolves this issue, and never hides the handle because of occlusion.

color
color=hexCodeString
Typically, the color of handles is given by the viewport highlight color. color allows users to override this behavior with a preferred color hexCodeString is specified as an RGB hex code string within quotes, for example, color="#FF0000".
<div>The vertical linear handle uses the parameter color="#FFF00", the horizontal linear handle color="#33FF33", the selector handle color="#FF0000", and the toggle handle color="#33FFFF".</div> <div></div>

Linear type options

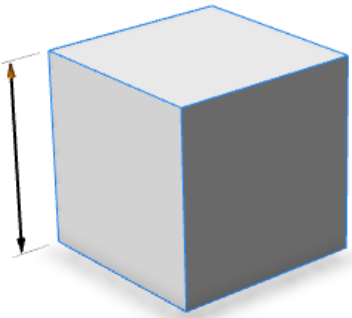
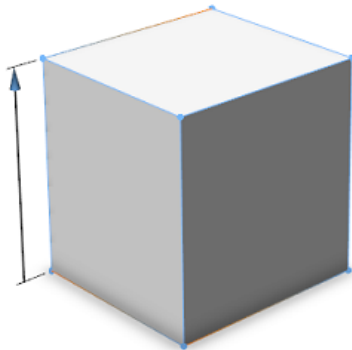
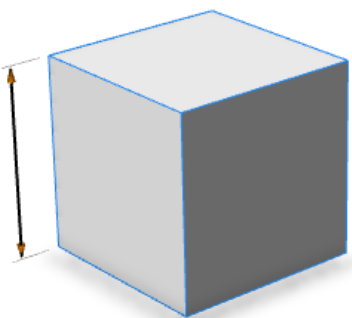
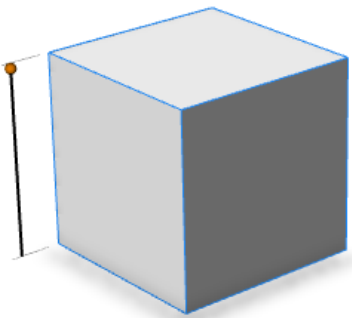
Linear handles are assumed to be associated with a scope edge. The length of the linear handle is given by the scope edge specified by axis.

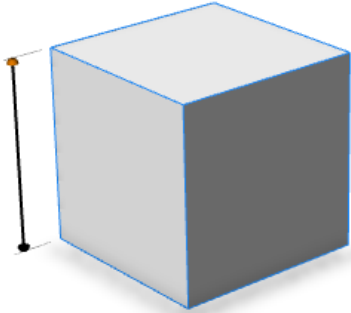
reference
reference=edges* center origin radial
Specifies the location of the reference handles.
<div> <b>Note:</b></div> The location of radial handles depends on the camera position.
Handles are moved to their offset position using the direction given by the slip option.

<p><b>reference=edges</b> Typical use cases are boxes or cuboid geometries (as shown in the first example on this page).</p>	
<p><b>reference=center</b> Suitable for the length of cylindrical or spherical objects (such as the tree's height in the second example on this page).</p>	
<p><b>reference=origin</b> Used when a feature is one dimensional or when the exact position of a handle must be specified. The shape's origin is at the right of the cube.</p>	
<p><b>reference=radial</b> Can be used for the width of cylindrical objects (such as the dead tree model above).</p>	

<b>axis</b>
<code>axis=y* x z x- y- z-</code>
The linear handle is positioned along the specified scope axis of the shape.

<b>skin</b>
<code>skin=doubleArrow* singleArrow diameterArrow sphere hemisphere</code>
Linear handles have several options for the rendering of terminators. These options do no change the behavior of the handle, with the exception of <code>skin=diameterArrow</code> , which scales symmetrically about its center when dragged.

<code>skin=doubleArrow</code>	
<code>skin=singleArrow</code>	
<code>skin=diameterArrow</code>	
<code>skin=sphere</code>	

<code>skin=hemisphere</code>	
------------------------------	---



Move type options

Move handles are similar to linear handles and are also associated with a scope edge. However, the length of the move handle is always constant, in contrast to linear handles where it is given by the scope edge. When dragging one end of the handle, the other end also moves by the same amount. This makes them suitable for attributes representing positions or offsets.

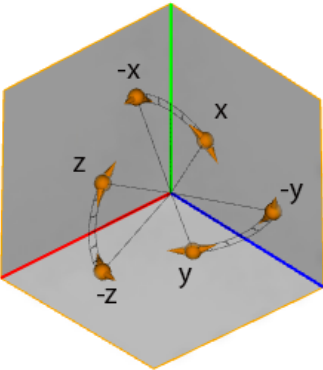
axis and skin are similar to linear handles. For reference, only center and origin are supported. Also, the slip value is restricted to inside.

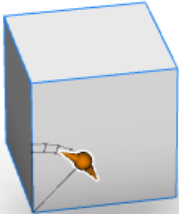
Angular type options

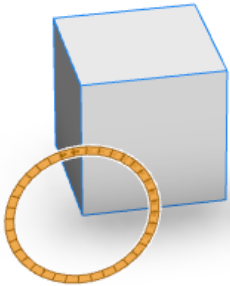
Angular handles allow you to manipulate angles. Typically a combination of axis and translate is used to position the handle in the appropriate location. By default, angular handles do not offset themselves (slip=inside). This behavior can be altered by specifying another slip value.

reference	
<code>reference=edges   center*   origin</code>	
Specifies the location of the reference handles. In the images below, the shape origin is in the lower right of the screen.	
<code>reference=edges</code>	
<code>reference=center</code>	

<code>reference=origin</code>	
-------------------------------	---


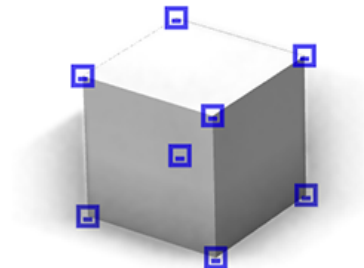


<b>axis</b>	
<code>axis=y* x z x-* y-* z-</code>	
Selects the rotation scope axis. A negative axis reverses the direction of rotation.	
The red, green, and blue lines are the x-, y-, and z-axes. For example, <code>axis=x</code> rotates around the x-axis towards the z-axis. The angular handles each have an attribute value of 30 degrees	

<b>skin</b>	
<code>skin=doubleArrow* ring</code>	
Rotational handles have two skin options: <code>doubleArrow</code> and <code>ring</code> .	
<code>skin=doubleArrow</code>	

<code>skin=ring</code>	
------------------------	---

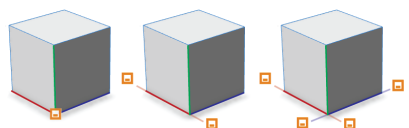
## Toggle, selector, and color type options

Toggle, selector, and color types share layout options.

<b>reference</b>	
<code>reference=edges center* origin radial</code>	
Specifies the location of the reference handles. In the images below, the shape origin is in the lower right of the screen.  <b>Note:</b> The location of radial handles depends on the camera position.	
<code>reference=edges</code>	
<code>reference=center</code>	
<code>reference=origin</code>	

<code>reference=radial</code>	
-------------------------------	--

axis

<b>if slip=scope</b>	
<code>axis=x y* z xy xz yz</code>	
Specifies the directions in which the handle can move when <code>slip=scope</code> is set. For example, handles with the option <code>axis=xy</code> can move along the x- or y-axes.	
Left: the reference position for the toggle handle at the origin, and the three axes. Middle: <code>axis=x</code> —the handle can move along the x-axis. Right: <code>axis=xy</code> —the handle can move along the x- or y-axis to find the best location.	


<b>if slip=screen</b>	
<code>axis=x y* z</code>	
In the case of <code>slip=screen</code> , the handle moves in the same way as a linear handle with the same axis parameter.	

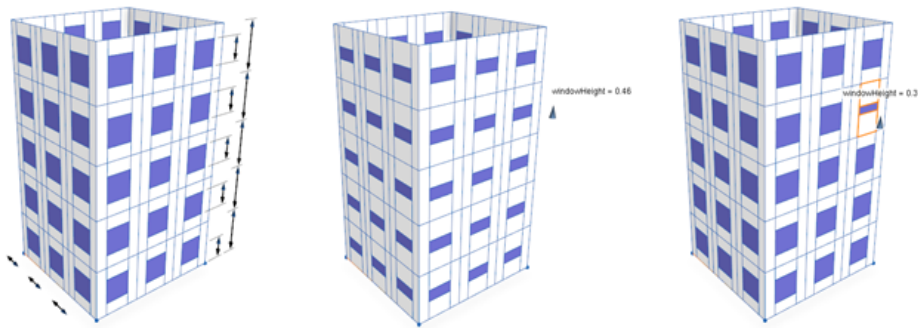
<b>if slip=inside</b>	
axis does not affect the placement of the handle if <code>slip=inside</code> .	

# Local edits

## Local edits



With handles, you can edit attributes of a selected CGA model in the **Viewport** window. When an attribute is edited, the change globally affects all occurrences of the attribute in the CGA model. While this behavior is useful in many cases, sometimes a more granular control is needed. With local edits, you can edit the value of an attribute for each occurrence. This increases the artistic control over a CGA model significantly.

For example, the window height of all windows in a building is defined by the `windowHeight` attribute. Editing this attribute using handles, or the **Inspector**, changes all the windows in the same way. Using the **Local Edits** tool , you can set the window height for each window independently.



From left to right, building windows are shown with no edits, with global edits, and with local edits.

## Activate local edits

To do this, first activate the **Local Edits** tool , and then click a building window in the **Viewport** window. Changing an attribute in the **Inspector** window or using handles will now only affect this window. To exit the **Local Edit** tool, switch to the **Selection** tool .

To use the **Local Edits** tool, complete the following steps:

1. Generate a building using a rule file with handles, for example:

```
@Handle(shape=Block, axis=y)
attr bldgHeight = 10


@Handle(shape=Floor, axis=y)
attr floorHeight = 2

@Handle(shape=Window, axis=y)
attr windowHeight = 1


@Handle(shape=Window, axis=x)
attr windowWidth = 1

Init --> extrude(bldgHeight) Block
Block --> split(y) { ~floorHeight : Floor }*
Floor --> comp(f) { side : Facade }
Facade --> split(x) { ~0.5 : Wall. | windowWidth : Tile | ~0.5 : Wall. }*
Tile --> split(y) { ~0.5 : Wall. | windowHeight : Window | ~0.5 : Wall. }
Window --> color(0.4, 0.4, 0.75)
```

There are handles for `windowHeight`, `windowWidth`, and `floorHeight` that allow for local edits.

2. Activate the **Local Edits** tool .
3. Select the building.  
The global handles appear.
4. Select a part of the CGA model or use the handles to make global edits.  
The global handles of the selected part disappear. Handles for local edits are displayed instead.

Keep the following in mind when using the **Local Edits** tool:


- The **Local Edits** tool  is available automatically for all attributes that have a handle annotation. For more information, see [Create handles](#).
- Press **Shift** and click to select multiple parts of a building. For more information, see [Multiple selections](#).
- You can right-click to use the context menu for **select higher/lower level** and **next/previous pattern** to select logical groups such as rows and columns of parts. For more information, see [Local edits on patterns](#).
- Attributes with local edits are marked orange in the Inspector. For more information, see [Manage local edits](#).


# Work with local edits

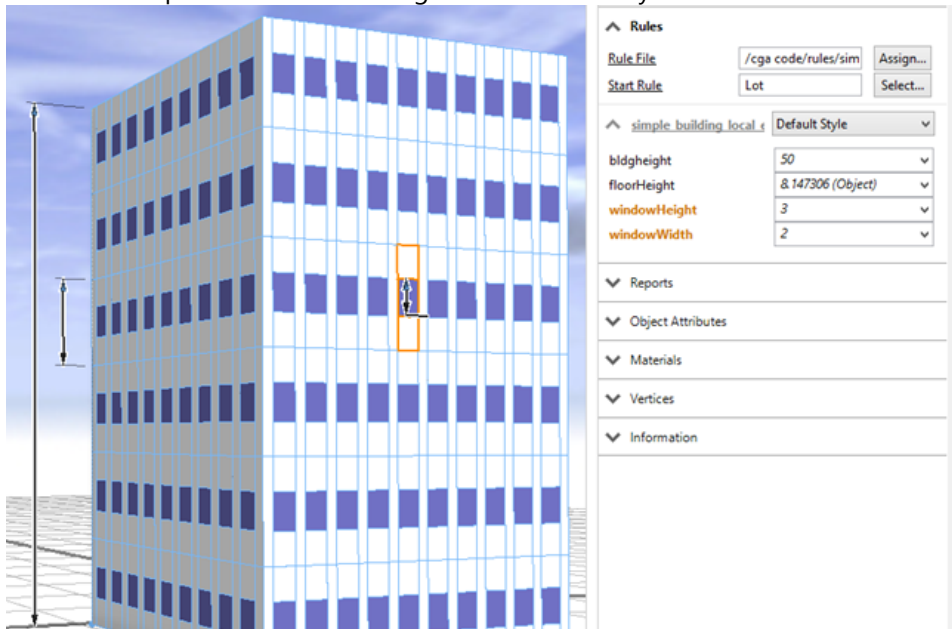
The following workflows provide detailed descriptions of **Local Edits** tool behavior and functionality.

## Use local edits in a single selection

The area of effect of a local edit depends on the selection. You can select single or multiple parts of a building.

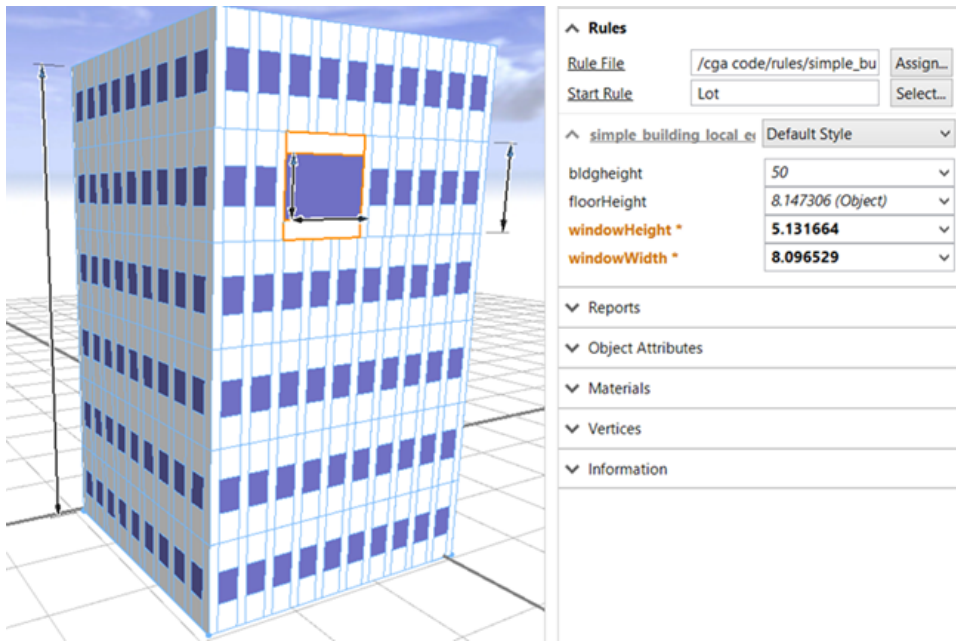
When you select components of a building with the **Local Edits** tool , an orange highlight shows the current area of effect in the **Viewport** window. The editable attributes in this area are also highlighted in orange in the **Inspector** window.

1. Click the **Local Edits** tool  to activate local editing.
2. Click the component of the building whose attributes you want to edit.

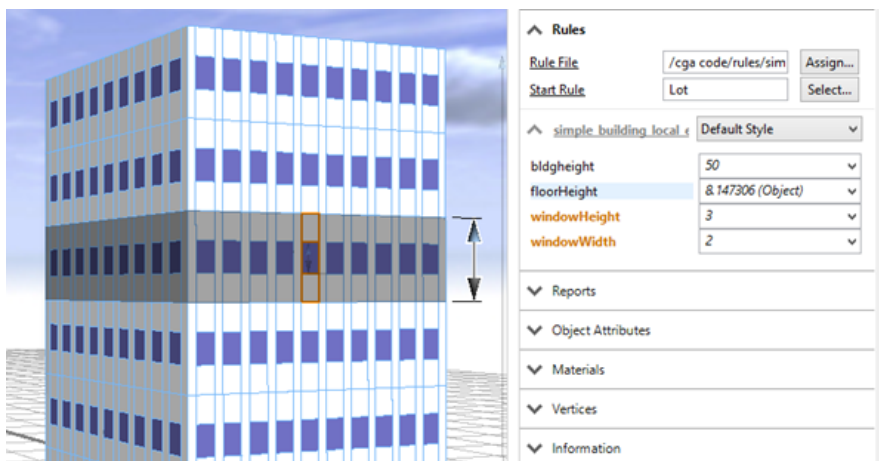


3. Edit the attributes by adjusting the handles of a highlighted attribute, or make the edits directly in the **Inspector** window. You are affecting the attributes only for the current highlighted selection.

After making local edits, the edited attributes have orange highlights with an asterisk (\*) in the **Inspector** window. The asterisk indicates that the attributes have local edits applied in the current selection. The **Inspector** window also displays the attribute values in bold after changes are made.



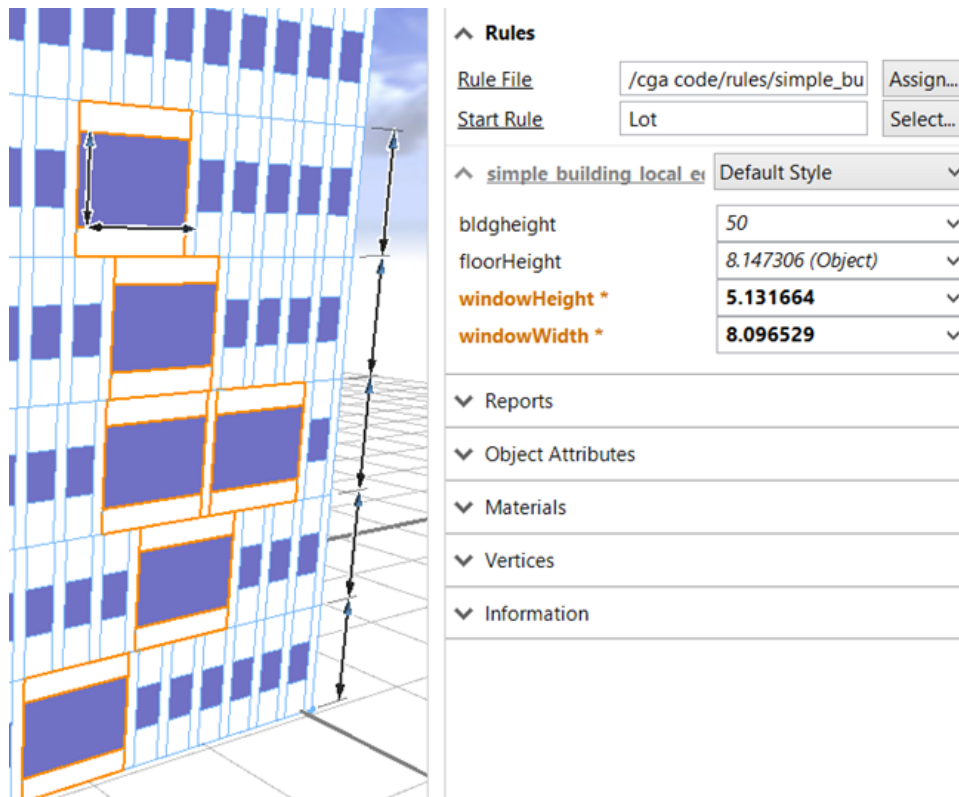
You can also edit attributes that are outside the current orange highlight, such as `floorHeight`. This has a greater area of effect. To see the area, hover over the corresponding handle.



## Use local edits in multiple selections

You can select several components and edit the attributes for the whole selection at one time. To select multiple components, do the following:

1. Press **Shift** and click to add elements to the selection.  
Edits will only affect attributes of the selected components.



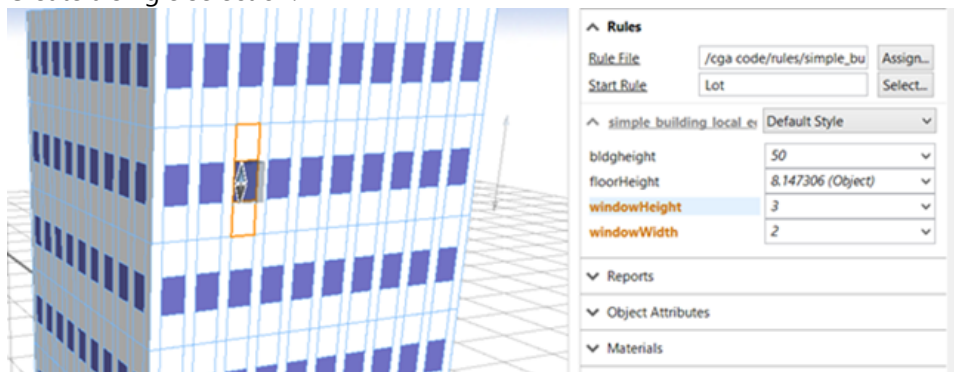
2. Press **Ctrl+Shift** and click to remove items from the selection.

## Use local edits on higher levels

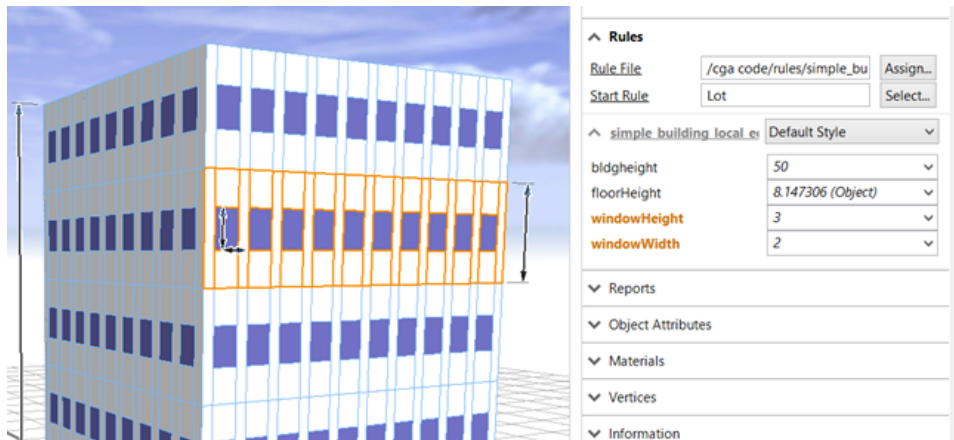
During rule application, a shape tree is created with multiple levels, such as windows, floors, and facades. See [Model Hierarchy](#) for more details.

You can use this shape tree to make local edits on multiple objects at once, for example, on all windows on one floor or on all floors on one facade. This is called local edits on higher levels. The levels where edits are possible are determined automatically, and you can select them using **Select higher level** as follows:

1. Create a single selection.

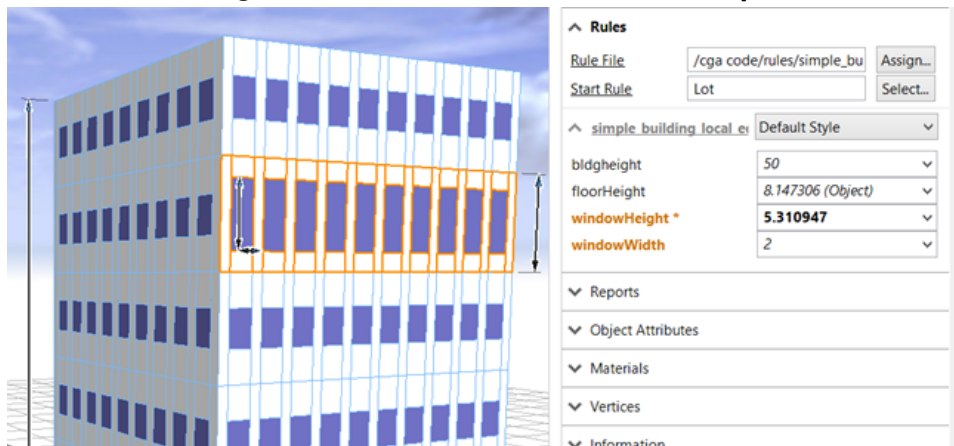


2. Right-click the local selection and choose **Select higher level**, or press **Page Up** to select the next level up.



You've selected all the windows on the higher level. The `windowHeight` and `windowWidth` attributes are highlighted in orange in the **Inspector** window, which indicates they can be locally edited.

3. Edit the `windowHeight` attribute with the handles or in the **Inspector** window.



All the windows in the selection change.

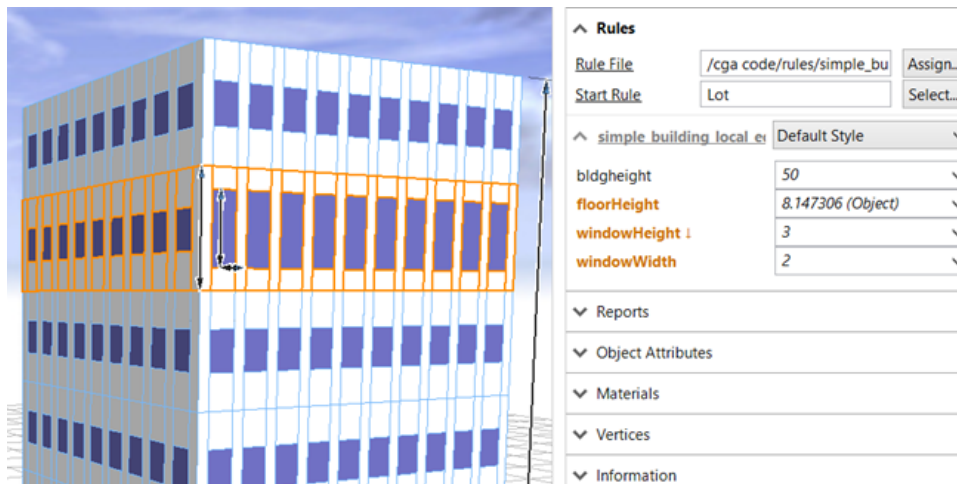
The **Inspector** window now displays the following:

- The `windowHeight` and `windowWidth` attributes are highlighted, indicating that changes will affect only the current selection.
- `windowHeight*`—Local edits have been applied to the `windowHeight` attribute at the current selection level; its bold value indicates that edits have been applied.

## Use local edits on multiple levels

By stepping even higher up, you can perform edits on multiple levels. When multiple edits apply, the lowest edit, which is closest to the leaf shape, always has precedence.

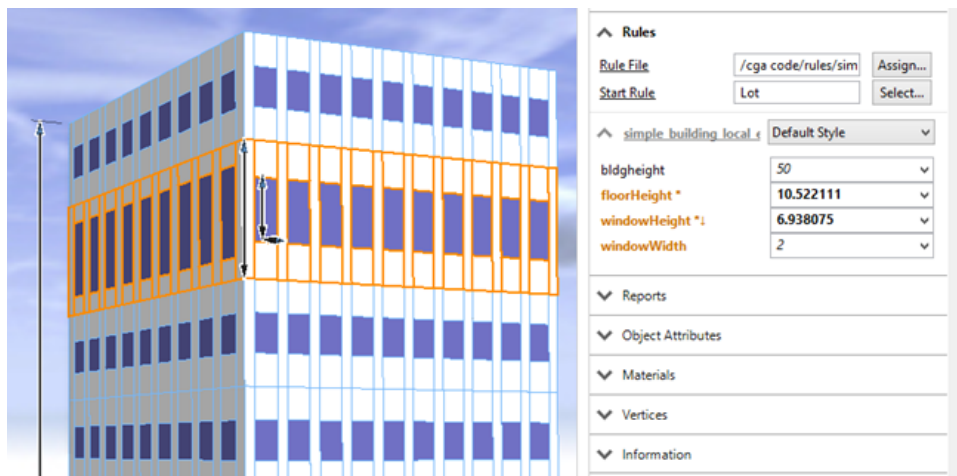
1. Press **Page Up** again to select the next highest level.



You can see in the **Inspector** window that **floorHeight** is now also highlighted.

- **floorHeight**, **windowHeight**, and **windowWidth** are highlighted, indicating local edits will only affect the selection.
  - **windowHeight↓**—Local edits have been applied at a lower level.
2. Make local edits to the **floorHeight** and **windowHeight** attributes.

Edits to **windowHeight** don't affect the previously selected level because the lower level already has an edit, which takes precedence. However, the other levels are affected, because they do not have a lower-level edit.

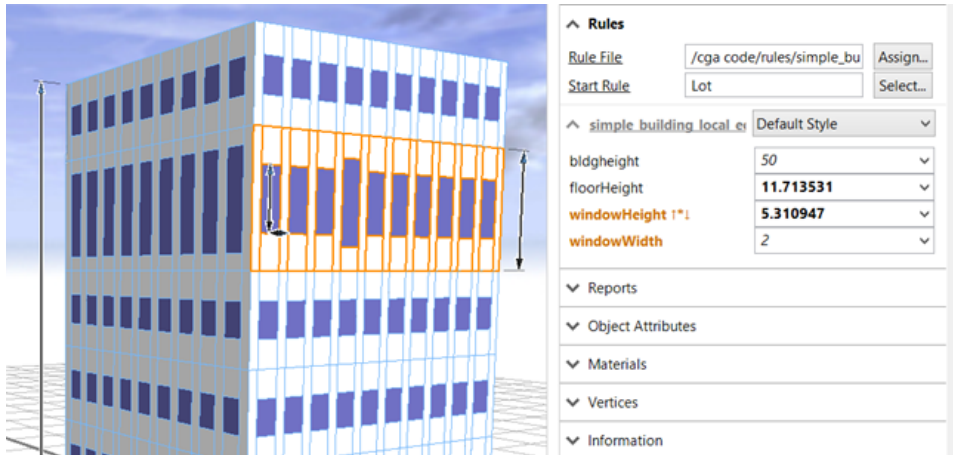


The **Inspector** window displays the following:

- **/floorHeight**, **windowHeight**, and **windowWidth** indicate local edits will only affect the current selection.
- **floorHeight\***—Local edits have been applied at the current selection level.
- **windowHeight\*↓**—Local edits have been applied at the current and lower level.
- The ↓ and ↑ symbols indicate that a local edit is applied at a higher or lower level for this attribute, relative to the current selection. Press the **Page Up** or **Page Down** keys to change the current level and display the asterisk (\*) in the **Inspector** window for attributes edited at that level.

You can add more edits on the lowest level. It is possible to have an attribute in the **Inspector** window that indicates local edits at the current selection level, a higher level, and a lower level.

1. Press **Page Down** until you get to the lowest level.
2. Edit the `windowHeight` attribute.
3. Press **Page Up** to go to a higher level.



The **Inspector** window displays the following:

- `windowHeight` and `windowWidth`—Local edits will only affect the current selection.
- `floorHeight`—Local edits have been applied at a higher level, but the attribute is not currently selected.
- `windowHeight↓*↑`—Local edits have been applied at the current, lower, and a higher level.

#### **Note:**

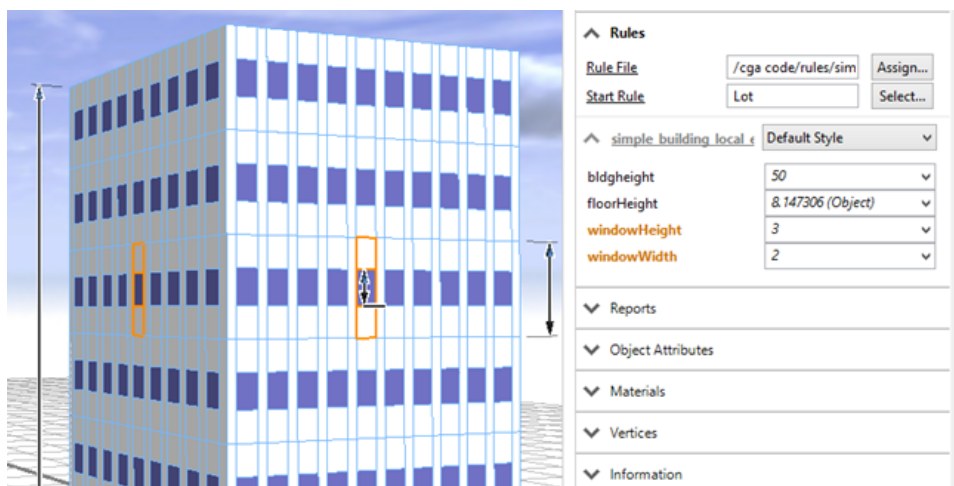
To select different level components, you may need to press **Page Up** or **Page Down** to move the selection level to match the component level you want to select. For example, if you are on the floor selection level and you want to select a window on another row, press **Page Down** to go to the lower level and then make your selection. Alternatively, you can deselect the object and select it again with the **Local Edits** tool to reset the selection.

## Use local edits on patterns

Use pattern selections when you want to select objects based on a pattern. Patterns are automatically detected.

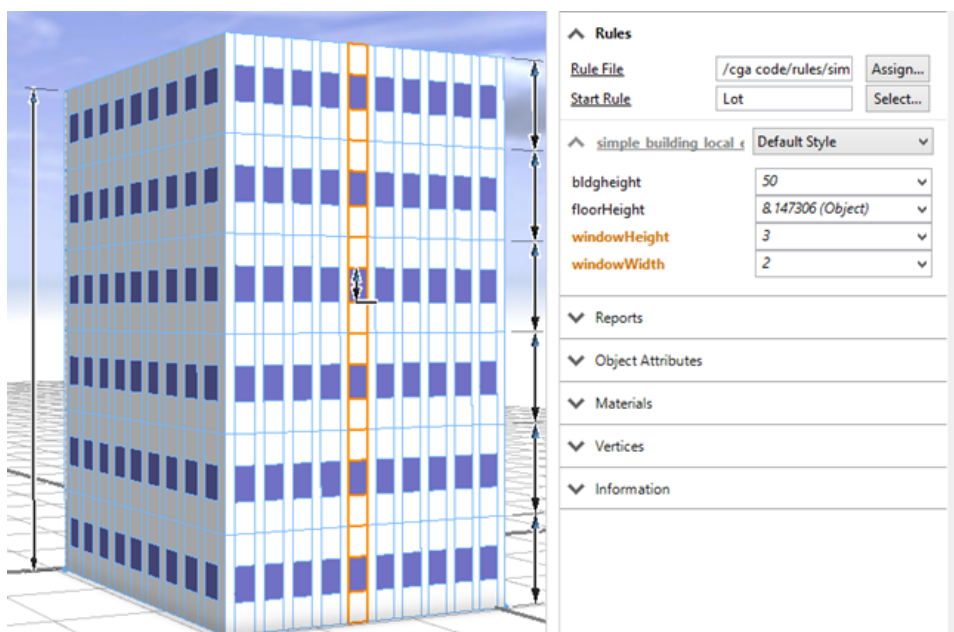
1. Click a window to make a selection at the lowest level.
2. Right-click the local selection and choose **Select next pattern** or press **End**.

The **Local Edit** tool estimates a pattern to your selection and makes multiple selections based on the structure of the [model hierarchy](#). In the example below, a pattern was selected of a single window in the same location on all four sides of the building.



3. Press **End** to select the next pattern.

This time, the **Local Edits** tool calculates a selection with windows in the same column.




4. Right-click and choose **Select previous pattern** or press **Home** to scroll back down the pattern options.

## Manage local edits

There are many way to manage local edits.

### Discover local edits

To see if local edits are applied to a specific selection, first click the **Local Edits** tool . When making a selection, the **Inspector** window now indicates whether local edits are applied to the attributes of this selection using the following symbols:

\*— This attribute has been locally edited at the current selection level.

↑— Local edits have been applied on a higher level than the current selection.

↓— Local edits have been applied on a lower level than the current selection.

To see edits applied on higher levels, use **Page Up** and **Page Down** to change the selection level. Stepping all the way up will highlight all performed local edits in the **Inspector** window with a ↓ symbol.

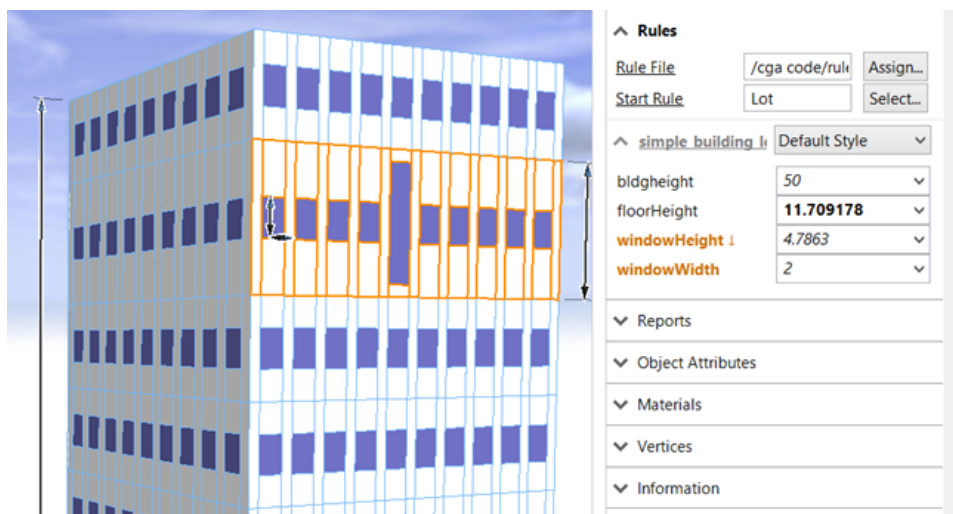
## Reset local edits

You can return to the default value of a specific attribute (without local edits applied). All local edits on the current and higher levels for this attribute will be removed.

1. Click the drop-down menu next to the attribute value.
2. Select **Rule Default**.



The locally edited windowHeight attributes at the current and higher levels are removed. Local edits at lower levels are not removed, as shown by the single taller window in the middle.



## Reset local edits for multiple attributes

To revert multiple attributes at once, you can use the context menu while in the local edits mode. Right-click the local selection or right-click in the **Inspector** window to see the following options:

- **Reset highlighted local edits**—Revert all the selected attributes to the values without local edits.
- **Reset all local edits**—Revert all locally edited attributes to the values without local edits.

# Rule package

A rule package (\*.rpk) is a compressed package containing compiled CGA rule files (\*.cgb), plus all required referenced assets and data. Rule packages allow you to share procedural rules in a single file. For finding the package content, the rules are analyzed.



## Note:

You can create complex asset dependencies in CGA, so it cannot be guaranteed that all assets are included in every case. When in doubt, add the files manually.

## Create a rule package

To create a rule package, complete the following steps. You can either store rule packages locally on your machine or publish them to ArcGIS Online. For more information, see [ArcGIS Online and ArcGIS Enterprise overview](#).

1. Locate the CGA rule in the **Navigator** window.
2. Right-click the CGA file and choose **Share As**.  
The **Rule Package** dialog box appears.
3. Choose **Save package to file** and set the file path.
  - Click **Upload package** and specify a name for the rule package.
  - For **Save package to file**, set a file path.
4. Check **Include CGA source code** to include readable text files.
5. Click **Item Description**.  
You can add meta information such as a summary, tags, and a description to the rule package.
6. Click **Additional Files**.  
You can remove proposed files and folders. You can also add new ones to enforce the packaging of specific assets.
7. Click **Sharing** (ArcGIS Online only).  
You can specify with whom you want to share the rule package.
8. Click **Analyze** to validate the rule package.  
If any issues are discovered, an error is reported. You must validate and resolve all errors before you can save the rule package to disk or share it to ArcGIS Online.
9. Click **Share** to upload or save the rule package.



## Note:

Keep the following in mind when you publish to ArcGIS Online:

- If you aren't already signed in, you will be prompted to sign in.
- If the rule package already exists on the portal, it will be updated with the new version and will keep the same ID.
- Rule packages can be uploaded to other ArcGIS portals as well. See [Share data on a different portal](#).

# Importing

# Import data

You can import data into CityEngine in the following ways:

- [Drag to folder](#)—Drag a file into a folder in the **Navigator** window to add it to your workspace. Drag it into the **Viewport** window to add it to the current scene. In this step, the import options defaults are used. These are suitable for most cases. This is the beginner import method.
- [Import dialog box](#)—Use the import dialog box for batch import and when default settings need to be adjusted. This is the advanced import method.
- [From ArcGIS Online](#)—Use the **Navigator** window to connect to your ArcGIS Online (or portal) account and download data into a project.
- [From the Get Map Data tool](#)—Import an extract of Esri world terrain and satellite imagery. Building footprints and street networks from OpenStreetMap (OSM) can be imported directly into the current scene.

See [Import a file](#) to learn how to import data into a [project](#).

# Import by drag and drop

You can import data by dragging a file.

- 1. Drag a file into a folder in the **Navigator** window to add it to your workspace.
- 2. Drag a file into the **3D Viewport** window to add it to the current scene.  
In this step, defaults are applied that are suitable for most cases.

## File types

File type	Description
DAE (COLLADA)	Imports as a static model
DWG (Autodesk)	Imports as a static model
DXF (AutoCAD)	Imports as shapes or graph segments
FBX (Autodesk)	Imports as a shape or static model
FGDB (Esri File Geodatabase)	Imports all supported layer types as shapes, point shapes, or graph networks
glTF (Khronos Group)	Imports as a shape or static model
IFC (buildingSMART)	Imports as a static model
KMZ/KML (Keyhole Markup Language)	Imports all referenced DAE objects as static models
OBJ (Wavefront)	Imports as a shape or static model
OSM (Open Street Map)	Imports all layers as shapes or graph segments
SHP (Esri Shapefile)	Imports as shapes, point shapes, multipatch shapes, or graph networks
USD (Universal Scene Description)	Imports as a shape or static model

## Drag to import image data

Dragging an image file from the **Navigator** window into a **3D Viewport** window opens the **Terrain** dialog box. Georeferenced data—such as .km1, .gdb, and .shp files—is placed at its georeferenced location. Data with no georeferenced information is placed at the drop spot, taking the data's center as its pivot.

# Import by dialog

In CityEngine, you can import files into a scene in the following ways:

- In the **Navigator** window, right-click and select **Import**.
- From the main menu, click **File > Import**.

In contrast to [dragging a file to](#) import, each import format presents a file dialog box with additional options.

Depending on the file format, the following table shows the resulting layer type when importing in CityEngine:

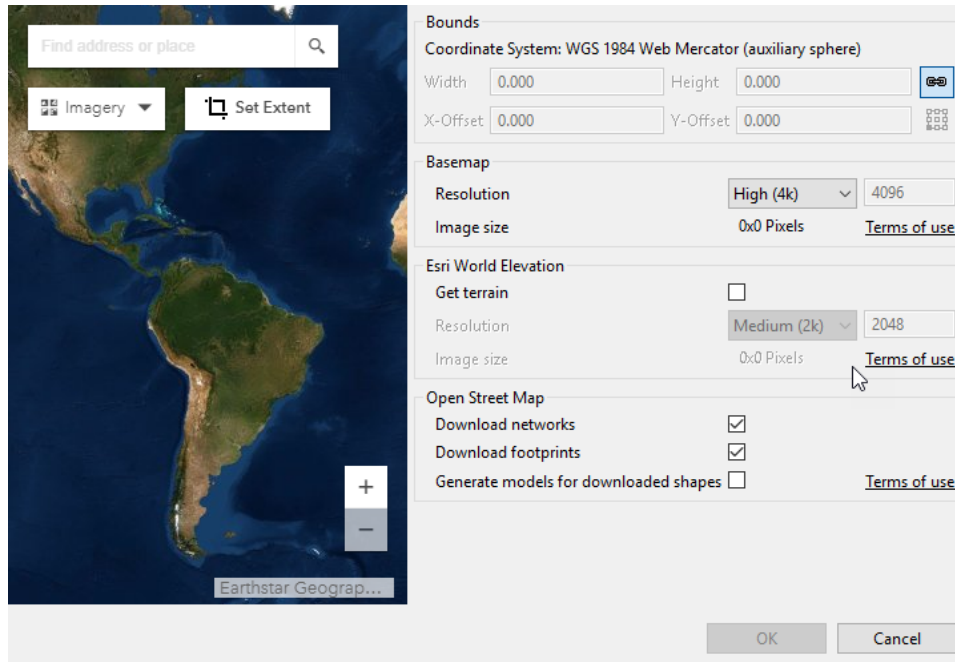
File format	Resulting layer type
<a href="#">DAE</a> (COLLADA)	Shape, static model
<a href="#">DWG</a> (Autodesk)	Shape, static model
<a href="#">DXF</a> (AutoCAD)	Shape, graph
<a href="#">FBX</a> (Autodesk)	Shape, static model
<a href="#">FGDB</a> (Esri File Geodatabase)	Shape, graph
<a href="#">glTF</a> (Khronos Group)	Shape, static model
<a href="#">IFC</a> (buildingSMART)	Shape, static model
<a href="#">KMZ/KML</a> (Keyhole Markup Language)	Static model
<a href="#">OBJ</a> (Wavefront)	Shape, static model
<a href="#">OSM</a> (Open Street Map)	Shape, graph
<a href="#">SHP</a> (Esri Shapefile)	Shape, graph
<a href="#">USD</a> (Universal Scene Description)	Shape, static model
Image file	Terrain, texture layer, shape texture

# Get Map Data

Get map data from ArcGIS Online and ArcGIS Enterprise to add context, such as streets, footprints, terrain, and basemaps, to a new or existing CityEngine scene.

To use the **Get Map Data** tool, [sign in](#) to your account and do one of the following:


- Choose **Build a City from Map Data** from the CityEngine welcome screen to create a scene from downloaded data.
- Select **File > Get Map Data** from the main menu to import downloaded data into an existing scene.

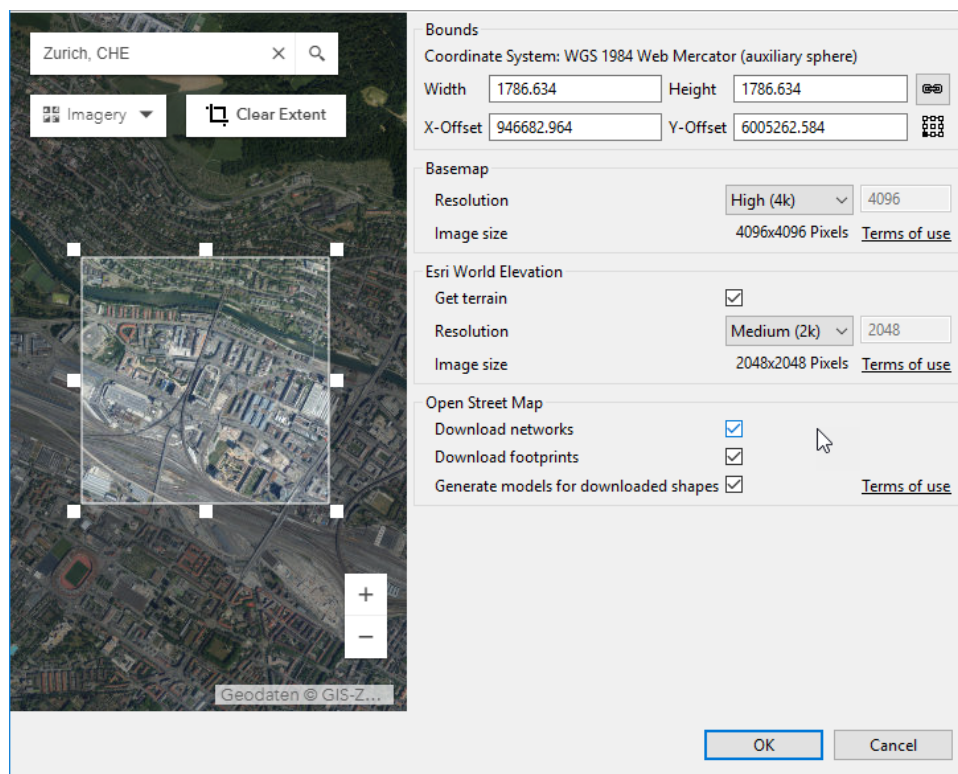


The Get Map Data dialog box is shown.

## Download data




To download map data into CityEngine, follow these steps:

1. Navigate to the area of interest.
2. Click **Set extent**  and specify the extent of the downloaded data.



The extent is set for Zurich, Switzerland.

You can also do the following:

- Change the size and location by adjusting the rectangle.
- Click **Clear Extent**  to remove the extent.
- Search for a location.
- Click **Select Basemap**  to change the type of basemap from the drop-down menu.
  - The **Get Map Data** tool allows you to select supported basemaps [configured](#) in your organization's portal basemap gallery.
  - When layers are available in a basemap such as **Dark Gray Canvas**, click **Layers**  to turn on and off individual layers, such as labels.

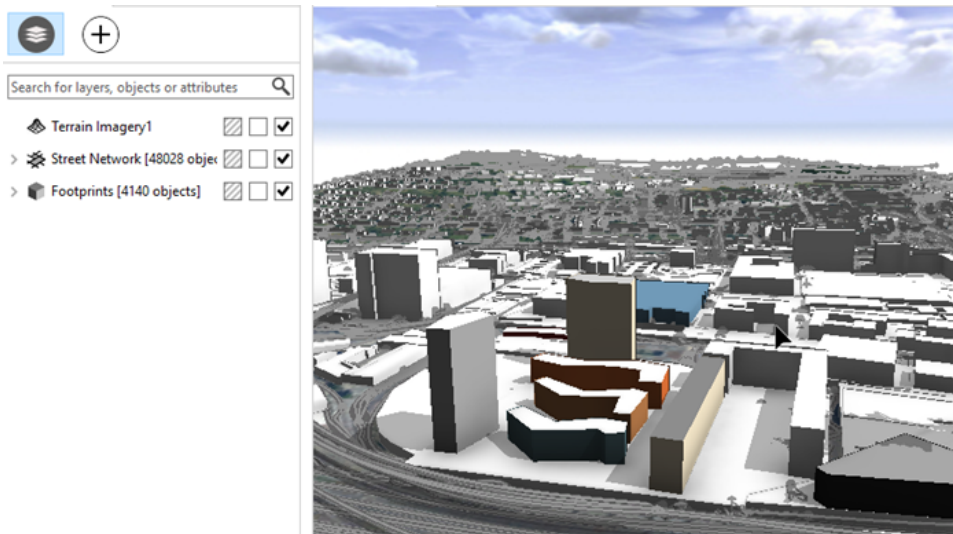
#### **Note:**

- There is a maximum limit for the extent.
- In an existing CityEngine scene, if an extent is farther than 500 kilometers from the scene, a red rectangle appears. CityEngine displays an error at the top prompting you to change the location of the extent or create a scene.

3. Specify the settings for the downloaded data.  
See the [Get Map Data settings](#) section below for more information.

4. Click **OK**.

- If you choose to download OpenStreetMap (OSM) data, an additional dialog box appears where you can configure the [OSM options](#). When the **OSM** dialog box appears, you can accept the default settings and click **Finish**, or click **Next** and review the settings of each option you checked.
- Map imagery, elevation, and OSM data, such as building footprints and streets, are downloaded according to your settings.
- All data is added to the scene. Default rules from ESRI.lib are assigned to the OSM data (see [OpenStreetMap](#) for more information).
- Data is saved to the /maps folder of the current project.



Zurich data is downloaded in a scene.



#### Tip:




The following information can help you improve the quality of the OSM network data imported into CityEngine:



- The graph networks are usually contained in the highway layer in the layer list of the **OSM** dialog box.
  - You can expand the highway layer and select or deselect the different network types. For example, you may want only the major streets in the scene.
  - After running the tool, the graph networks are merged into one graph layer in CityEngine.
  - You can import the `map.osm` file from the `project/maps` folder multiple times. This allows you to add the highway layer a number of times into CityEngine and create different graph layers to give you greater control of the OSM network data.
- After the OSM data is added to the scene, you may need to manually clean up the networks to improve the quality of the data.

See the [Import OSM data](#) section in Tutorial 4: Import streets for more information.

## Get Map Data settings

The **Get Map Data** tool includes the following settings:

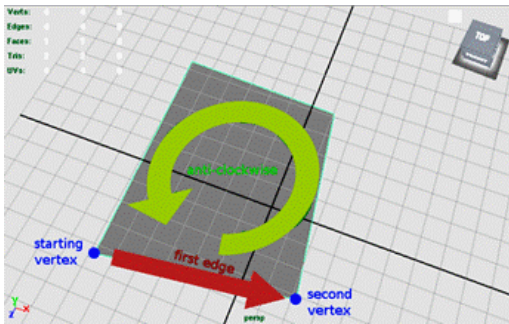
Bounds	<ul style="list-style-type: none"><li>• <b>Coordinate System</b>—The CityEngine scene coordinate system.<ul style="list-style-type: none"><li>▪ In a new scene, the <b>Get Map Data</b> tool sets the scene coordinate system to the best matching WGS84 UTM coordinate system.</li><li>▪ In an existing scene, the <b>Get Map Data</b> tool automatically converts the downloaded data to the current scene coordinate system.</li></ul></li><li>• <b>Width</b>—The width of the extent.</li><li>• <b>Height</b>—The height of the extent.</li><li>• <b>X-Offset</b>—The x-coordinate of the selected vertex of the extent.</li><li>• <b>Y-Offset</b>—The y-coordinate of the selected vertex of the extent.</li><li>• <b>Aspect ratio</b> —Maintains the aspect ratio when manually entering values.</li><li>• <b>Vertex location</b> —Specifies which vertex of the extent rectangle to reference for the x- and y-coordinates.</li></ul>
Basemap	<ul style="list-style-type: none"><li>• <b>Resolution</b>—The basemap has the following resolution options:<ul style="list-style-type: none"><li>▪ <b>Low (1k)</b>—Low resolution. The longer side of the extent is 1024 pixels.</li><li>▪ <b>Medium (2k)</b>—Medium resolution. The longer side of the extent is 2048 pixels.</li><li>▪ <b>High (4k)</b>—High resolution. The longer side of the extent is 4096 pixels.</li><li>▪ <b>Very High (8k)</b>—Very high resolution. The longer side of the extent is 8192 pixels.</li><li>▪ <b>Custom</b>—Manually set the longer side of the extent in pixels. The maximum size is 8192 pixels.</li></ul></li><li>• <b>Image size</b>—The dimensions of the extent in pixels.</li></ul> <p> <b>Note:</b> The ratio of the image size dimensions matches the ratio of the extent width and height. Consequently, the image size dynamically changes as the extent size changes.</p>

Esri World Elevation	<p>Esri provides a comprehensive elevation map of the world offering terrain at different scales. See <a href="#">Elevation Coverage Map</a> for more information.</p> <ul style="list-style-type: none"><li>• <b>Get terrain</b>—Include elevation data with the basemap.</li><li>• <b>Resolution</b>—The elevation has the following resolution options:<ul style="list-style-type: none"><li>▪ <b>Low (1k)</b>—Low resolution. The longer side of the extent is 1024 pixels.</li><li>▪ <b>Medium (2k)</b>—Medium resolution. The longer side of the extent is 2048 pixels.</li><li>▪ <b>High (4k)</b>—High resolution. The longer side of the extent is 4096 pixels.</li><li>▪ <b>Very High (8k)</b>—Very high resolution. The longer side of the extent is 8192 pixels.</li><li>▪ <b>Custom</b>—Manually set the longer side of the extent in pixels. The maximum size is 8192 pixels.</li></ul></li><li>• <b>Image size</b>—The dimensions of the extent in pixels.</li></ul> <p> <b>Note:</b> Elevation data is only available for subscribed users or those with an <a href="#">ArcGIS organizational account</a>.</p>
Open Street Map	<ul style="list-style-type: none"><li>• <b>Download networks</b>—Include street data when downloading a map.</li><li>• <b>Download footprints</b>—Include polygonal shapes such as building footprints.</li><li>• <b>Generate models for downloaded shapes</b>—Automatically generate models from the shapes after download. The models are generated after the <b>Get Map Data</b> tool applies the following <a href="#">ESRI.lib</a> rules to the shapes:<ul style="list-style-type: none"><li>▪ Networks—<code>Street_Modern_Standard.cga</code></li><li>▪ Footprints—<code>Building_From_OpenStreetMap.cga</code></li></ul></li></ul> <p> <b>Caution:</b> You may experience long download times if the extent is too large.</p>

# Prepare data for import

When using imported polygons as initial shapes for building generation, consider the following when creating the source data in a third-party application:

- Polygons that are drawn counterclockwise result in shapes facing upward. This is the desired orientation.
- The first edge of a polygon is interpreted as the front of the generated building. This is useful when you want to create different facades for the front, side, and back of a building.



*A polygon's edge orientation is shown.*

- Depending on the file format, CityEngine interprets group, entity, or object names as names for start rules. Consequently, you must group the meshes and assign names that are valid CGA shape grammar identifiers (no special characters, space, and so on).
- Some tools have different units for exporting. Review the exported file in a text editor to see the actual dimensions. Most import wizards allow you to apply a uniform scaling of the imported data.

# Import DAE (COLLADA)

The DAE format is used to exchange data between interactive 3D apps. It is based on COLLADA, which is an open-standards 3D geometry exchange format based on XML. It is mainly used in conjunction with [KML/KMZ](#) as an embedded container for georeferenced 3D models. Consider DAE a legacy format for all other use cases, as it lacks support for modern materials (for example, PBR) as well as import and export support in 3D apps. [glTF](#) is the recommended alternative.

## Import options

For DAE import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select a .dae file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes is created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry is automatically aligned to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object. <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>

# Import DWG (Autodesk)

Autodesk DWG is a proprietary 3D geometry format native to Autodesk AutoCAD. Although it doesn't have modern material support, it is widely used in the BIM and CAD industries.

 **Note:**

The Autodesk DWG importer only imports 3D aspects of the file. It silently drops all 2D elements. In Autodesk AutoCAD, you can convert unsupported objects (such as polylines) using one of the following commands: AECTOACAD, CONVTOMESH, CONVTOSOLID, or CONVTOSURFACE.

## Import options

For DWG import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select a .dwg file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes are created from the provided (if available, textured) polygons and are ready to be used with CGA rules.
Align to terrain	When checked, the geometry automatically aligns to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object: <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>

# Import DXF (AutoCAD)







AutoCAD DXF (Drawing Exchange Format) is a legacy format originating in the CAD industry to exchange 2D vector data. CityEngine imports the following entity types for DXF:

- Shapes—Circle, LwPolyline (must be closed), Polyline (must be either closed or polyface mesh), and Insert
- Graph segments—Line, Arc, Circle, Polyline, LwPolyline, and Insert

DXF import allows you to browse layers and individual entities contained in the .dxf file. You can select the entities to import by dragging them.


## Import options

For DXF import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a dialog box and select a .dxf file to import. After opening a file, the DXF entities are automatically classified as graph segment or shape and moved to the containers on the right.
Offset	<p>In the x- and y- offset fields, you can specify an offset to translate the data before importing. This can be useful to center the imported data and to avoid floating point precision problems. Click the <b>Center</b> button to automatically compute the offset of the selected DXF entities.</p> <p> <b>Note:</b> When changing the set of objects in the entity listings, the offset may also change. If this occurs, click <b>Center</b> again.</p>
Scale	Scale is the size factor that is applied to the selected entities.
Entity Listings	<p>In the container on the left, all entities contained in the .dxf file are listed. Entities that can be imported as shapes can be moved to the container at the lower right. All entities in this container are imported as shapes. Entities can be moved between and inside containers by dragging them.</p> <p> <b>Note:</b> Each entity is marked with an icon on the left indicating whether it can be imported as a graph segment, shape, or both.</p> <ul style="list-style-type: none"><li>• Graph only </li><li>• Shape only </li><li>• Graph or shape </li><li>• Entity cannot be imported </li></ul> <p>To delete entities on the right, click <b>Del</b>.</p>

## Graph settings

The following graph settings are available:

Option	Description
<b>Run Generate Bridges Tool after Import</b>	When enabled, the <a href="#">Generate Bridges</a> tool is run on a subsequent wizard page.
<b>Run Simplify Graph Tool after Import</b>	When enabled, the <a href="#">Simplify Graph</a> tool is run on a subsequent wizard page.
<b>Run Graph Cleanup Tool after Import</b>	Depending on the DXF data, it may be necessary to clean up the graph segments after import. If enabled, the <a href="#">Cleanup Graph tool</a> is run on a subsequent wizard page.
<b>Create Street/Intersection Shapes from Graph</b>	When enabled, the <a href="#">shape creation</a> parameter of the graph nodes and segments is enabled and street shapes are created.
<b>Create Block/Lot Shapes from Graph</b>	<p>When enabled, the shape creation parameter of potentially created street blocks is enabled and shapes are created.</p> <p> <b>Note:</b> Presets can be saved and applied.</p>

# Import FBX (Autodesk)

Autodesk FBX is a proprietary 3D geometry exchange format with a freely available SDK. It has gained popularity as the main exchange format for the Unreal and Unity game engines and related AR/VR applications. It currently lacks modern material support (for example, PBR) and metadata transport. For use cases in which AV/VR applications are not relevant, import using [glTF](#) instead.

## Import options

For FBX import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select an .obj file to import.
Import as static model	When checked, the file will be imported as is and will not be modifiable by CGA rules. Otherwise, start shapes will be created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry will be automatically aligned to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object. <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>


# Import FileGDB (Esri File Geodatabase)

The Esri File Geodatabase (FileGDB) format is a file-based database for vector and raster data. It is identified by the suffix `.gdb`, for example, `myDatabase.gdb`.

It supports many GIS data types such as points, lines, polygons, 3D geometry (multipatch), raster, and so on. It is the recommended way to exchange GIS data between Esri applications such as CityEngine and ArcGIS Pro.


## Import options

For FileGDB import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a dialog box and browse to the <code>.gdb</code> directory to import.
Layer Listing	<p>If the file entry contains a valid <code>.gdb</code> extension, the upper part of the wizard page shows the layers available to import. These are the available columns in the FileGDB inspector:</p> <ul style="list-style-type: none"><li>• <b>Layer</b>—The name of the layer. Check the check box to import the layer.</li><li>• <b>Alias</b>—An additional name used to reference the data; it is often the more commonly known name.</li><li>• <b>Type</b>—If the layer is a feature class, its geometry type is displayed. Otherwise, the layer type is displayed. A list of supported layer types is below.</li><li>• <b>Count</b>—If a feature class or a table, the number of rows is displayed.</li><li>• <b>Readable</b>—A layer is readable (can be imported) if it is a feature class with a non-zero row count and has a supported coordinate system. If you hover over the Layer error icon , a ToolTip describes why the layer is not readable.</li><li>• <b>CS Authority</b>—Displays the coordinate system EPSG authority ID.</li><li>• <b>CS Description</b>—Displays the coordinate system description.</li></ul>

The following layer types are supported:

- Point
- Polygon
- Polyline
- Multipatch (with textures)
- Table (indirectly, if connected through relationship classes)
- Relationship classes (indirectly)

Each FileGDB layer with geometry (also called a feature class) is imported as a separate CityEngine layer. Layer types that are not supported are marked with the Layer error icon  and are not imported.

**Note:**

If FileGDB data is imported into a new scene without a coordinate system, the scene inherits the coordinate system of the first imported layer. Subsequent layers with differing coordinate systems are reprojected.

Ensure that the input feature classes use the same coordinate units in the planar and up directions. CityEngine does not support separate units for the planar and up directions.

## Graph options


The following Graph options are available:

Option	Description
<b>Run Generate Bridges Tool after Import</b>	When enabled, the <a href="#">Generate Bridges</a> tool is run on a subsequent wizard page.
<b>Run Simplify Graph Tool after Import</b>	When enabled, the <a href="#">Simplify Graph</a> tool is run on a subsequent wizard page.
<b>Run Graph Cleanup Tool after Import</b>	Depending on the FileGDB data, it may be necessary to clean up the graph segments after import. When enabled, the <a href="#">Graph Cleanup tool</a> is run on a subsequent wizard page.
<b>Create Street/Intersection Shapes from Graph</b>	When enabled, the <a href="#">shape creation</a> parameter of the graph nodes and segments are enabled and street shapes are created.
<b>Create Block/Lot Shapes from Graph</b>	When enabled, the shape creation parameter of potentially created street blocks are enabled and shapes are created.

## Feature attribute options

The following feature attribute options are available:

Option	Description
<b>Import and map attributes</b>	<ul style="list-style-type: none"> <li>When enabled, all nongeometry attributes of a feature are also imported.</li> <li>This mapping controls the width of the street shapes generated from the graph center lines. In the default behavior, the object attribute <code>width</code> is used to determine the resulting street width and defaults to 8 if no object attribute is found.</li> </ul> <p><b>Note:</b> The function code can be edited after import in the inspector when selecting the imported layer.</p> <p>The default mapping code can be edited by changing the CGA code in the <code>gdb.ceattr</code> file, in the <code>/ce.lib/rules/</code> folder.</p>

Option	Description
<b>Import database scheme and relationships</b>	<p>When enabled, attributes from tables that are connected to a selected feature class by relationship classes are also imported and assigned to the imported shapes as object attributes.</p> <p> <b>Note:</b> Each object attribute keeps its FileGDB data type information (including domain) and original related table. This information can be used in the <a href="#">FileGDB exporter</a> to re-create feature classes, relationship classes, and tables.</p>
<b>Import textures</b>	<p>If the feature class is of type Multipatch, its textures are also imported and assigned to the scene shapes. Each texture is extracted from the feature class and saved as a .jpg or .png file (in case of transparency). These new texture files are placed in a new data/[FileGDB name without extension]-data project folder.</p>
<b>Use selection query and spatial envelope</b>	<p>When enabled, an attribute selection query and envelope can be used to reduce the number of imported features from each selected feature class. The layer ToolTip lists the available field names, aliases, and data types.</p>
<b>Select * Where</b>	<p>To obtain a list with all layer fields, hover over the layer name. Use these names to build an SQL query for filtering the imported shapes, for example: <code>SELECT * WHERE edits = 'yes'</code>.</p>
<b>Selection Envelope</b>	<p>Set the size (width, height) and reference point (X-offset, Y-offset) of the selection envelope.</p>

## FileGDB object attributes

Consider the following when working with FileGDB object attributes:

- Attributes of features are imported along with the feature. After a successful import, the attributes appear in the **Inspector** window on the **Object Attributes** tab.
- Feature attributes imported through related tables are prefixed with the name of the related table followed by an underscore.
- A set of imported shape attributes is displayed in the **Inspector** window. The shape inspector displays any array attributes resulting from importing related fields in italics, and you can select **Edit Table** from the drop-down menu.
- Use **Edit Table** to edit array attributes of one or more shapes in one place.
- Once a CGA rule file with matching attributes is assigned to this shape, the matching object attributes are connected to the rule file and control the generation of models. See [Work with object attributes](#) for more information.

# Import glTF/glb (Khronos Group)

glTF/glb (version 2.0) is a JSON-based 3D geometry delivery format. It supports modern materials (PBR) and geometry instancing and is supported by many desktop and web apps. It is a recommended exchange format for new projects.

The glb file format variant is a binary form of glTF that inlines textures instead of referencing them as external images.

For details, see the [KhronosGroup GitHub page](#).

## Import options

For glTF import, the following options are available:

Option	Descriptions
File	Click <b>Browse</b> to open a file dialog box to select a .gltf or .glb file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes are created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry is automatically aligned to the terrain.
Scale	Scale is the factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object: <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>

# Import IFC (buildingSMART)

buildingSMART IFC is an open-source 3D geometry exchange format to describe building elements. It is mainly used in the BIM and CAD industries to store project information for design, procurement, and construction phases as well as reference for operations. CityEngine only supports version 2x3.

## Import options

The following import options are available for IFC:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select an <code>.ifc</code> file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes are created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry is automatically aligned to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object. <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>

# Import KML/KMZ (Keyhole Markup Language)

Keyhole Markup Language (KML) is an XML-based exchange format used to describe GIS features on the globe. It supports points, lines, and polygons and can reference 3D models through COLLADA. It is used to produce data for ArcGIS Earth.

KMZ is a compressed version of KML in which all referenced COLLADA and texture files are contained in one zipped file.

## Models in CityEngine



### Note:

CityEngine supports only placemarks defining a model, which in turn refer to a COLLADA DAE model.

The following model attributes are read: `name`, `altitudeMode`, `Location`, `Orientation`, and `Scale`. In a model, the georeference is stored in the `Location` attribute consisting of longitude and latitude in degrees and altitude in meters. The geographical coordinate system is WGS84.



### Note:

If you click the **Finish** button and the current CityEngine scene's coordinate system is not yet set, a dialog box appears suggesting the UTM zone corresponding to the given longitude.

## Import options

No import options are available for KMZ/KML. To import KMZ/KML data, click **Browse** and select a `.kmz` file, a `.km1` file, or a folder containing one or more `.km1` files. When importing `.kmz` files, only a single file can be imported at once.



### Note:

You can also drag either a `.kmz` files or a folder containing a `.km1` file and associated files into the CityEngine workspace from the file manager (for example, File Explorer) and import it from there. KMZ/KML import works in the CityEngine workspace.

# Import OBJ (Wavefront)

Wavefront OBJ is a text-based legacy 3D model exchange format. Despite its limitations in efficiency and features (for example, no modern material support or geometry instancing) it remains popular due to its simple syntax and manual editability.

## Import options

OBJ import options are described in the following table:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select an <code>.obj</code> file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes is created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry is automatically aligned to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object. <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>



# Import OSM (OpenStreetMap)

An .osm file is XML based and typically used to export an extent of the OpenStreetMap (OSM) GIS service into other applications. These files typically contain georeferenced descriptions of streets, blocks, parcels, and points.

OSM geometry can be classified in the following categories:


- **Nodes**—The dots between which segments are drawn.
- **Ways**—An ordered list of nodes, displayed as connected by line segments in the editor.
- **Closed Ways**—Closed ways are ways that go in a complete loop. They are used to describe areas such as buildings, parks, lakes, and islands.

OpenStreetMap is a continually evolving open standard with a strong community.

By default, ways and closed ways are converted into graph segments. However, if a closed way contains one of the amenity, area, boundary, building, geological, historic, land use, leisure, natural, place, shop, sport, or tourism tags it's converted into a shape with the respective symbol,  or  displayed next to it.

## Import options

For OSM import, the following options are available:

Option	Description
File	Click <b>Browse</b> to open a dialog box and select an .osm file to import.
Element Listing	<div>Lists the layers and OSM ways contained in the selected .osm file. Select the element to import.</div> <div> <b>Note:</b><ul style="list-style-type: none"><li>• .osm files may contain a large number of layers, of which the highway layer is typically used to create street networks.</li><li>• You can select multiple OSM layers per import session. Then, all ways that are converted into graph segments are merged into one graph layer. To generate several graph layers from one .osm file, repeat the process.</li></ul></div>
Select / deselect all	Select or deselect all layers.

## Graph options

Option	Description
Map OSM tags	If enabled, street and sidewalk widths are mapped from tags contained in the .osm file. See OSM tag mapping below.
Align Graph to terrain	If enabled, OSM streets are aligned to a user-defined terrain.
Align Shapes to terrain	If enabled, the OSM shapes are aligned to a user-defined terrain.
Run Generate Bridges Tool after Import	If enabled, the <a href="#">Generate Bridges</a> tool is run on a subsequent wizard page.

Option	Description
Run Simplify Graph Tool after Import	If enabled, the <a href="#">Simplify Graph</a> tool is run on a subsequent wizard page.
Run Graph Cleanup Tool after Import	Depending on the OSM data, you may need to clean up the graph segments after import. If enabled, the <a href="#">Cleanup Graph tool</a> is run on a subsequent wizard page.
Create Street/Node Shapes from Graph	If enabled, the <a href="#">shape creation</a> parameter of the graph nodes and segments is enabled and street shapes are created.
Create Street/Node Shapes for Tunnels	If enabled, the shape creation parameter honors the vertical level of streets. Therefore, tunnels will be created.

OSM tag mapping

If **Map OSM tags** is checked, the created layer contains the following layer attribute code:



The screenshot shows the 'Layer Attributes' window in ArcGIS CityEngine. The window has a title bar with a small icon and the text 'Layer Attributes'. Below the title bar is a text area containing code. The code is a function for mapping OSM tags to street and sidewalk widths. The code is as follows:

```
//-----
// Example OSM Tag Mapping

streetscale = 1 // street width scale factor

width = getObjectAttr("width")
lanes = getObjectAttr("lanes")

attr streetWidth = // street width dependeding on available attributes
  case width > 0 : width * streetscale
  case lanes > 0 : lanes * 3 * streetscale
  else : streetWidthByClass * streetscale * oneway

class = getObjectAttr("highway")

streetWidthByClass =
  case class == "primary" : 8
  case class == "secondary" : 7
  case class == "tertiary" : 6
  case class == "motorway" : 12
  case class == "trunk" : 11
  case class == "road" : 6
  case class == "residential" : 5
  case class == "footway" : 2
  case class == "cycleway" : 2
  case class == "steps" : 2
  else : 4

oneway = // oneway width correction
  case getObjectAttr("oneway") == "yes" : 0.5
  else : 1

sidewalkscale = 1 // sidewalk width scale factor

sidewalkWidth =
  case class == "primary" : 2
  case class == "secondary" : 2
  case class == "tertiary" : 2
  case class == "residential" : 2
  else : 0

attr sidewalkWidthLeft = sidewalkWidth * sidewalkscale
attr sidewalkWidthRight = sidewalkWidth * sidewalkscale
```

OSM tag mapping code is shown.

By default, an example function code maps common OSM tags to street and sidewalk widths. The function code is copied into the newly created street layer, and the street and sidewalk width parameters of the imported street segments are correctly mapped.

Street Parameters

shapeCreation	✓ Enabled
streetWidth	4 (map)
streetOffset	0
sidewalkWidthLeft	0 (map)
sidewalkWidthRight	0 (map)
precision	0.5
laneWidth	3.5

Object Attributes

connectionEnd	JUNCTION
connectionStart	JUNCTION
highway	track
name	Passeggiata fuori le mura
osm_id	2.8884763E7
surface	stone
tracktype	grade2
type	MAJOR

+ Add new object attribute...

Street parameters and object attributes are shown.

 **Note:**


- You can edit the function code after import in the **Inspector** window when selecting the street layer.
- You can edit the default mapping code by changing the CGA code in the `osm.ceattr` file in the `/ce.lib/rules/` directory.
- Presets can be saved and applied.


# Import SHP (Esri Shapefile)

Shapefile is an Esri legacy format used to describe georeferenced GIS features (points, lines, polygons, and limited multipatches). It has limited support for GIS attributes and should be considered superseded by [Esri FileGDB](#).

## Import options

For shapefile import, the following options are available:

Option	Description
File	<ul style="list-style-type: none"><li>Click <b>Browse</b> to open a dialog box and select an .shp file to import.</li><li>After opening a file, the dialog box detects the shape type of the file. The type is displayed in the header of the dialog box (for example, <b>Shape Type: POLYGON. Importing shapes</b>). The following types can be imported as shapes: Polygon, PolygonZ, PolygonM, Multipatch, Point, and PointZ. Other types are imported as graph segments.</li></ul> <p> <b>Note:</b></p> <ul style="list-style-type: none"><li>Shapefiles containing points can also be imported. In this case, a marker (0.1 x 0.1m quad) is created for each point.</li><li>Shapefile polygons containing negative polygons that cut holes into polygons are not supported and are imported as normal shapes instead of holes.</li></ul>
Coordinate System	<ul style="list-style-type: none"><li>The .prj file of the shapefile is read by SHP import. If successful, the corresponding coordinate system is displayed on the dialog box. Otherwise, a pop-up prompts you to choose a coordinate system.</li><li>Ensure that the input shapefiles use the same coordinate units in the planar and up directions. CityEngine does not support separate units for the planar and up directions.</li></ul>

Option	Description
Using Attributes from SHP file	<ul style="list-style-type: none"> <li>A .shp file has an accompanying .dbf file that contains associated attributes for elements. After importing the .shp file, these attributes appear in the <b>Inspector</b> window on the <b>Object Attributes</b> tab.</li> <li>To use these attributes with the CGA grammar, you must declare CGA attributes with matching names. A simple CGA rule file may look like this: <pre>attr height = 10 Lot --&gt; extrude(height)</pre> </li> <li>After assigning this rule file to the shapes, the height CGA attribute appears on the <b>Object Attributes</b> tab of the <b>Inspector</b> window.</li> </ul> <p> <b>Note:</b> The Source field is set to Object, which denotes that the CGA height attribute is controlled by the object attribute of the lot.</p>

## Graph options

The following graph options are available:

Option	Description
Run Generate Bridges Tool after Import	If enabled, the <a href="#">Generate Bridges</a> tool is run on a subsequent wizard page.
Run Simplify Graph Tool after Import	If enabled, the <a href="#">Simplify Graph</a> tool is run on a subsequent wizard page.
Run Graph Cleanup Tool after Import	Depending on the shapefile data, it may be necessary to clean up the graph segments after import. If enabled, the <a href="#">Cleanup Graph tool</a> is run on a following wizard page.
Create Street/Intersection Shapes from Graph	If enabled, the <a href="#">shape creation</a> parameter of the graph nodes and segments is enabled and street shapes are created.
Create Block/Lot Shapes from Graph	If enabled, the shape creation parameter of potentially created street blocks is enabled and shapes are created.

## Map shapefile attributes

If enabled, an imported graph layer contains the following layer attribute code:

### Layer Attributes

```
//-----
// Example OSM Tag Mapping

streetscale = 1 // street width scale factor

width = getFloatObjectAttr("width", false)
lanes = getFloatObjectAttr("lanes", false)

attr streetWidth = // street width depended on available attributes
  case width > 0 : width      * streetscale
  case lanes > 0 : lanes * 3.5 * streetscale
  else          : streetWidthByClass * streetscale * oneway

class = getStringObjectAttr("highway", false)

streetWidthByClass =
  case class == "primary"      : 8
```

Map shape attributes are shown.

This mapping controls the width of the street shapes generated from the graph center lines. In the default behavior, the width object attribute is used to determine the resulting street width and defaults to 8 if no object attribute is found.

You can edit the default mapping code by changing the CGA code in the shp.ceattr file in the /ce.lib/rules/ directory.

# Import USD (Universal Scene Description)

Use USD import to read [Universal Scene Description \(USD\)](#) assets as static models or start shapes. It supports .usd, .usda, .usdc, and .usdz file extensions and recognizes the USD mesh and cube node types. USD import can read scenes consisting of more than one .usd file; see the [USD export](#) file structure for more information. Additionally, for static models, if the USDPreviewSurface material type is present, they are displayed as [CityEngine PBR Material](#).

## Import options

USD import options are described in the following table:

Option	Description
File	Click <b>Browse</b> to open a file dialog box and select a .usd, .usda, or .usdc file to import.
Import as static model	When checked, the file is imported as is and cannot be modified by CGA rules. Otherwise, start shapes are created from the provided (if available, textured) polygons, ready to be used with CGA rules.
Align to terrain	When checked, the geometry automatically aligns to the terrain.
Scale	Scale is the size factor applied to the imported object.
Offset	<b>Offset</b> adds the specified offset to the imported object: <ul style="list-style-type: none"><li>• <b>Center</b>—Sets the offset such that the object is centered on the scene's origin.</li><li>• <b>Reset</b>—Sets the offset back to zero.</li></ul>

# Exporting

# Export data

When you export data from CityEngine, you can move it into other 3D authoring tools, publish it on ArcGIS Online, or transform it into a user experience on the web or in virtual reality (VR). Each layer type can be exported into a variety of data types.

- For the most common use cases for exporting in CityEngine, click **File > Export Models**. Use the **Export Models** dialog box to export CGA models, manually drawn shape models, static models, and terrain meshes into a variety of file formats.
- Click **File > Export** to access additional export methods, as well as the methods from the **Export Models** dialog box.

The following table lists the data types that can be exported from CityEngine layer types:

Layer type	Export data type
<a href="#">Model</a>	Mesh
<a href="#">Map</a>	Image, mesh
<a href="#">Graph</a>	Line
<a href="#">Shape</a>	Mesh

 **Note:**

- [Exporting projects](#) and [scenes](#) to share data with another CityEngine user or locally between workspaces is not covered in this section.
- [A 360 VR experience](#) is a type of export format that can include entire scenes and is independent of layer types.

# Export models

# Export a model

The following are the basic steps to export a model:

1. Select the objects to export in the **Viewport** window or **Scene Editor** window.
2. Click **File > Export Models** from the main menu or press **Ctrl + E**.
3. Select the file format.

## Note:

The model exporter is independent of the generated models in the current scene and can export large scenes. This means that you do not need to generate a scene before exporting a model.

## Supported formats

The supported 3D formats are described in the following table:

3D format	Description
<a href="#">ABC (Alembic)</a>	An open, interchange framework designed for VFX content production pipelines.
<a href="#">DAE (COLLADA)</a>	Loads a large number of DCC tools and rendering engines with support for asset instancing, layered textures, and file referencing.
<a href="#">DATASMITH (Unreal and Twinmotion)</a>	Provides simple data transfer from CityEngine into the Unreal Engine Editor and Twinmotion. It supports modern materials (PBR) as well as geometry instancing.
<a href="#">DWG (Autodesk)</a>	A proprietary 3D geometry format native to Autodesk AutoCAD. It doesn't have modern material support but is widely used in the BIM and CAD industries.
<a href="#">FBX (Autodesk)</a>	Provides export into Autodesk Maya, 3ds Max, MotionBuilder, and other DCC tools equipped with an FBX importer. It includes support for layered textures (multitexture) and per-texture UVW transformations (scale, translation, and rotation).
<a href="#">glTF (Khronos Group)</a>	A recent JSON-based 3D geometry delivery format. It supports modern materials (PBR) as well as geometry instancing and is supported by many desktop and web apps. Currently it is the recommended exchange format for new projects.
<a href="#">IFC (buildingSMART)</a>	An open-source 3D geometry exchange format used to describe building elements. It is mainly used in the BIM and CAD industries to store project information for design, procurement, and construction phases as well as reference for operations. CityEngine only supports version 2x3.
<a href="#">OBJ (Wavefront)</a>	Transfers models with solid coloring or single texture layers to a DCC tool or rendering engine. The material definitions are exported into accompanying .mtl files. It does not support instancing.

3D format	Description
<a href="#">USD (Universal Scene Description)</a>	Stores large CityEngine scenes with minimal loss of information, which makes it suitable as an interchange format for further processing in VFX pipelines.
<a href="#">VOB (e-on VUE)</a>	A native object format for polygonal meshes of e-on software's VUE product. CityEngine supports VUE 8.5 and later. It is recommended that you use the latest version.

The supported 3D GIS formats are described in the following table:

3D GIS format	Description
<a href="#">FileGDB (Esri file geodatabase)</a>	A common file format for GIS workflows. Export multipatch, multipoint, point, and polyline geometry types into a file geodatabase. Textures are supported.
<a href="#">KMZ/KML (Keyhole Markup Language)</a>	Exports into georeferenced earth browsers such as ArcGlobe and ArcGIS Earth. The models can be shared, for example, in Google Warehouse.

The supported ArcGIS Online formats are described in the following table:

ArcGIS Online format	Description
<a href="#">MSPK (mobile scene package)</a>	A mobile scene package (.msp file) is a custom, web-optimized file type that can be used to share complete scenes on the ArcGIS Platform.
<a href="#">SLPK (Esri scene layer package)</a>	A scene layer package (.slpk file) is a custom, web-optimized format that can be shared and published on ArcGIS Online and viewed with Scene Viewer.
<a href="#">3VR (360 VR Experience)</a>	Exports panoramic photos of CityEngine scenes.

The supported CGA Material format is described in the following table:

Material format	Description
CGA Material	<p>The material descriptions of a CGA model are collected and written into individual subdirectories. Each subdirectory contains all the material—including the used textures—which can be copied into a folder of assets, for example. The name of the subdirectory and the .cgamat file is the <a href="#">material.name</a> attribute but is made unique if there are several materials with the same name.</p> <p>The &lt;matName&gt;.cgamat file is in CSV format, as is the result of the CGA function: <code>getMaterial(used, changed)</code>.</p>

The supported custom export format is described in the following table:

Custom export format	Description
<a href="#">Python (script-based export)</a>	Allows Python commands to be arbitrarily run during batch export.

## General export options

The general options can be used for all export file formats. Certain formats contain additional options, which are described in the corresponding subsection below.


 **Note:**

- In the Export Wizard, each export option displays a ToolTip with a description.
- To compare the material and shading functionality of the various formats, see the [Format feature comparison](#) section below. .

### General options

The general options are described in the following table:

Option	Description
Output Path	The path to the export location. The path must exist.
Base Name	The base name of the exported files. A suffix will be appended depending on the other export option settings.
Export Geometry	<ul style="list-style-type: none"><li>• <b>Models with Shape Fallback</b>—If the model generation fails, the start shape geometry is exported. This is the default.</li><li>• <b>Models</b>—The shape is ignored if the model generation fails.</li><li>• <b>Shapes</b>—Only the start shape geometry is exported.</li></ul>
Terrain Layers	<ul style="list-style-type: none"><li>• <b>Do not export any terrain layers</b></li><li>• <b>Export all visible terrain layers</b></li><li>• <b>Export all selected terrain layers</b>—This is the default</li><li>• <b>Export all terrain layers</b></li></ul>
Simplify Terrain Meshes	The <a href="#">reduceGeometry operation</a> is used to simplify the terrain geometry prior to export. This can take a long time, especially for higher resolution terrains.

Option	Description
<b>Terrain Mesh Resolution</b>	<p>Determines the number of terrain mesh vertices in the U and V directions to export. Depending on the quality of the terrain resolution, you can have the following options:</p> <ul style="list-style-type: none"> <li>• <b>Original</b>—Takes the maximum resolution of the selected terrains for each direction and applies the value to all of them (only available if a terrain is selected).</li> <li>• <b>4K</b>—4096 x 3079 pixels.</li> <li>• <b>2K</b>—2048 x 1540 pixels.</li> <li>• <b>1K</b>—1024 x 770 pixels.</li> <li>• <b>Custom</b>—Manually set the resolution.</li> <li>• <b>Aspect ratio</b> —Maintains the aspect ratio when manually entering values.</li> </ul>

## Granularity options

The granularity options are described in the following table:

Option	Description
<b>Memory Budget</b>	<p>The maximum memory consumption in megabytes.</p> <p>The geometry data is written to a single file if <b>Memory Budget</b> is zero. Otherwise, a new file is written to disk whenever the memory budget is reached.</p> <ul style="list-style-type: none"> <li>• The geometry size is measured on the raw meshes before any optimization. The file size is typically smaller than the setting.</li> <li>• The check is performed per input shape not per rule output shape; that is, very complex rules may generate more data than the limit setting.</li> </ul>
<b>Mesh Granularity</b>	<ul style="list-style-type: none"> <li>• <b>Do not merge any meshes</b>—No meshes are merged; each mesh is optimized individually.</li> <li>• <b>Merge meshes by material</b>—All meshes with the same material properties are merged and optimized.</li> <li>• <b>Reuse asset instances, merge generated meshes by material</b>—Inserted assets and meshes (see the <a href="#">CGA insert operation</a>) are preserved and instanced. Meshes generated by the grammar are merged by material.</li> </ul>
<b>Feature Granularity</b>	<ul style="list-style-type: none"> <li>• <b>One Feature Per Shape</b>—One feature per start shape is created in the package.</li> <li>• <b>One Feature Per Leaf Shape</b>—One feature per leaf shape is created in the package.</li> </ul>

## Geometry options

The geometry options are described in the following table:

Option	Description
<b>Vertex Normals</b>	<ul style="list-style-type: none"> <li>• <b>Write vertex normals</b>—Vertex normals are written when they're present on the mesh.</li> <li>• <b>Do not write any vertex normals</b>—Vertex normals are not written even if they're present on the mesh.</li> <li>• <b>Write face normal if vertex normal is missing</b>—Face normals are used for vertex normals if the mesh does not contain vertex normals, .</li> <li>• <b>Always write face normal</b>—Existing vertex normals present on the geometry are ignored and replaced by face normals.</li> </ul>
<b>Normals Indexing</b>	Choose between indexed polygon normals or separated normal copies per polygon.
<b>Texture Coordinates</b>	<ul style="list-style-type: none"> <li>• <b>Do not write any UVs</b>—No texture coordinates will be exported.</li> <li>• <b>Only write first layer of UVs</b>—Only the first layer of texture coordinates (uv-set 0 corresponding to the color map) is exported. For more information, see <a href="#">Texturing: Essential knowledge</a>.</li> <li>• <b>Write all UV layers</b>—All layers are exported.</li> </ul>
<b>Local Offset</b>	<p>Local offset is only active when <b>File Granularity</b> is set to <b>One File per start shape</b>, as it is computed for each individual shape.</p> <ul style="list-style-type: none"> <li>• <b>None</b>—Local offset is not used.</li> <li>• <b>Model Centroid</b>—The centroid of the generated model is used as the offset.</li> <li>• <b>Model Centroid Bottom</b>—The centroid projected on the bottom bound of the generated model is used as the offset.</li> <li>• <b>Shape Centroid</b>—The centroid of the shape is used as the offset.</li> <li>• <b>Shape Centroid Bottom</b>—The centroid projected on the bottom bound of the shape is used as the offset.</li> </ul>
<b>Global Offset</b>	The global offset for generated geometry. Set the offset for the x, y and z axes (Cartesian coordinate values).
<b>Vertex Precision</b>	Use these values to reduce the floating point precision of different geometry data prior to the file output (used to reduce file size in OBJ).
<b>Merge within precision</b>	Coordinates (vertices, normals, and texture coordinates) that have a Euclidean distance equal to or less than precision are merged and their corresponding indexes are updated.

Option	Description
<b>Triangulate Meshes</b>	All faces are converted to triangles.
<b>Faces with holes</b>	<ul style="list-style-type: none"> <li>• <b>Write as holes</b>—The hole information is written. This option is only available for some formats.</li> <li>• <b>Triangulate faces with holes</b>—All faces containing holes are triangulated.</li> <li>• <b>Discard holes</b>The hole information is ignored.</li> <li>• <b>Convert holes to faces</b>Holes are converted to faces (coplanar faces are created).</li> </ul>


## Materials options

The materials option is described in the following table:

Option	Description
<b>Include Materials</b>	When enabled, material definitions (for example, diffuse color) and textures are included in the export. Otherwise, the default material is referenced (if required by the format).

## Texture options


Texture options are described in the following table:

Option	Description
<b>Create Texture Atlas</b>	<p>Texture atlases that combine a set of textures into one are created, reducing the number of textures and materials. All textures except color maps are removed. For more information, see <a href="#">CGA material attribute</a>.</p> <p> <b>Note:</b> In case of repetition (that is, UV coordinates outside [0,1]), textures can not be put into an atlas.</p>
<b>Texture Atlas Max Dimension (2^n)</b>	Specifies the maximum dimension of the texture atlases in powers of two pixels, for example, a value of 11 gives a 2048x2048 atlas.
<b>Texture Atlas Border</b>	A border of pixels that repeat the texture content is added to prevent incorrect scaling of artifacts.

## Advanced options

Advanced options are described in the following table:

Option	Description
<b>File Type</b>	Choose binary- or text (ASCII)-based file creation.
<b>Embed Textures</b>	Reference the file path (the default) or embed the texture files in the binary file.

Option	Description
<b>Shape Name Delimiter</b>	<p>The specified character is used to resolve duplicate shape names. For example, a delimiter of "_" resolves {"shape", "shape", "shape"} to {"shape", "shape_1", "shape_2"}. Existing suffixes such as "_1" are recognized.</p> <p> <b>Note:</b> To use the name resolver manually for a selection of shapes, click <b>Edit &gt; Make Names Unique</b>.</p>
<b>Existing Files</b>	<ul style="list-style-type: none"> <li>• <b>Overwrite existing files</b>—No file check is performed before writing the geometry files. All files are overwritten.</li> <li>• <b>Skip existing files</b>—Existing files (geometry, material, and texture files) are not overwritten.</li> </ul>

## Additional options

One addition option is described in the following table:

Option	Description
<b>Script</b>	A workspace path to a python script for additional python operation parallel to export.

## Format feature comparison

The features in the table below reference the output of CityEngine, not the theoretical features of the individual format. Compare this list with the known interoperability limitations and recommendations in the [Format recommendations](#) section of the Model export application notes topic.

Format	Instancing	Shader	Multitexture	Tex Trafo	Triangulation	Referencing
ABC	Yes	No 2	Yes	Yes	Yes	No
DAE	Yes	No	No	Yes	Yes 1	Yes
DWG	No	No	No	No	No	No
DATASMITH	Yes	Yes	Yes	No	Yes 4	No
FBX	Yes	No	Yes	Yes	Yes	No
FileGDB	No	No	No	No	Yes	No
glTF	Yes	No	No	No	No 3	No
KMZ / KML	Yes	No	No	Yes	Yes 1	Yes
OBJ	No	No	No	No	Yes	No
SLPK	No	No	No	No	No	No
USD	Yes	No	Yes	No	No	Yes

- 1. Triangulation is necessary when exporting to ArcGIS Earth. See the [Format recommendations](#) section of the Model export application notes topic for more information.

- 2. Alembic does not store shaders, but you can use CGA reports to write out a shader name as a property and bind it in a downstream tool.
- 3. The glTF format only allows triangulated geometry.
- 4. CityEngine always passes triangulated geometry to DATASMITH.

Feature types

Supported feature types are described in the following table:

Feature	Description
Instancing	Supports instancing of geometry. Typically, the assets are stored separately in an unmodified state from the nodes and transformation data.
Shader	Supports custom shader names.
Multitexture	Supports multiple texture channels.
Tex Trafo	Supports separate texture coordinate transformations independent from the texture coordinates stored in the meshes (for example, texture rotation).
Triangulation	Confirm whether the exporter supports optional triangulation on export.
Referencing	Confirm whether referencing of entities between files is supported. This allows the definition of a primary file to combine all scene elements.

# Export ABC (Alembic)

The CityEngine Alembic exporter exports geometry and metadata (materials, CGA reports, object attributes, and so on) into Alembic .abc files. Alembic is an open, high-performance interchange framework designed for VFX content production pipelines. See <https://github.com/alembic/alembic> for more details.

## Use cases

The Alembic exporter has been designed with focus on the following:

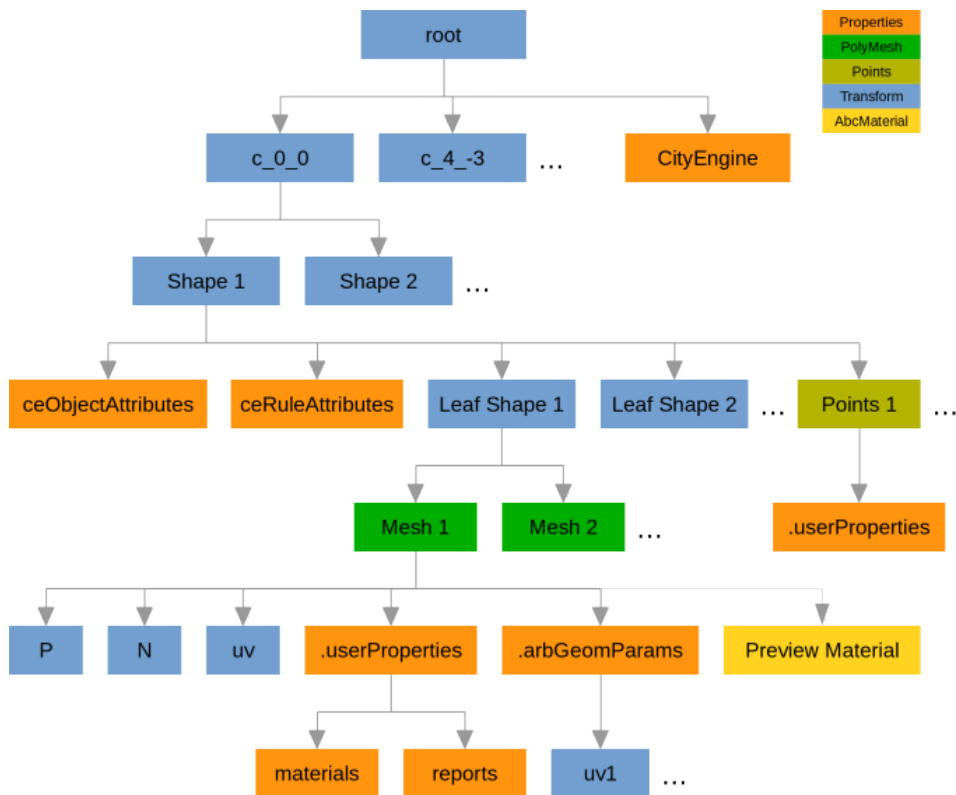
- Compatibility with established software and functionality such as SideFX Houdini Mantra, Autodesk Maya, Foundry Katana, Pixar Renderman, Solid Angle Arnold, and Chaos Group V-Ray.
- Export of very large models with complex rules while retaining shape granularity (no merging of geometry). This allows edits (for example, move and scale) on individual objects (for example, buildings) in downstream tools without re-exporting the whole scene.
- Preservation of instances, that is, inserted assets that are not touched by CGA operations, are exported as separate nodes. This allows limited editing (translate and scale) of these assets in downstream tools.
- Direct export of all material properties and CGA report values on each mesh. This allows customization of materials using CGA reports in downstream tools.

## Export options

The Alembic exporter shares most export options with the other formats. See the [general export options](#) for details. The only exception is the file name. Because of the way Alembic handles large files (multi GB), the exporter writes a single file per run.

## CityEngine Alembic node hierarchy

The Alembic node hierarchy that is created in CityEngine is shown in the following diagram:



The mappings from Alembic entities to CityEngine entities are described below (refer to the diagram above, from upper left to lower right).

- `c_0_0` transform nodes are cells of a sparse grid derived from the exported scene. The cell size can be adjusted in the export wizard.
- Shape transform nodes are created from CityEngine scene objects (initial shapes). The transformation stores the translation defined by the global or local offset options.
- The CityEngine property node is unique and receives the current username and CityEngine version.
- `ceObjectAttributes` and `ceRuleAttributes` property nodes are attached to the shape node and receive the object attribute values and rule attribute values of the corresponding shape.
- Leaf Shape transform nodes contain the geometry output of CGA leaf shapes. For uncached meshes (that is, geometry created by CGA), one node is created with the CGA leaf shape name. For instanced assets, one transform node is created for each mesh.
- Points nodes receive all single vertex geometries from a shape (for example, generated by `scatter()` or `comp(v)`). One points node for each unique material or report combination is created. Only reports are attached as user properties.
- Mesh nodes are created for each set of polygons with the same instance transformation (or none, in the case of uncached geometry) and the same material and CGA report values. The unique material or report combination is useful to customize materials with report values.
- P (points), N (normals) and uv0 (first texture coordinate set) transform nodes store the polygonal geometry.
- The `.userProperties` property node stores CGA materials and reports per mesh or points node.

- The .arbGeomParams property node stores additional texture coordinate sets (Alembic only supports one set on the mesh).
- The Preview Material AbcMaterial node stores the current material properties in the (experimental) [Alembic Preview Material](#) version 2.1 schema.

# Export DAE (COLLADA)

COLLADA is an open-standards 3D geometry exchange format based on XML. It is typically used in conjunction with KML and KMZ as an embedded container for georeferenced 3D models. It is considered a legacy format for all other use cases, as it lacks modern materials support (for example, PBR) and import and export support in many 3D apps. It is recommended that you use [glTF](#) in these cases instead.

## Export options

See the [general export options](#).

## CGA mapping to COLLADA

The following table lists the mapping of the main CityEngine elements to COLLADA elements.

### Note:

The instancing and multitexture behavior of COLLADA is similar to [FBX](#).

#### Whole scene or selection

```
<library_visual_scenes>
  <visual_scene id="VisualSceneNode">
    <node id="rootInst_lot1697" type="NODE">
      <instance_node url="./model_lot1697.dae#root_lot1697"/>
    </node>
  </visual_scene>
</library_visual_scenes>
```

#### Model or shape

```
<scene>
  <instance_visual_scene url="#VisualSceneNode"/>
</scene>
```

Leaf shapes (with references to assets and materials and optional transformation matrices)

```
<library_visual_scenes>
  <visual_scene id="VisualSceneNode" name="scene_lot1697">
    <node id="root_lot1697" type="NODE">
      <node id="VisualSceneNode1"
        name="mat0_CityEngineMaterial_CE" type="NODE">
        <instance_geometry name="mat0_CityEngineMaterial_CE"
          url="#Geometry">
          <bind_material>
            <technique_common>
              <instance_material symbol="mat0_CityEngineMaterial_CE"
                target="#VisualMaterial">
                <bind_vertex_input semantic="cityengine_colormap"
                  input_semantic="TEXCOORD" input_set="0"/>
                <bind_vertex_input semantic="cityengine_dirtmap"
                  input_semantic="TEXCOORD" input_set="2"/>
              </instance_material>
            </technique_common>
          </bind_material>
        </instance_geometry>
      </node>
    </node>
  </visual_scene>
</library_visual_scenes>
```

CityEngine assets and meshes

```
<library_geometries>
  <geometry id="Geometry" name="mesh1">
    <mesh>
      ... <triangles> or <polylist> ...
    </mesh>
  </geometry>
</library_geometries>
```

## CityEngine materials

```

<library_materials>
  <material id="VisualMaterial" name="mat0_CityEngineMaterial_CE">
    <instance_effect url="#Effect"/>
  </material>
</library_materials>

<library_effects>
  <effect id="Effect">
    <profile_COMMON>
      <newparam sid="Image-surface">
        <surface type="2D">...</surface>
      </newparam>
      <newparam sid="Image-sampler">
        <sampler2D>...</sampler2D>
      </newparam>
      <newparam sid="Image1-surface">
        <surface type="2D">...</surface>
      </newparam>
      <newparam sid="Image1-sampler">
        <sampler2D>...</sampler2D>
      </newparam>
      <technique sid="common">
        <lambert>
          <emission><color>0 0 0 1</color></emission>
          <ambient><color>0 0 1 1</color></ambient>
          <diffuse>
            <texture texture="Image-sampler"
              texcoord="cityengine_colormap">
              ...
            </texture>
            <texture texture="Image1-sampler"
              texcoord="cityengine_dirtmap">
              ...
            </texture>
          </diffuse>
        </lambert>
      </technique>
    </profile_COMMON>
  </effect>
</library_effects>

```

## CityEngine bump and normal maps

```
<effect id="MyEffect">
  <profile_COMMON>
    <technique sid="common">
      <lambert>...</lambert>
      <extra>
        <technique>
          <bump>
            <texture>...</texture>
          </bump>
        </technique>
      </extra>
    </technique>
  </profile_COMMON>
</effect>
```

## CityEngine textures

```
<library_images>
  <image id="Image">
    <init_from>../brickwall2.tif</init_from>
  </image>
  <image id="Image1">
    <init_from>../dirtmap.15.tif</init_from>
  </image>
</library_images>
```

# Export DATASMITH (Unreal and Twinmotion)

You can export Unreal Engine and Twinmotion (DATASMITH) data from CityEngine to the Unreal Engine Editor and Twinmotion. This is typically used for architectural and design visualization workflows, but you can also use it for games and other tasks.

 **Note:**


- Exporting Unreal data is only available on Windows.
- To export Unreal data in CityEngine, the [Unreal Engine Launcher](#) must be installed first.
- The CityEngine Model Loader Project is no longer needed and will be removed from the Unreal Engine Launcher.

## Export options

In addition to the [general export options](#), the following options are available for DATASMITH export:

Option	Description
Export Twinmotion Compatible	Makes the DATASMITH export compatible with Twinmotion. The export uses Unreal Base Materials, which are compatible with both Twinmotion and Unreal (the <b>Use Unreal Base Materials</b> export option must be set to true). It also disables features that are not compatible with Twinmotion: <ul style="list-style-type: none"><li>• Texture atlasing</li><li>• Instancing options</li><li>• Exporting terrains</li><li>• Exporting metadata</li><li>• Exporting LODs</li><li>• Exporting scenarios</li></ul>
Global Offset	Applies global offset for generated geometry for x, y, and z axes (Cartesian coordinate values).
Mesh Merging	Sets how the resulting meshes are merged: <ul style="list-style-type: none"><li>• <b>Per Initial Shape</b>—Meshes are merged by initial shapes.</li><li>• <b>Per Initial Shape by Material</b>—Meshes are merged by initial shapes by material. This results in one mesh per material for each initial shape.</li><li>• <b>Globally</b>—Meshes are merged globally.</li><li>• <b>Globally by Material</b>—Meshes are merged globally by material.</li></ul>

Option	Description
<b>Instancing</b>	<p>Sets how instancing is handled:</p> <ul style="list-style-type: none"> <li>• <b>Disabled</b>—Instancing is disabled.</li> <li>• <b>Use Instanced Meshes</b>—Instancing is enabled, which results in meshes being shared between actors, if possible.</li> <li>• <b>Use Instanced Mesh Actors</b>—Instancing is enabled, and instances are added using hierarchical instanced static mesh (HISM) actors.</li> </ul>
<b>Metadata</b>	<p>Exports metadata when <b>Mesh Merging</b> is set to either <b>Per Initial Shape</b> or <b>Per Initial Shape by Material</b>. The following are the metadata export options:</p> <ul style="list-style-type: none"> <li>• <b>All</b>—Writes object attributes and reports to object metadata.</li> <li>• <b>Attributes</b>—Writes object attributes to object metadata.</li> <li>• <b>Reports</b>—Writes generated report data to object metadata.</li> <li>• <b>None</b>—Object attributes and report data are not written to object metadata.</li> </ul> <p>The exported metadata is attached to the root object of each initial shape in Unreal.</p>
<b>Terrain Layers</b>	<p>Exports terrain layers to Unreal Engine <a href="#">Landscapes</a>. The following options are supported:</p> <ul style="list-style-type: none"> <li>• <b>Do not export any terrain layers</b></li> <li>• <b>Export all visible terrain layers</b></li> <li>• <b>Export all selected terrain layers</b></li> <li>• <b>Export all terrain layers</b></li> </ul>
<b>Use Unreal Base Materials</b>	<p>Uses Unreal Base Materials as the primary material for all exported materials. These predefined materials are compatible with both Twinmotion and Unreal. They support the following CityEngine material attributes:</p> <ul style="list-style-type: none"> <li>• <code>material.colormap</code> and <code>material.color</code></li> <li>• <code>material.normalmap</code></li> <li>• <code>material.opacitymap</code> and <code>material.opacity</code></li> <li>• <code>material.emissivemap</code> and <code>material.emissive.{r g b}</code></li> <li>• <code>material.roughnessmap</code> and <code>material.roughness</code></li> <li>• <code>material.metallicmap</code> and <code>material.metallic</code></li> </ul>
<b>Use Texture Atlases</b>	<p>Enables texture atlases that combine exported textures and reduce the number of materials. Texture atlases typically have the greatest impact when used on slower hardware, for example, on mobile VR devices such as the Oculus Quest.</p>

Option	Description
<b>Export LOD</b>	Enables export of level of detail (LOD). Exporting LOD is only enabled when exporting per initial shape.
<b>LOD Attribute</b>	<p>The name of the CGA attribute that specifies the LOD. The attribute must have an Enum annotation. For each element of the Enum annotation, a derivation is triggered when the LOD attribute is set to this element value.</p> <p>The following example generates three LODs per initial shape:</p> <pre>@Enum("low", "medium", "high") attr Lod = "low"</pre> <p> <b>Note:</b> If the attribute is defined in multiple CGA files, it must have the same elements (LODs).</p>
<b>LOD Order</b>	<p>The semantic order of the <b>LOD Attribute</b> values.</p> <ul style="list-style-type: none"> <li>• <b>Ascending</b>—The lowest level of detail is defined first.</li> <li>• <b>Descending</b>—The highest level of detail is defined first.</li> </ul> <p>For example, if the <b>LOD Attribute</b> option is defined as follows, the order is ascending, as the lowest level of detail is defined first.</p> <pre>@Enum("low", "medium", "high") attr Lod = "low"</pre>
<b>Export Scenarios</b>	Enables export of CityEngine <a href="#">scenarios</a> to Unreal Engine <a href="#">variants</a> . For each scenario, you can individually specify whether to export it as a variant.
<b>Default Objects</b>	<p>Defines how default objects are exported.</p> <ul style="list-style-type: none"> <li>• <b>From Selection</b>—Only the selected default objects are exported.</li> <li>• <b>Visible</b>—All default objects that are visible in the scene are exported and the selection is ignored.</li> </ul>

## Mapping between CityEngine and Unreal entities

### Geometry

Each exported mesh is represented as an [actor](#) in Unreal. Each unique mesh is exported as a [static mesh](#) and reused if instancing is enabled.

### Materials

Materials are mapped to Unreal by passing CGA material attributes to an Unreal [material instance](#), which is based on a certain primary (or parent) material.

By default, appropriate primary materials are generated that handle the mapping from CGA material attributes to Unreal material attributes (see also [Default primary materials](#)).

### *Custom primary materials*

A custom primary material can be assigned by setting the `material.shader` CGA attribute to the corresponding Unreal material path, for example, `/Game/Materials/CityEngineMaterials/M_CE_MyMaterial1`.

Each CGA material attribute is exported and can be accessed in Unreal using a material parameter with a specific name and data type. The following table lists the CGA material attributes and their respective Unreal parameter name and type:

CGA attribute	Unreal parameter name	Unreal type	Notes
<code>material.color</code>	Color	Vector3	
<code>material.opacity</code>	Opacity	float	
<code>material.reflectivity</code>	Reflectivity	float	
<code>material.shininess</code>	Shininess	float	
<code>material.bumpValue</code>	BumpValue	float	
<code>material.ambient</code>	Ambient	Vector3	
<code>material.specular</code>	Specular	Vector3	
<code>material.colormap</code>	ColorMap	Texture2D	
<code>material.dirtmap</code>	DirtMap	Texture2D	
<code>material.specularmap</code>	SpecularMap	Texture2D	
<code>material.opacitymap</code>	OpacityMap	Texture2D	
<none>	OpacitySource	float	For the <code>OpacitySource</code> value, select from the following: <ul style="list-style-type: none"> <li>• 0—Use color channels (gray level) from <code>OpacityMap</code>.</li> <li>• 1—Use alpha channel from <code>OpacityMap</code>.</li> </ul>
<code>material.bumpmap</code>	BumpMap	Texture2D	
<code>material.normalmap</code>	NormalMap	Texture2D	
<none>	IsPBR	float	<ul style="list-style-type: none"> <li>• 1—If <code>material.shader</code> is set to <code>CityEnginePBRShader</code></li> <li>• 0—Otherwise</li> </ul>
<code>material.metallic</code>	Metallic	float	
<code>material.roughness</code>	Roughness	float	

CGA attribute	Unreal parameter name	Unreal type	Notes
<code>material.emissivecolor</code>	EmissiveColor	Vector3	
<code>material.metallicmap</code>	MetallicMap	Texture2D	
<code>material.roughnessmap</code>	RoughnessMap	Texture2D	
<code>material.occlusionmap</code>	OcclusionMap	Texture2D	
<code>material.emissivemap</code>	EmissiveMap	Texture2D	

### *Default primary materials*

By default, an appropriate primary material (opaque, transparent, or masked) is generated during export, based on the following rules:

- If `material.opacity` is smaller than 1 or `material.opacitymap` is set
  - and the `opacitymap.mode` is blend, a Transparent primary material is used.
  - and the `opacitymap.mode` is masked, a Masked primary material is used.
- Otherwise the Opaque primary material is used.

#### **Note:**

Unreal Engine sorts transparency per actor. Exporting overlapping transparent actors leads to rendering artifacts. Masked binary opacity is not affected by this limitation.

### *Default material attribute mapping*

The default primary materials map the CGA material attributes roughly to a physically based Unreal Material. If `material.shader` is set to `CityEnginePBRShader` (this happens automatically if, for example, an inserted glTF model is exported), the default primary materials in Unreal use the PBR material attributes (roughness, metallic, and emissive).

#### **Note:**

The specular color is not used in Unreal. The specular color in Unreal depends on the base color and how metallic the material is.

# Export DWG (Autodesk)

Autodesk DWG is a proprietary 3D geometry format native to Autodesk AutoCAD. Although it doesn't have modern material support, it is widely used in the BIM and CAD industries.

## Export options

In addition to the [general export options](#), the following option is available for DWG export:

Option	Description
<b>Flip vertical axis of textures</b>	Flips the textures along the vertical axis. This is necessary because Autodesk AutoCAD and Autodesk TrueView use different orientations.

# Export FBX (Autodesk)

Autodesk FBX is a proprietary 3D geometry exchange format with a freely available SDK. It has gained popularity as the main exchange format for the Unreal and Unity game engines and related AR/VR applications. It currently lacks modern material support (for example, PBR) and metadata transport. For use cases in which AV/VR applications are not relevant, import using [glTF](#) instead.

## Export options

In addition to the [general export options](#), the following options are available for FBX export:

Option	Description
Create Shape Groups	If enabled, a transformation node is inserted for each shape (building). Meshes are not merged by material across shapes.
Embed Textures	If enabled, textures are stored in the binary FBX file.

## CGA mapping to FBX

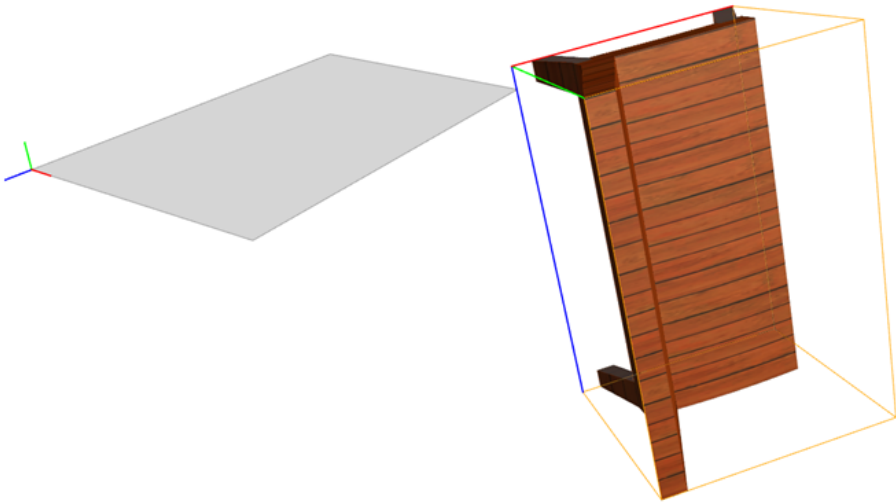
As the FBX file format is verbose, selected examples are shown below. Refer to [the Autodesk documentation](#) for further information.

### Geometry and transformation data

The subsections below contain examples of exporting CityEngine data to FBX in Autodesk Maya.

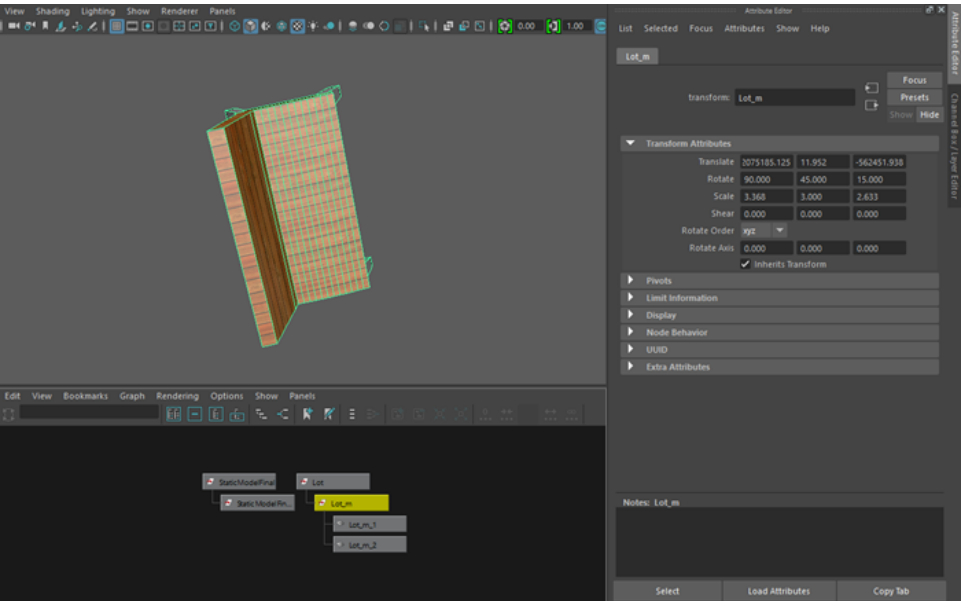
#### *CityEngine geometry and transformation data*

The pivot is in the origin; the scope (yellow) contains a translation and a rotation.



#### *Exported FBX data in Autodesk Maya*

The following screen capture shows an exported scene with FBX data in Autodesk Maya. Both **One Mesh Per** options have been disabled to avoid merging assets. In this case, each asset is parented to a transformation node. Otherwise, the transformation is applied to the vertices.



Multitexture and layered texture nodes

CityEngine

The CityEngine material model multiplies all color textures (color map and dirt map, if present) with the diffuse color. This example displays the layering and multiplication of a color map and a dirt map.

A screenshot of the CityEngine software interface. The main 3D viewport shows a perspective view of a building model with a grid-like texture. The top menu bar includes View, Shading, Lighting, Show, Renderer, and Panels. The bottom toolbar contains various editing tools. On the right, the Attribute Editor panel is open, showing the 'Lot\_m' attribute. It displays a 'transform' field set to 'Lot\_m' and a 'Transform Attributes' section with fields for Translate, Rotate, Scale, Shear, Rotate Order, and Rotate Axis. Below this, there are sections for Pivots, Limit Information, Display, Node Behavior, UUID, and Extra Attributes. The 'Notes' section at the bottom of the panel shows 'Notes: Lot\_m'.

Exported FBX data inAutodesk Maya

For FBX data, CityEngine exports multiple textures as layered texture nodes whose blend modes are set to multiply.

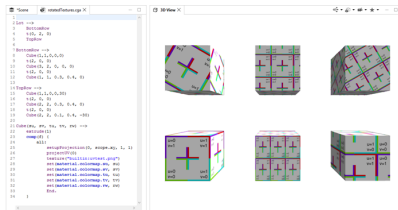
A screenshot of the Autodesk Maya software interface. The main 3D viewport shows a perspective view of a building model with a grid-like texture. The top menu bar includes File, Edit, Windows, and Help. The bottom toolbar contains various editing tools. On the right, the Attribute Editor panel is open, showing the 'Lot\_m' attribute. It displays a 'transform' field set to 'Lot\_m' and a 'Transform Attributes' section with fields for Translate, Rotate, Scale, Shear, Rotate Order, and Rotate Axis. Below this, there are sections for Pivots, Limit Information, Display, Node Behavior, UUID, and Extra Attributes. The 'Notes' section at the bottom of the panel shows 'Notes: Lot\_m'.

Work with per-texture transformations

CityEngine

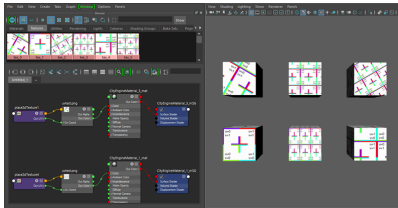
CityEngine features **material attributes** to scale and translates and rotates textures independently of the actual texture coordinates stored inside the assets:

- material.{...}.tu/tv = translate/offset texture
- material.{...}.su/sv = scale/repeat texture
- material.{...}.rw = rotate texture around face normal



## Exported FBX to Autodesk Maya

Upon exporting FBX data into Autodesk Maya, the CityEngine tu and tv attributes are mapped to the Offset parameter of the Autodesk Maya place2dTexture nodes, su/sv are mapped to the RepeatUV parameter, and rw is mapped to the Rotate Frame parameter.



# Export FileGDB (Esri File Geodatabase)

The Esri File Geodatabase (FileGDB) format is a file-based database that supports many GIS data types such as points, lines, polygons, 3D geometry (multipatch), raster, and so on. It is the recommended way to exchange GIS data between Esri applications such as CityEngine and ArcGIS Pro.

 **Note:**

CityEngine supports the export of datasets of the types "FeatureClass", "Table" and "Relationship Class."

## Export options

In addition to the [general export options](#), the following options are available for FileGDB export:

Option	Description
Geodatabase Name	The name of the geodatabase directory. The .gdb suffix is required.
Export Features	Controls what is exported per selected scene shape. The following are available values: <ul style="list-style-type: none"><li>• <b>Models</b>—Only the output of an attached CGA rule is exported.</li><li>• <b>Shapes</b>—Only the geometry of the shapes is exported, and any CGA rules are ignored.</li><li>• <b>Models and Shapes</b>—Both the shapes and the generated models are exported. This typically results in more than one feature class.</li></ul>
Export object attributes	Exports all object attributes as feature class fields. Information about data types and domains is reused, if available, from an earlier FileGDB import. Array values are ignored if the next option, <b>Export dataset relationships</b> , is not selected and the attribute is not part of a relationship.
Export dataset relationships	<ul style="list-style-type: none"><li>• All (array) attributes that result from an earlier FileGDB import with relationship import are exported to the corresponding related tables.</li><li>• Due to a limitation in the underlying FileGDB API, relationship classes cannot be generated in CityEngine directly. Instead, an ArcGIS workspace .xml file is generated. See <a href="#">Importing a geodatabase schema</a> for more information.</li></ul>
Emit Reports	Any reports generated by the attached CGA rule are written into feature class fields.

In addition to the global export options, FileGDB export contains an export page with the following per-layer options:

Option	Description
Export Layer	Include or exclude selected layers from export.

Option	Description
<b>Layer Name</b>	Update the name of the selected layers. The name of the written FileGDB datasets are based on this value.
<b>Write Strategy</b>	Specify how existing feature classes are handled: <ul style="list-style-type: none"> <li>• <b>Replace Feature Class</b>—This deletes the current feature class and creates a new one.</li> <li>• <b>Update Feature Class</b>—If objects (based on OBJECTID) exist in the feature class, geometry and attributes are updated on export. If the object does not exist, it is appended to the feature class.</li> <li>• <b>Update Feature Class Geometry</b>—Only the geometry of the existing feature class is updated.</li> </ul>
<b>Geometry Type (for Shapes)</b>	Specify the geometry type for the shapes: <ul style="list-style-type: none"> <li>• <b>Polygon</b></li> <li>• <b>Multipatch</b></li> </ul>

## Exported feature class names

For FileGDB export, multiple feature classes may be output based on a single CityEngine scene layer. For example, if you export a wall shape of a building layer, the following are the potential results:

- If **Export Features** is set to **Shapes** or **Models and Shapes**, the wall shape is written directly to the building feature class.
- If wall has a CGA rule attached that produces polygons, its output results in a `building_ProcedurallyGeneratedMultipatches` multipatch feature class.
- If wall has a CGA rule attached that produces points (for example, by `comp(v)`), its output results in the `building_ProcedurallyGeneratedPoints` or `building_ProcedurallyGeneratedMultipoints` point or multipoint, respectively, feature class.
- If wall has a CGA rule attached that produces edges (for example, by `comp(e)`), its output results in the `building_ProcedurallyGeneratedLines` polyline feature class.

# Export glTF/glb (Khronos Group)

glTF/glb (version 2.0) is a JSON-based 3D geometry delivery format. It supports modern materials (PBR) and geometry instancing and is supported by many desktop and web apps. It is a recommended exchange format for new projects.

The glb file format variant (see the [glTF Output Format](#) option in the table) is a binary form of glTF that inlines textures instead of referencing them as external images. This results in a single file for the exported scene.

For details, see the [KhronosGroup GitHub page](#).

## Export options

In addition to the [general export options](#), the following glTF option is available:

Option	Description
<b>glTF Output Format</b>	When set to <b>glTF binary</b> , the file is stored as a .glb file that contains all geometry and texture data. When set to <b>glTF JSON</b> , the file is stored as a .gltf file with geometry stored in .bin files and textures as .png files.

## glTF to CGA mapping

The following table lists the material mapping from glTF to CGA:

glTF material	CGA material
baseColorTexture	material.colormap and material.opacitymap
baseColorFactor	material.color and material.opacity
alphaMode	material.opacitymap.mode
metallicRoughnessTexture	material.metallicmap and material.roughnessmap
metallicFactor	material.metallic
roughnessFactor	material.roughness
normalTexture	material.normalmap
occlusionTexture	material.occlusionmap
emissiveTexture	material.emissivemap
emissiveFactor	material.emissive

 **Note:**

- doubleSided, alphaCutoff, normalTexture.scale, and occlusionTexture.strength materials are not supported.
- Other CGA material maps are ignored.

# Export IFC (buildingSMART)

buildingSMART IFC is an open-source 3D geometry exchange format to describe building elements. It is mainly used in the BIM and CAD industries to store project information for design, procurement, and construction phases as well as reference for operations. CityEngine only supports version 2x3.

## Export options

The [general export options](#) are used for IFC.


# Export KML/KMZ (Keyhole Markup Language)

Keyhole Markup Language (KML) is an XML-based exchange format used to describe GIS features on the globe. It supports points, lines, and polygons and can reference 3D models through COLLADA. It is used to produce data for ArcGIS Earth.

KMZ is a compressed version of KML in which all referenced COLLADA and texture files are contained in one zipped file.

## Export options

In addition to the [general export options](#), the following options are available for KMZ/KML export:

Option	Description
Altitude Mode	Controls which altitude tags are written to the .kml and .kmz files: <ul style="list-style-type: none"><li>• <b>clampToGround</b>—Aligns the object to the ground, and altitude is ignored.</li><li>• <b>absolute</b>—Ignores the ground elevation.</li></ul> <div> <b>Note:</b> For <b>absolute</b>, the ground elevation and the altitude must coincide precisely.</div>
Write Compressed Files	When set, the written .kml file is put into a .kmz archive file along with the other necessary files such as .dae files and texture images.
Heading Correction	ArcGlobe, Google Earth, and other earth browsers interpret the COLLADA .dae file content differently. For Google Earth, the Heading Correction switch must be turned on for correct results.
Placemark Locations	<ul style="list-style-type: none"><li>• <b>Exact spatial references per model</b>—Unprojects placemark locations per shape. This is recommended for unconnected objects (such as buildings) for bigger areas.</li><li>• <b>Optimized Model placement</b>—Optimizes placemark locations to ensure correct relative positions (unproject only in the lead shape, other shapes are referenced relatively). This is recommended for connected objects (such as streets) over small areas.</li></ul>

 **Note:**

Use the KML export presets **Google Earth Compatibility**, **ArcGlobe Compatibility**, or **SketchUp Compatibility** to set recommended options for these programs.

## CityEngine KML output

When saving some selected objects by the name MyPlacemarks, the export process writes the MyPlacemarks.kml file. Next to it, there is a folder named MyPlacemarks.kml-files containing the associated .dae file and textures.

*Example KML file output*

The following is sample code for KML output:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:atom="http://www.w3.org/2005/
Atom" xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2">
  <Document>
    <name>MyPlacemarks</name>
    <Folder>
      <Placemark>
        <name>aPlacemark</name>
        <Model>
          <altitudeMode>clampToGround</altitudeMode>
          <Location>
            <longitude>-75.17193217718045</longitude>
            <latitude>39.95381194059072</latitude>
            <altitude>10.82616576552391</altitude>
          </Location>
          <Orientation>
            <heading>0.0</heading>
            <tilt>0.0</tilt>
            <roll>0.0</roll>
          </Orientation>
          <Scale>
            <x>1.0</x>
            <y>1.0</y>
            <z>1.0</z>
          </Scale>
          <Link>
            <href>MyPlacemarks.kml-files/aPlacemark.dae</href>
          </Link>
        </Model>
      </Placemark>
      <Placemark>
        <name>anotherPlacemark</name>
        ...
      </Placemark>
      ...
    </Folder>
  </Document>
</kml>
```

# Export MSPK (Esri Mobile Scene Package)

Esri mobile scene packages (MSPK) are custom, web-optimized files that can be shared across the ArcGIS Platform. They can be uploaded on ArcGIS Online and viewed in ArcGIS Pro as well as ArcGIS Earth.

To export only parts of a scene, select them in the **Viewport** window or the **Scene Editor** window. To export the whole scene, you don't have to select anything. To start the export, click **File > Export > CityEngine > Export Mobile Scene Package** from the main menu.

## Export options

For MSPK export, the following options are available:

Option	Description
Output File	The file to which the mobile scene package is exported.
Export Content	<ul style="list-style-type: none"><li>• <b>Complete Scene</b>—Exports all exportable parts of the scene.</li><li>• <b>Selection</b>—Exports only the elements selected in the <b>Viewport</b> or <b>Scene Editor</b> window.</li></ul>
Scene Environment	Choose between local and global scene export. Local scenes remain in the coordinate system of the scene, and global scenes are converted to be viewable on the whole globe.

## Upload MSPK data to ArcGIS Online

To upload MSPK data, complete the following steps:

1. Locate the exported .msp file in the [Navigator](#) window.  
By default the .msp file is in the data folder of the project.
2. Right-click the .msp file and choose **Share As**.  
The mobile scene package dialog box appears.
3. Click **Upload package to my ArcGIS Online account**.
4. Name the .msp file.
5. Fill in the required fields on the **Item Description** tab.
6. Click **Analyze** to validate the .msp file for any errors or issues.  
Fix the errors before continuing.
7. Click **Share** to upload the .msp file to ArcGIS Online.

.msp files can be uploaded to other ArcGIS portals as well. See [Share data on a different portal](#) for additional information.

# Export OBJ (Wavefront)

Wavefront OBJ is a text-based legacy 3D model exchange format. Despite its limitations in efficiency and features (for example, no modern material support or geometry instancing), it remains popular due to its simple syntax and manual editability.

## Export options

The [general export options](#) are used for OBJ export. If there are issues displaying the exported model correctly in other programs, set **Triangulate Meshes** to true.

## CGA mapping to OBJ

The subsections below outline CGA mapping to OBJ.

### Geometry

Geometry mapping is outlined in the following table:

OBJ element	CGA feature
v	Vertex data from asset meshes.
vn	Vertex normal data from asset meshes.
vt	Texture coordinates from colormap texture channel of asset meshes. The per-texture transformations (see <a href="#">material.colormap.{su, sv, tu, tv, rw}</a> ) are not included in these texture coordinates.
f	The mesh faces or polygons defined by the vertex, vertex normals, and texture coordinates indices from the asset meshes.
g	The face group name. If the mesh comes directly from an inserted asset, the original name is used. If <b>Mesh Granularity</b> is set to merge meshes by material, this name is controlled by the <a href="#">material.name attribute</a> . Otherwise, an internal name is set depending on the operation that created the geometry.
s	Smoothing groups are not supported and are turned off. Use vertex normals instead.
usemtl	Material name and reference into the corresponding .mtl file (see the Material section below). It is only written if the <b>Materials</b> export option is enabled.
mtllib	Local reference to the corresponding .mtl file. It uses the same base name as the .obj file (but with the .mtl extension) and is written to the same directory.

### Material

Material mapping is outlined in the following table:

 **Note:**

The material definitions are exported into separate .mtl files.

MTL element	CGA feature
newmtl	The material name corresponds to the usemtl statement in the .obj files.

MTL element	CGA feature
illum	If the material of the mesh contains a specular color component equal to (0,0,0) a Lambert material is exported (illum is set to 3). For Phong materials (specular component != zero or reflectivity), it is set to 4.
Kd	Diffuse color. Use the <code>material.color</code> setting.
map_Kd	Diffuse texture. Use the <code>material.colormap</code> setting. This statement is only written if a texture file is assigned to the colormap channel. Optionally, the following uv translation and scaling factors are exported:  If <code>material.colormap.{su,sv}</code> are != 1.0, the -s option is appended with the scaling factors.  If <code>material.colormap.{tu,tv}</code> are != 0.0, the -o option is appended with the translation values.
Ka	Ambient color. Use the <code>material.ambient</code> setting.
Ks	Specular color. Use the <code>material.specular</code> setting if the material is of type Phong (illum = 4).
d	Opacity,. Use the <code>material.opacity</code> setting.
map_d	Opacity map. Use the <code>material.opacitymap</code> setting.
Ns	The specular exponent of the Phong lighting model, also called shininess. Use the <code>material.shininess</code> setting.
Tf	For Maya compatibility, set to (1.0, 1.0, 1.0).
Ni	For Maya compatibility, set to 1.0.

See [Rendering, import, and export](#) for more information.

# Script-based export (Python)

[Script-based export](#) runs a Python script during export model generation, without writing geometry to a file.

To export to a geometry format and run an export script simultaneously, use a format export and specify the script in the export dialog box using the **Script** option.

## Export options

For script-based export, the following export option is available

Option	Description
<b>Script</b>	The workspace path to the Python script

# Export SLPK (Esri Scene Layer Package)

Esri scene layer packages (SLPK) are custom, web-optimized files that can be shared on ArcGIS Online. For details, refer to <https://github.com/Esri/i3s-spec/>.

Select the content to export in the **Viewport** window, and begin exporting. Click **File > Export Models > Esri Scene Layer Package** from the main menu.

## Export options

In addition to the [general export options](#), the following options are available for SLPK export:

Option	Description
Combine Layers	Instead of writing one package per layer, you can write one package containing the data in all layers. If you chose to merge all layers, the per-layer options are not shown.
Emit Reports	Reports generated by the attached CGA rule are written to attribute fields.
File size	The size of the file: <ul style="list-style-type: none"><li>• <b>Small File — Low Detail</b>—Creates a small file; all selected models are exported with possible reduced quality.</li><li>• <b>Midsized File — Medium Detail</b>—Creates a midsized file; models are exported with medium level of detail.</li><li>• <b>Large File — Good Detail</b>—Creates a large file; models are exported with good level of detail.</li><li>• <b>Huge File — High Detail</b>—Creates a huge file; models are exported with high level of detail.</li></ul>
Scene Environment	Choose between local and global scene export. Local scenes remain in the coordinate system of the scene, and global scenes are converted to be viewable on the whole globe.

In addition to the global export options, there is an extra export page with the following per-layer options:

Option	Description
Export	Check boxes denote if a layer's contents are exported.
Layer Name	The layer name, as it is set in the CityEngine scene.

## Export tips

If a scene layer created from a CityEngine export doesn't export correctly, consider reducing the size of the SLPK by doing one of the following:

- Choose a smaller extent for export (select fewer objects).
- Fine-tune the texture quality export options (use compact or half-size).
- Reduce the geometric complexity of the models (for example, lower level of detail or less details on streets).

When preparing an SLPK, keep in mind that a scene layer may not run as well (or run at all) on other, less powerful systems (less memory, less powerful graphics card). Reduce the exported extent and .spk file size to ensure wider compatibility.



#### Note:

The SLPK format has a limit on the geometric complexity of a single model. If, after triangulation, a model has more than 127,000 triangles, it may not display in clients.

## Publish the SLPK on ArcGIS Online

Upload a file and publish a scene layer.

### Upload to ArcGIS Online

Complete the following steps to upload an SLPK:

1. Locate the exported SLPK in the **Navigator** window.
2. Right-click the SLPK and click **Share As**.  
The **Scene Package** dialog box appears.
3. Click **Upload package to my ArcGIS Online account** and name the SLPK.
4. Fill in the required fields for **Item Description**.
5. Click **Analyze** to validate the SLPK for any errors or issues.  
You must validate and resolve all errors before you can save the SLPK to disk or share it to ArcGIS Online. If any issues are encountered, an error is reported. You must fix the error before you can continue.
6. Click **Share** to upload the SLPK to ArcGIS Online.

.spk files can be uploaded to other ArcGIS portals as well. See [Share data on a different portal](#).

### Publish a scene layer from the SLPK

After uploading the SLPK to ArcGIS Online or ArcGIS Enterprise, you must publish it to a scene layer. To create a scene layer, do the following:

1. In the **Navigator** window, click **Portal > My Content**.
2. Find the SLPK you just uploaded and double-click it.  
Your browser opens with the correct item open.
3. Click **Publish** to create a scene layer.  
The title of the package is preset as Scene Title. Change it to only alphanumeric characters and underscores.
4. Change the tags if necessary.
5. Click **Publish**.  
After waiting for the service to be created, you can open the scene layer.

# Export USD (Universal Scene Description)

You can export CityEngine scenes into a set of [Universal Scene Description](#) (USD) files with minimal loss of information, which makes it suitable as an interchange format for further processing of (potentially) large scenes in VFX pipelines. CityEngine layers, object boundaries, and asset instance information are preserved after export. Any PBR materials are exported to the [USDPreviewSurface](#) material, which guarantees correct shading in any compatible downstream application without additional material assignment work.

## Use cases

The following are use cases for exporting USD data:

- Focus on compatibility with tools such as SideFX Houdini and its Solaris component, Foundry Katana, and corresponding USD-compatible renderers.
- Export large models with complex rules while retaining shape granularity (no merging of geometry). This allows edits (for example, move or scale) to individual objects (for example, buildings) in downstream tools without the exporting the whole scene again.
- Preserve instances—for example, inserted assets not involved in CGA operations are exported as separate nodes. This allows limited editing (translate or scale) of these assets in downstream tools.
- Support for PBR materials using the standard USDPreviewSurface material schema.

## Export options

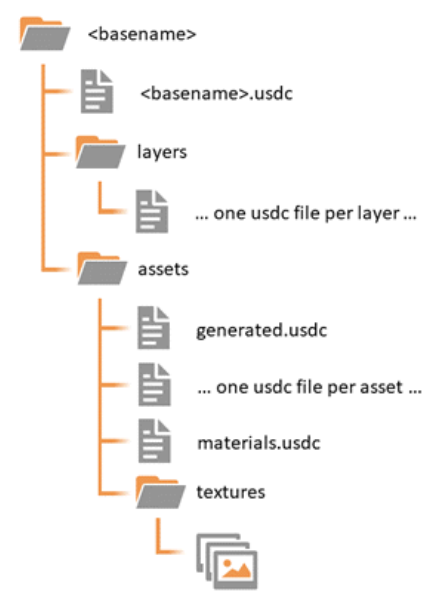
In addition to some of the [general export options](#), the following options are available for USD export:

Option	Description
Mesh Merging	Choose between creating one mesh per material per object, or one mesh per object with materials being mapped to polygon groups.
File Type	Choose between exporting a hierarchy of .usdc files and textures or a single .usdz package.

## Exported file layout

For each USD export session, a set of directories and files is created:

- <basename>.usdc is the root file with references to the CityEngine layers.
- The layers directory receives one .usd file per involved CityEngine layer. The layer files contain the scene objects (for example, building models), which reference the geometry assets.
- The assets directory receives one .usd file per asset as well as two special files:
  - generated.usdc receives all geometry that is generated by CGA.
  - materials.usdc receives all unique materials.
- The textures directory receives all texture files referenced by materials.usdc.



## Scene layout

Any USD importer will compose the above USD layer files into the following composed USD stage (scene graph):

Scene Hierarchy	USD Prim Type	USD Kind
/<export base name>	Xform	Group
/<CityEngine layer(s)>	Xform	Group
/<CityEngine object(s)>	Xform	Assembly
/<generated CGA leaf shape(s)>	Xform	Component
/mesh	Mesh	
/material	Material	
/PBRShader	UsdPreviewSurface	
/<texture(s)>	UsdUVTexture	
/<uv0Reader>	UsdPrimvarReader_float2	

Items not in angle brackets are named as indicated in the table, all other prims take the corresponding CityEngine scene, asset, or UV set names. As an example, here's the USD prim path to a PBRShader prim in an exported model from the "Philadelphia" example: /Philadelphia/Development\_\_\_Complete\_Block/Shape/shape/material/PBRShader.

## Metadata export

The USD exporter always writes metadata (object attributes, the main CGA rule attributes, non-default CGA user attribute values, and CGA reports) to USD prims (= the USD scene graph nodes). Because of different granularity, not all metadata are written to the same USD prims. The resulting USD attribute names are prefixed with "CityEngine:", see below.

By referring to the scene layout above, the metadata are written as follows:

- "CityEngine:ID" is written to all "CityEngine object(s)" and contains the CityEngine object ID (OID), as visible in the Information section of the CityEngine Inspector.
- Object attributes are also exported to "CityEngine object(s)" prims. The names are prefixed with "CityEngine:Object:". For example, the CityEngine "streetWidth" object attribute is exported as "CityEngine:Object:streetWidth" with values of type double array.
- The main CGA rule attributes are also written to the "CityEngine object(s)" prims:
  - "CityEngine:Rule:RuleFile" receives the absolute workspace path to the cgb file, for example `/Example_Philadelphia__2022_1/bin/Generic Modern Buildings.cgb`.
  - "CityEngine:Rule:StartRule" receives the start rule name used to generate the model.
  - "CityEngine:Rule:RandomSeed" receives the per-object random seed, as visible in the Information section of the CityEngine Inspector.
- Non-default CGA attributes are also written to "CityEngine object(s)" prims and prefixed with "CityEngine:Rule:User:".
- CGA report values are written on the level of "generated CGA leaf shape(s)" as array values in non-summarized form. For example, the report produced by the CGA statement `report("Lot Area (m2)", Lot_Area)` ends up on the leaf shape prim as "CityEngine:Reports:Footprint\_Area\_m2" (Parentheses are converted to underscore) with an array value of all emitted reports.

## Export VOB (e-on VUE)

VOB is the native object format for polygonal meshes of e-on software's VUE. CityEngine supports VUE 8.5 and later. It is recommended that you use the latest version of VUE.



### **Note:**

- VOB has no additional export options (see [General export options](#)).
- VOB only supports a single layer of texture coordinates (CGA uv layer 0).

# Model export application notes

## Note #1: Format Recommendations

Format recommendations for various software and functionality are listed in the table below.

### Note:

Some of the software and functionality listed require additional plug-ins to load all formats.

Software or functionality	Format	Required options
ArcGIS Earth	KML	
Autodesk Max	OBJ, FBX	For obj, enable import of smoothing groups in Max.
Autodesk Maya	FBX, ABC, OBJ	(ABC materials cannot be imported out of the box in Maya.)
Autodesk MotionBuilder	OBJ, FBX	
Blender	OBJ, FBX, ABC	Multitexturing
Cinema 4D	OBJ, DAE	Multitexturing
Deep Exploration	OBJ, FBX, DAE	
SideFX Houdini	ABC, USD	
Foundry Katana	ABC, USD	
Lightwave	OBJ	
Polytrans	OBJ, FBX, DAE	
e-on VUE	VOB, OBJ	

## Note #2: Working with Expensive Assets

If you work with large assets, you can create simplified proxy assets and switch between them with a global level of detail (LOD) attribute. To avoid scattering the LOD attribute over the rules, you can put the conditions into separate asset loader rules, as shown in the following sample code:

```
attr LOD = 0
...
Shaft -->
    s(diameter,'1,diameter) center(xz) color(shaftC) ShaftAsset

ShaftAsset -->
    case LOD == 0: primitiveCube()
    else: i("path/to/expensive/asset.obj")
...
```

## Note #3: Working with Large Models

To create large models, implement global CGA attributes into the rule sets so you can selectively block the generation of polygon-intensive model features. For example, you can replace high-polygon Greek columns with simple cuboids using an attribute LOD together with a corresponding condition in the CGA rules.

## Application example

To render a large scene with Maya or Arnold when you have a CityEngine scene with an LOD switch, export the scene with LOD = 0 to a single .obj file (without textures) and import it. For example, if you import it into Maya, you can quickly set up the lights and camera without using heavy geometry. Once the environment is ready, you can go back to CityEngine and export the scene with LOD = 1 to USD format files and link them to the render setup.

## Implementation example

An example of a modified version of the Parthenon temple shape grammar is shown below. The LOD attribute is used.

The temple with LOD = Low, or ~90k polygons

```
...

# ----- model
parameters -----

@Group("Model Options", 0)
@Order(1)
@Enum("High","Low")
attr Level_of_Detail = "Low"
const HighLOD =
Level_of_Detail == "High"

...

### Columns ###

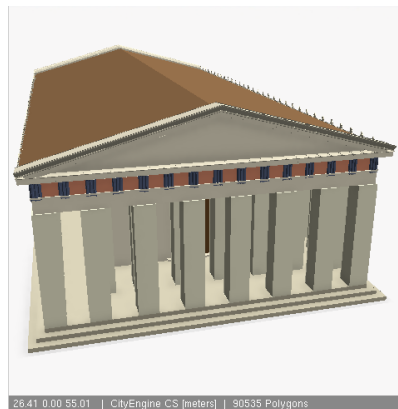
...

ColumnTile -->
  set(trim.vertical,false)
  primitiveCube() s( Diameter ,1,
Diameter ) t(' -0.5,0,' -0.5)
  color(Column_Color)

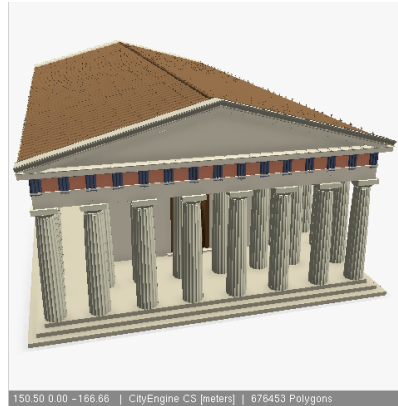
  [ case HighLOD : Column
    else : Column. ]

Column --> ...

...
```



The temple with LOD = High, or ~675k polygons



# Export terrain

You can export terrains as geometries, tile packages (.tpk file), or images by selecting the map layers to export.

## Export as a geometry

To export a terrain as a geometry, complete the following steps:

1. In the **Scene Editor** window, click **Terrain Layers**.
2. Click **File > Export > CityEngine > Export Models of Selected Shapes and Terrain Layers**.
3. For **Format**, choose a model format to export.  
The rest of the process is similar to the steps in [Export a model](#).



### Note:

The KML and scene layer package formats do not support exporting terrains.

## Export as a tile package


To export a terrain as a tile package, complete the following steps:

1. In the **Scene Editor** window, click **Terrain Layers**.
2. Click **File > Export Selected Layers as TPK**.
3. Under **Maps**, choose to export the basemap file, the elevation terrain file, or both as tile packages.  
If you export both the basemap file and elevation file, a {filename}\_Basemap.tpk file and a {filename}\_Elevation.tpk file are added to the output folder.
4. Use **Scene Environment**, to export the terrain for either a global or local scene.
5. Sign in to ArcGIS Online or ArcGIS Enterprise to share the tile packages with your organization or the public.
6. Right-click the .tpk file to share and click **Share as**.  
The **Tile Package** dialog box appears.
7. Complete the necessary fields and click **Share**.
8. Go to the **Content** tab in your account in ArcGIS Online or ArcGIS Enterprise and publish the tile package as a hosted elevation or tile layer.

## Export as an image

To export a terrain as an image, complete the following steps:

1. In the **Scene Editor** window, click **Terrain Layers**.
2. Click **File > Export > CityEngine > Export Selected Terrains as Image**.
3. For **Format**, choose an image format to export.
4. For **Resolution**, choose the resolution for the image to export.  
The **Resolution** setting determines the number of terrain mesh vertices in the U and V directions to export.  
Depending on the quality of the terrain resolution, you can have the following options:

- **Original**—Takes the maximum resolution of the selected terrains for each direction and applies the value to all of them.
- **4K**—4096 x 3079 pixels.
- **2K**—2048 x 1540 pixels.
- **1K**—1024 x 770 pixels.
- **Custom**—Manually set the resolution.
- **Aspect ratio** —Maintains the aspect ratio when manually entering values.

# Export shapes

To export shapes to an external format, do the following.

- 1. Select the shapes to export.
- 2. Click **File > Export**.
- 3. Click **City Engine Layers > Export Selected Shapes**.
- 4. Click **Next**.
- 5. Choose the Esri shapefile or AutoCAD DXF format.



**Tip:**

You can export shape geometry in CityEngine in two ways: export pure shapes or export models and choose to only [export the shape geometry](#).

## Export to SHP

The Esri shapefile is a geospatial vector data format for GIS software. A shapefile commonly refers to a collection of files. The main file (.shp) contains the shape data and the database file (.dbf) contains attributes for each shape. You can export shapes and attributes to shapefiles. Additionally, an index file (.shx) is written.



**Note:**

The same process can also be used to export graph segments. For a graph segment specific description, see [Export graphs](#).

## Export options

The export dialog box consists of the file name and 3D options. Presets can be saved and applied.

### File

Click **Browse** to open a dialog box and select an .shp file to export.

### 3D options

The following table illustrates the effects of this option:

3D option	Shapefile type	Data
none	Polygon	2D
PolygonZ	PolygonZ	3D
Multipatch	Multipatch	3D



**Note:**

No .prj projection file is written. The data is stored in Cartesian coordinates in the current scene coordinate system.

## Export to DXF

AutoCAD DXF, Drawing Interchange Format, or Drawing Exchange Format is a CAD data format developed by Autodesk. A .dxf file contains a set of entities. For each selected shape, a closed Polyline type entity is written.



### **Note:**

You can also export graph segments. For a graph segment specific description, see [Export to DXF](#)

## Export options

No export options are available for DXF data. To create a .dxf export file, click the **Browse** button and click **Finish**.

# Export graphs

Graph segments can be exported to shapefiles, geodatabases, and DXF. To export dynamic shapes and models that are based on graphs, see [Export shapes](#) and [Export a model](#). To export graph segments, do the following:

1. Select the graph segments to export.
2. Click **File > Export** from the main menu.
3. Choose **CityEngine > Export Selected Graph Objects**.
4. Click **Next**.
5. For **Format**, choose the format to export.  
Depending on the file format, different options are available for export.

## Export to SHP

The Esri shapefile is a geospatial vector data format for GIS software. A shapefile commonly refers to a collection of files. The main file (.shp) contains the shape data and the database file (.dbf) contains attributes for each shape. You can export graph segments and attributes to these files. Additionally, an index file (.shx) is written.

 **Note:**

- You can also export shapes. For a shape-specific description, see [Export shapes](#).
- Shapefile support is not available in all CityEngine versions.

## Export options

The export dialog box consists of the file name and the 3D options. Presets can be saved and applied.

### File

Click **Browse** to open a dialog box and select an .shp file to export.

### 3D options

The following table lists the available 3D options:

3D option	Shapefile type	Data
none	Polyline	2D
PolylineZ	PolylineZ	3D

 **Note:**

A .prj file is not written. The data is stored in Cartesian coordinates in the current scene coordinate system.

## Export to a file geodatabase

The Esri file geodatabase (FileGDB) is a file-based database for vector and raster data. It is identified by the .gdb extension, for example, myDatabase.gdb.

## Export options

The export dialog box consists of the file name option. Presets can be saved and applied.

### *File*

Click **Browse** to open a dialog box and select a **.gdb** destination file to export to.

## Export to DXF

AutoCAD DXF, Drawing Interchange Format, or Drawing Exchange Format, is a CAD data format developed by Autodesk. A **.dxf** file contains a set of entities. For each selected graph segment, an entity of type Line is written.



### **Note:**

You can also export shapes. For a shape-specific description, see [Export shapes](#).

## Export options

The export dialog box consists of the file name and the street width option. Presets can be saved and applied.

### *File*

Click **Browse** to open a dialog box and select a **.dxf** file to export.

### *Export street width*

When enabled, the street width is written into the Thickness field of the DXF Line entity (group code 39).



### **Note:**

The value written is the total street width, that is, the street width plus the sidewalk widths.

## Export 3VR (360 VR Experience)

The 360 VR Experience (3VR) file format publishes and consumes panoramic images of CityEngine scenes. For more information, see [360 VR Experience export](#).

# Share as web scene

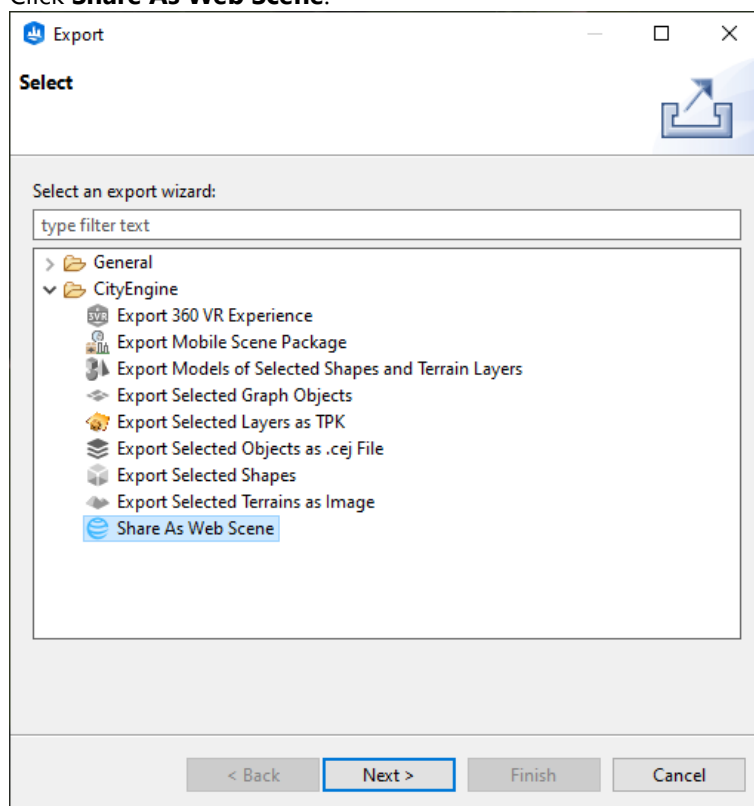
You can export a CityEngine scene or a selection of a scene to ArcGIS Online as a web scene that can be opened in Scene Viewer. Also, the layers from the CityEngine scene are published individually and can be opened in Scene Viewer and shared.

To learn more about sharing a CityEngine scene, visit [Tutorial 15: Publish web scenes](#).

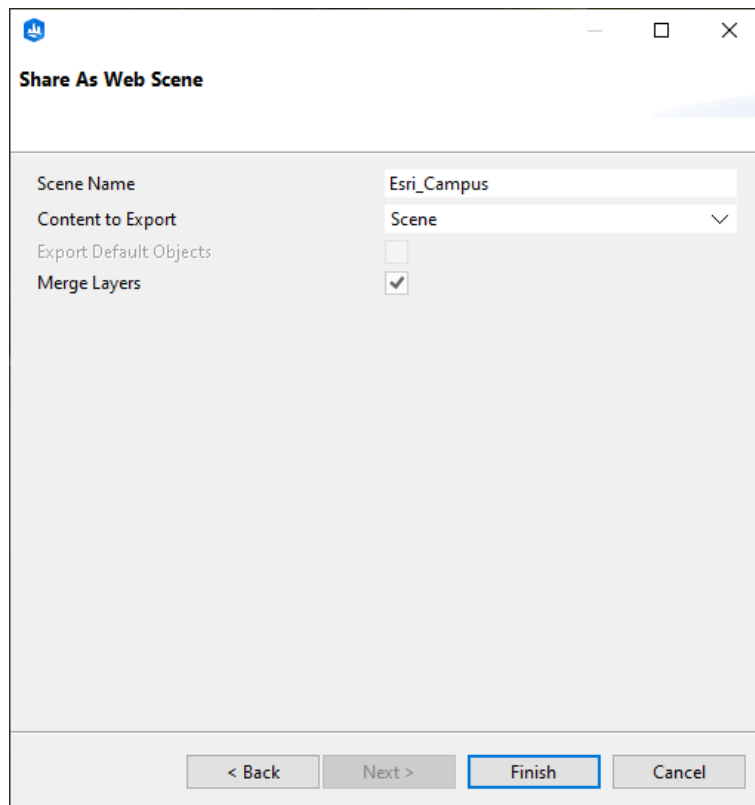
## Export a CityEngine scene to ArcGIS Online as a web scene

To export a CityEngine scene as a web scene, do the following:

1. Open the CityEngine scene you want to share.
2. Click **File** > **Export** to open the **Export** dialog box.
3. Click **Share As Web Scene**:



4. Click **Next** to open the **Share As Web Scene** dialog box.  
Sign in to ArcGIS Online if you are not already signed in to continue.
5. Keep the default export options:



If you created a scenario and want to export it individually, do the following:

- Click the **Content to Export** drop-down menu to select the scenario.
- Check the **Export Default Objects** check box to export the scenario with default objects.
- Check the **Merge Layers** check box to merge layers inside group layers.

 **Note:**

The terrain layers will not be merged.


6. Click **Finish**.

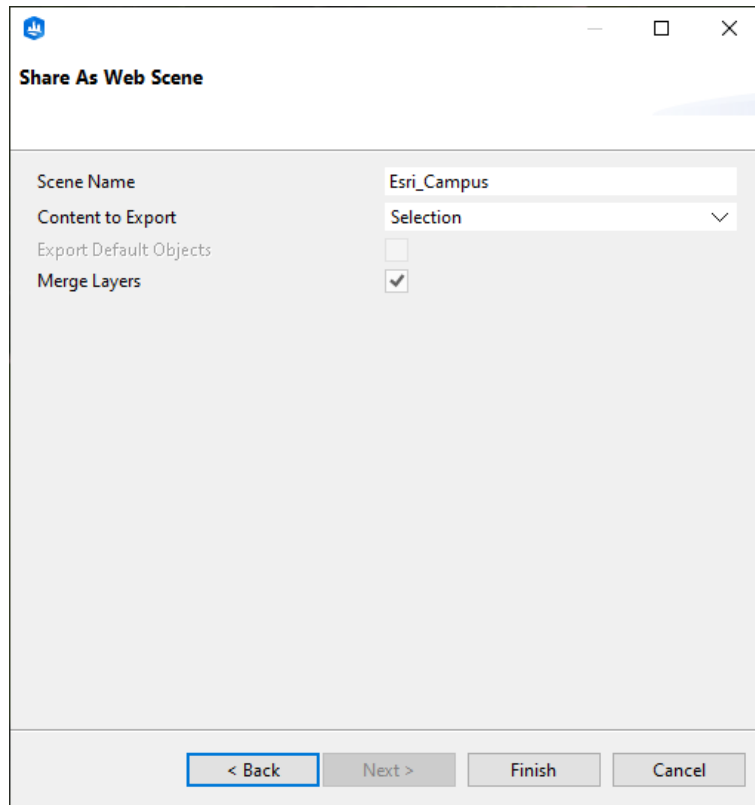
A web scene combining all layers is published to ArcGIS Online. Also, a tile package (.tpk) for each terrain layer and a scene layer package (.slpk) for all other layers are published. They are published to the current portal in which you are signed in. The CityEngine bookmarks with the saved camera configurations are exported to Scene Viewer as slides in the web scene.

After the scene is uploaded, your default browser opens the item in ArcGIS Online.

## Export a selection of a CityEngine scene to ArcGIS Online

To export a selection of a CityEngine scene, do the following:

- Click the **Select** tool  (Q) and to select part of the scene in the **Viewport** window.
- Open the **Share As Web Scene** dialog box as you did above.
- Click **Selection** in the **Content to Export** drop-down menu.



4. Click **Finish** to export the selection as a web scene.


## Share published items on ArcGIS Online

To share published items, do the following:

1. Click the item you want to share on the **Content** tab on ArcGIS Online.  
The item page appears.
2. Click **Share**.
3. Set the sharing level.
4. Click **Save**.

## Share published items using CityEngine

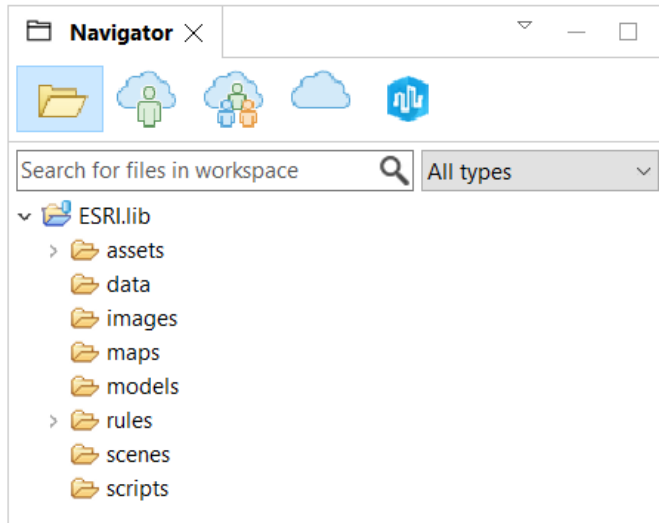
To share published scene items in CityEngine, do the following:

1. In the **Navigator** window, click **My Content**  to verify all the files and layers are there.  
If prompted, sign in to ArcGIS Online.
2. Right-click an item and click **Open in browser** to open the item page in ArcGIS Online.
3. On the ArcGIS Online item page, click **Share** to share the item.
4. Set the sharing level.
5. Click **Save**.

# Content Library (ESRI.lib)

# ESRI.lib

ESRI.lib is a CityEngine project that contains a library full of useful resources, such as CGA rules and assets. It is automatically installed in your CityEngine [workspace](#) and can be found in the [Navigator](#) window.



ESRI.lib is shown in the Navigator window.

You can apply rules and use assets as they are or import the rules into your own rules. For example, applying the `Plant_Loader.cga` rule to a shape in your scene inserts a plant model on that shape. In the [Inspector](#), you can adjust attributes for that shape, such as plant species and height. Alternatively, you can import the `Plant_Loader.cga` rule into one of your CGA files, and use CGA to control the insertion of plant models and set the species, height, and other attributes.

## Buildings

You can create different types of 3D building models when you use the building rules. Apply the `Building_From_Footprint.cga` rule to 2D building footprints to generate plausible 3D building models. The `Building_From_OpenStreetMap.cga` rule is automatically applied to 2D polygons downloaded using [Get Map Data](#). This rule is the same as the `Building_From_Footprint.cga` rule but also uses OpenStreetMap data (height, `building:levels`, `roof:height`, `roof:shape`, `building:colour`, `roof:colour`), when available, to determine the building height, roof form, number of levels, building color, and roof color. Apply the `Building_Mass_Texturizer.cga` rule when you have a 3D mass model as your initial shape.



Buildings generated from footprints (first), OpenStreetMap data (middle), and 3D mass models (last) are shown.

## Plants

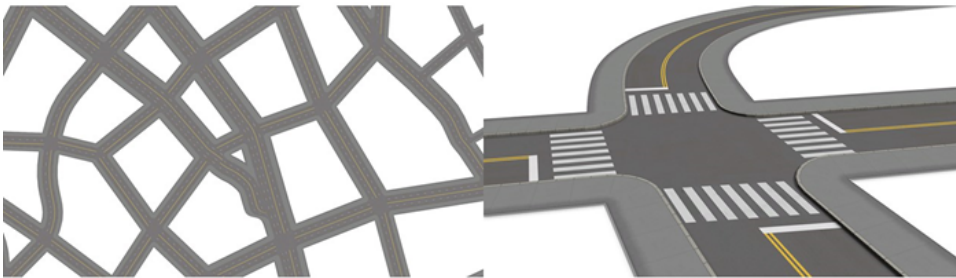
Insert vegetation models into your scene or model. The `ESRI.lib Vegetation` library is a [Webstyle](#) shared across the platform. The library contains a variety of species including broadleaf trees, conifers, and palms available in realistic, schematic, or fan representations. Use the `Plant Loader.cga` rule to insert a single plant model, or use the `Plant_Distributor.cga` rule to scatter multiple plants over an area.



*Realistic (first), schematic (middle), and fan (last) vegetation models are shown.*

## Streets

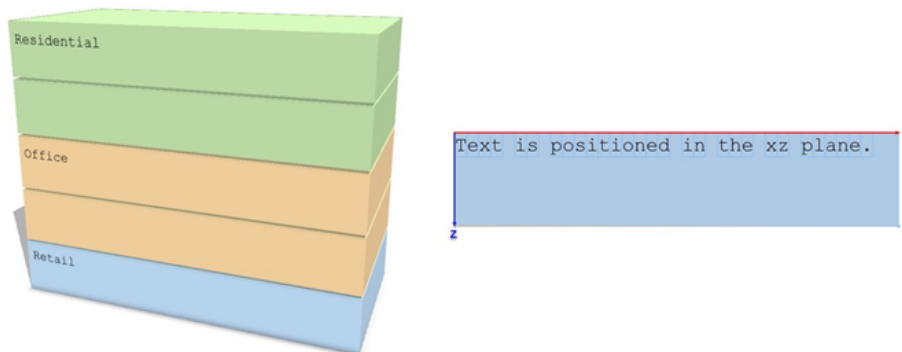
Apply rules to dynamic street shapes to generate textured streets. The `Street_Modern_Simple.cga` rule textures streets with lane markings, while the `Street_Modern_Standard.cga` rule also adds stop markings and crosswalks.



*Textured streets with lanes (first) and textured streets with lanes, stop markings, and crosswalks (second) are shown.*

## Text

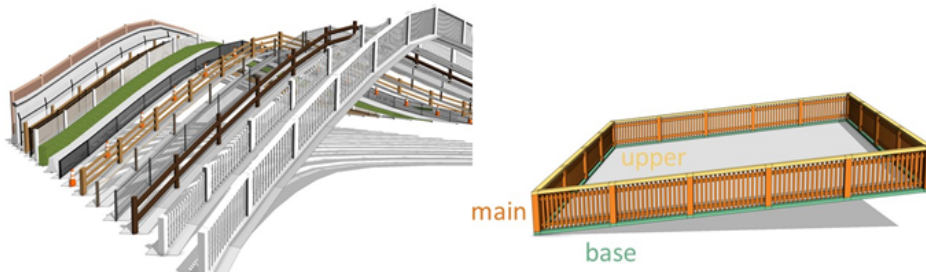
The `Text.cga` rule inserts text you can use for labeling 3D models. Text is inserted in the xz plane of the scope.



*The text rule is applied.*

## Fences

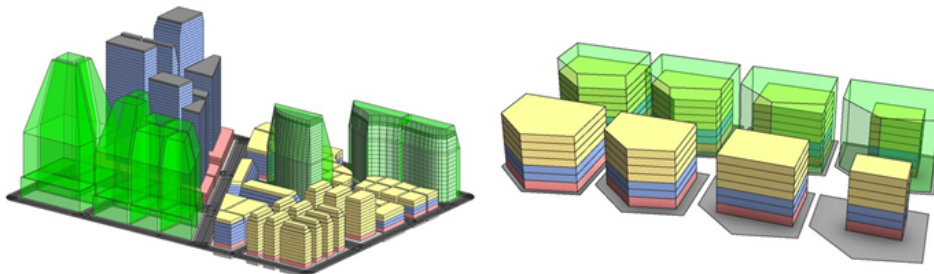
Fence rules generate fences along street shapes or along polygon edges. You can apply the `Fence_On_Graph.cga` rule to street shapes to create fences that follow the street shapes as they curve or go over hills. Apply `Fence_On_Polygon.cga` to polygonal shapes to create fences along the edges of the input polygon. You can control which polygon edges have fences through attributes and [local edits](#). Preset styles are available to create a variety of fences from picket fences to highway barriers.



*Fences generated along street shapes (first) and polygon edges (second) are shown.*

## Urban

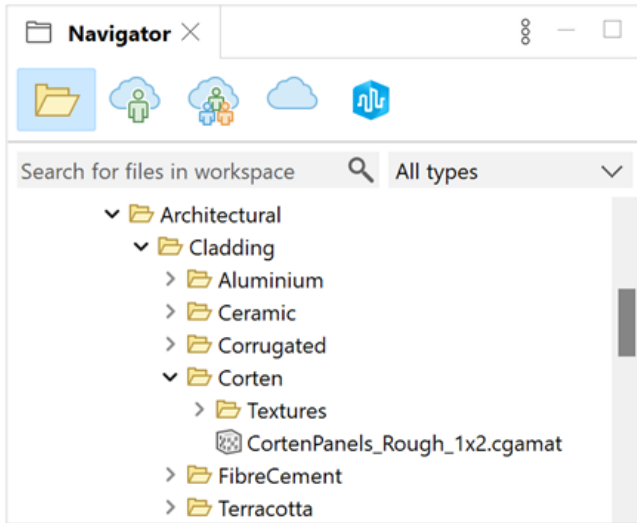
Use the urban rule to visualize zoning regulations and automatically generate buildings following the regulations. You can input zoning regulations such as setbacks and sky exposure planes, lot coverage, and FAR limitations, as well as building dimension constraints. Additionally, you can add building configurations to the rule, specifying footprint shapes, number and heights of floors, and a mass distribution policy. Using these constraints, the urban rule fits a building mass model onto the parcel. You can also use this rule to visualize 3D zoning envelopes given by setbacks and sky exposure planes.



*The urban rule is applied to zoning areas.*

## Materials

Materials are a collection of `cgm` materials for CityEngine. The materials are organized by application, such as `Architectural/Cladding`, and are optimized for seamless tiling and performance. The file name indicates the real-world size in meters, such as 10 m by 10 m.



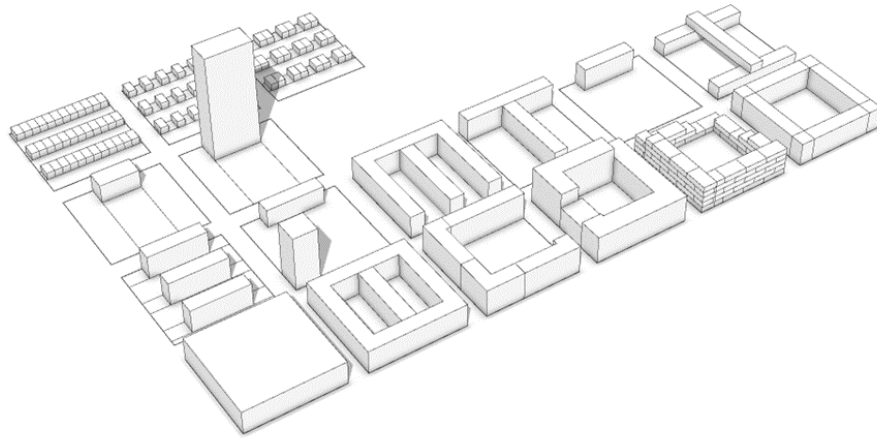
Materials are shown in the Navigator window.

## Components

Components are ready-made CGA rules that can be combined into complete VCGA designs for single buildings or city blocks.

Available component categories include the following:

- **Site partitioning**
  - Allows to encode setbacks from zoning code
  - Input: Parcel
  - Output: BuildableArea
- **Massing**
  - Allows for quick generation of the most common block typologies
  - Organized by block type
  - Input: BuildableArea
  - Output: Mass, Yard



*The massing typology is shown.*

## Webstyles

Webstyles are collections of 3D assets that are shared across the Esri platform. These collections include 3D assets such as vegetation models, vehicles, street furniture, and recreational equipment.



*Webstyles are shown.*

## Manage ESRI.lib

To update ESRI.lib and [Webstyles](#) online, do the following:

1. Click **File** > **Manage ESRI.lib** in the main menu.
2. Select the desired packages.
3. Click **Update**.

The packages are downloaded and directly installed in your current workspace. The [Webstyle](#) assets are located in the ESRI.lib/assets/Webstyles subfolder.

## ESRI.lib considerations

Consider the following when working with the ESRI.lib:

- When opening a workspace with a new version of CityEngine, ESRI.lib is automatically updated.
- Each workspace has a different copy of ESRI.lib unless you have a special setup.
- When working with different versions of CityEngine, it is recommended that you match the versions of the ESRI.lib and [Webstyle](#) libraries. CityEngine usually maintains compatibility, but if the versions do not match, the rules may produce unexpected results.

- To keep the size of the installed `ce.lib` and `ESRI.lib` (including webstyles) minimal, you can set the `CITYENGINE_LIB_MINIMAL` environment variable to `1`. This minimizes disk space consumption in workflows in which there are automatically generated workspaces and projects.

## Custom edits

Editing files inside `ESRI.lib` isn't recommended because updating `ESRI.lib` or CityEngine causes the files to be overwritten. To make custom edits to a rule, first copy the rule to your project folder and then make the changes to your copy. Any references to assets will still refer back to `ESRI.lib`, and therefore, the assets don't need to be copied to your project folder.



### **Note:**

The usage and distribution of `ESRI.lib` is governed by the Esri Master Agreement (EULA) located in the CityEngine installation folder.

# Analysis and Measurement

# Analysis

Analyzing and evaluating a design is an essential part of an iterative design process to ensure that qualitative and quantitative requirements are met. CityEngine includes a variety of tools for this.

## Dashboards

Visualize quantitative information about CGA models using real-time charts. See [Use dashboards](#) for more information.

## Scenarios

You can create multiple variations, or [scenarios](#), of a design proposal in one scene. Dashboards are scenario aware and can display data from multiple scenarios at once.



## Measurement tools


[Measure](#) distances and areas in 3D.

## Visibility analysis

Create Viewsheds and View Domes to identify areas that are visible or invisible from a given viewpoint interactively. Create a View Corridor to identify geometry that blocks the view. See [Visibility analysis](#) for more information.

# Dashboards



Key performance index (KPI) factors determine critical measures when evaluating an urban design proposal. Using CityEngine **Dashboards** , you can create charts of custom-defined key numbers. The **Charts** are driven by CGA report variables. With every change of a CGA model through an attribute change in the **Inspector**, a handle, or a local edit, the **Dashboard**  is updated automatically and always in sync with the 3D model in the **Viewport**.

- A **Dashboard** window  can consist of multiple pages that can hold multiple cards.
- There are multiple card types to choose from: **Key Number**, **Bar Chart**, **Stacked Bar Chart**, and **Pie Chart**.
- Use the **Title** and **Text Card** to provide static information about the charts.
- Cards can be resized and placed freely in a grid system using the mouse.
- The menu in the corner of a card allows you to edit, duplicate, or delete a card as well as copy values to the clipboard.
- Cards can display data coming from various [scenarios](#).

Check out the **Example Philadelphia** in the main menu (**Help > Download Tutorials and Examples**). This scene is a good example of how scenarios, CGA reports, and Dashboards interact.

## Configure a dashboard

To configure a dashboard, complete the following steps:


1. Click **Window > Dashboard** in the main menu to configure a **Dashboard** window .
2. Click the **Add card** tool  in the **Navigation Bar**.
3. Choose a card type.

Dashboards have the following parameters:

Icon & Title	Choose an icon and title.
<b>Report</b>	<ul style="list-style-type: none"><li>• Choose from a list of reports created by the applied CGA rule or rules (for example, Area by flooring type, Area by interior space type, Number of work places, and so on).</li><li>• The drop-down menu has a text box that allows text filtering among the report names.</li><li>• A <b>Key Number Card</b> can display the values of all reports.</li><li>• The <b>Stack Chart</b>, <b>Bar Chart</b>, and <b>Pie Chart</b> only display the values of group reports. To create a group of reports, you must create reports with the syntax <code>Group.SubReport</code>, for example:<ul style="list-style-type: none"><li>▪ FAR.Office</li><li>▪ FAR.Resdential</li><li>▪ FAR.Commerical</li></ul></li></ul>

<b>Divide by</b>	<ul style="list-style-type: none"> <li>• Divide a report by another report.</li> <li>• By using the reportName.* notation, corresponding subreports (with the same subreport name) can also be divided.</li> </ul>
<b>Multiply by</b>	Multiply the values of a report by a constant number.
<b>Reporter</b>	<p>You can choose to display the values of the reports computed on the entire scene or only on the selection. By default, a card displays the values on the entire scene.</p> <ul style="list-style-type: none"> <li>• <b>All Objects</b>—Default, means values from all objects in the scene are aggregated.</li> <li>• <b>Selected Objects</b>—Only values from selected objects are aggregated.</li> </ul>
<b>Aggregation</b>	<p>The aggregation type defines how to combine all the reported values from the whole scene into a single number for each report or subreport, that will then be shown in the chart:</p> <ul style="list-style-type: none"> <li>• Sum</li> <li>• Count</li> <li>• Mean</li> </ul>
<b>Unit</b>	The <b>Unit</b> displayed next to the value in the card. Has no influence on the calculation or the value displayed.
<b>Minimum and Maximum</b> (for Bar Chart)	<p>Defines the range of the value axis.</p> <ul style="list-style-type: none"> <li>• <b>Automatic</b>—Adjusts value automatically based on the displayed values.</li> <li>• <b>Custom</b>—Uses a fixed user-defined value.</li> </ul>
<b>Scaling</b> (for Bar Chart)	<p>Specifies the scaling of the axis:</p> <ul style="list-style-type: none"> <li>• Uniform (by default)</li> <li>• Logarithmic</li> </ul>
<b>Notation</b>	<ul style="list-style-type: none"> <li>• Decimal (for example, 48'876'592)</li> <li>• Scientific (for example, 4.888e+7)</li> </ul>
<b>Round values</b>	<b>Decimal places</b> —Specifies the number of decimals to display.
<b>Sorting</b> (Pie Chart)	<p>Sorting of reports:</p> <ul style="list-style-type: none"> <li>• <b>Alphabetical</b>—(Report names)</li> <li>• <b>Ascending</b>—Lowest value first, highest last</li> <li>• <b>Descending</b>—Highest values first, lowest last</li> </ul>
<b>Legend</b> (Stack Charts and Pie Charts)	<ul style="list-style-type: none"> <li>• Display a legend in charts.</li> <li>• The legend adjusts automatically to the reported values.</li> </ul>

## Colors

The colors of the reports in the **Dashboard** window  are, by default, defined by the color of the **scenarios** set in the **Scene Editor** window. It is possible to specify a color for each report in a CGA file by the hashtag #color, for example:

```
Lot -->
    report("report_name",1.0)
    report("report_name#color", "#ff0000")
```

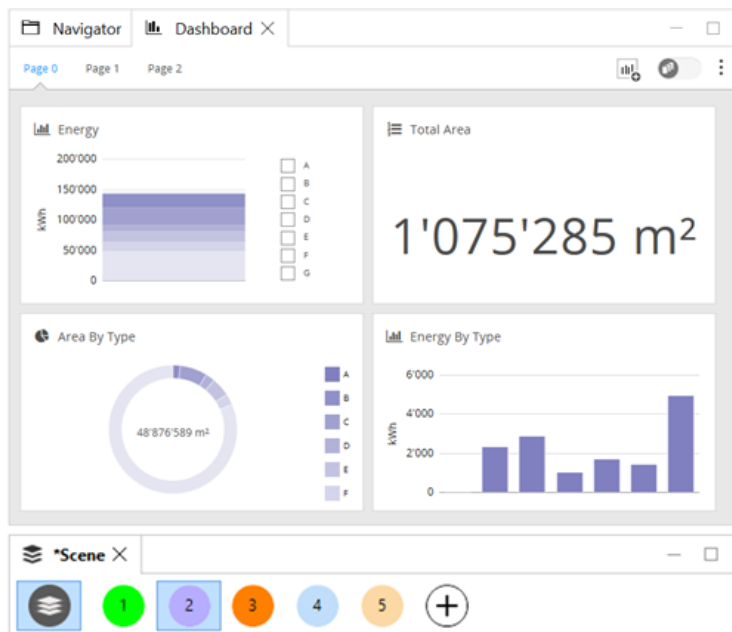
## Dashboards and scenarios

Two modes are available in the **Dashboard** window  concerning the data to display:

- Active Scenario
- Compare Scenarios

Switching from one mode to another is done by clicking the button in the navigation bar. In the Active Scenario mode, the cards will only display the report values of the object belonging to the scenario activated in the **Scene Editor** window.

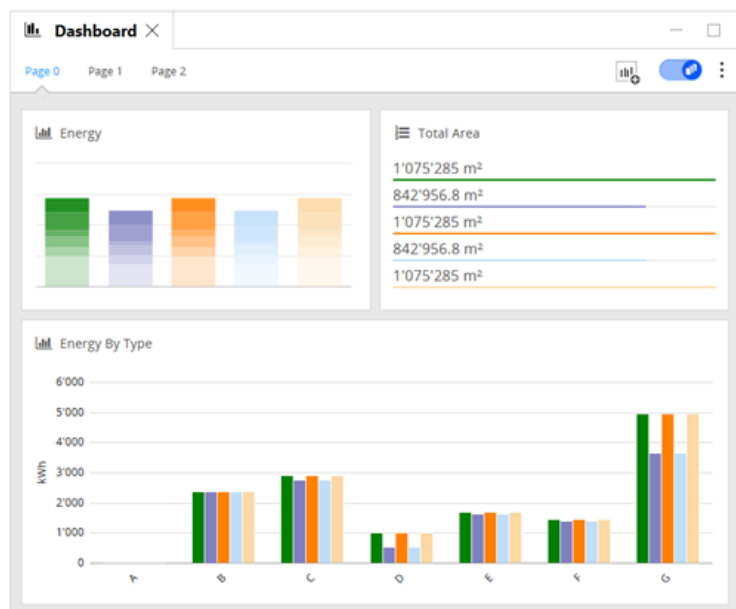
### Active scenario mode



Active scenario mode is shown.

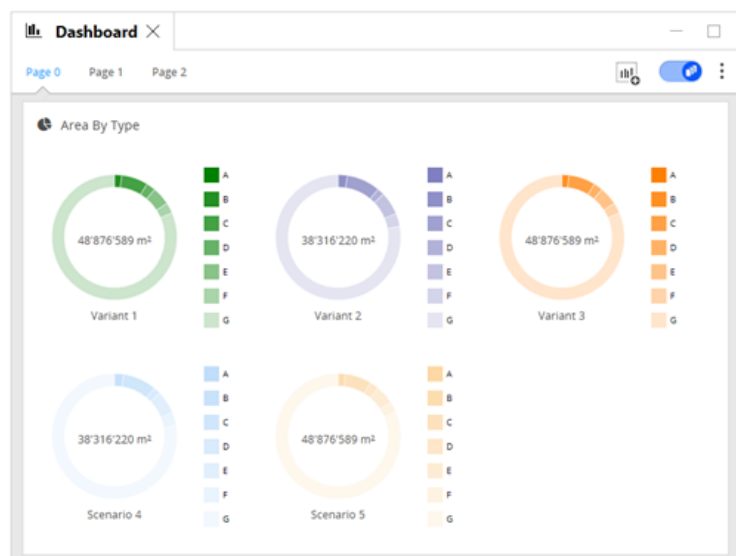
### Compare Scenarios mode

In the **Compare Scenarios** mode, the cards will display the report values of all the scenarios of the scene:



Compare Scenarios mode

In the **Compare Scenarios** mode, the **Pie Card** displays one pie chart per scenario:



Compare scenarios with pie charts.

For more information about scenarios, see [Scenarios](#).

## Manage the Dashboard


It is possible to create many pages in the **Dashboard** window. Click the **Options** tool on the **Navigation Bar** to display a menu with the following options:

- **Add pages**—Add a new page where you can set the name.
- **Rename Current Page**—Rename the current page.
- **Duplicate Current Page**—Duplicate the current page.
- **Delete Current Page**—Delete the current page (except if there is only one page in the dashboard).

- **Table View**—Display the values of reports by scenario in a table.

The **Table View** displays for each report its names, aggregates, and the color when it is specified in the CGA file. With the three drop-down menus on top of the Table View, it is possible to filter the reports by the following:

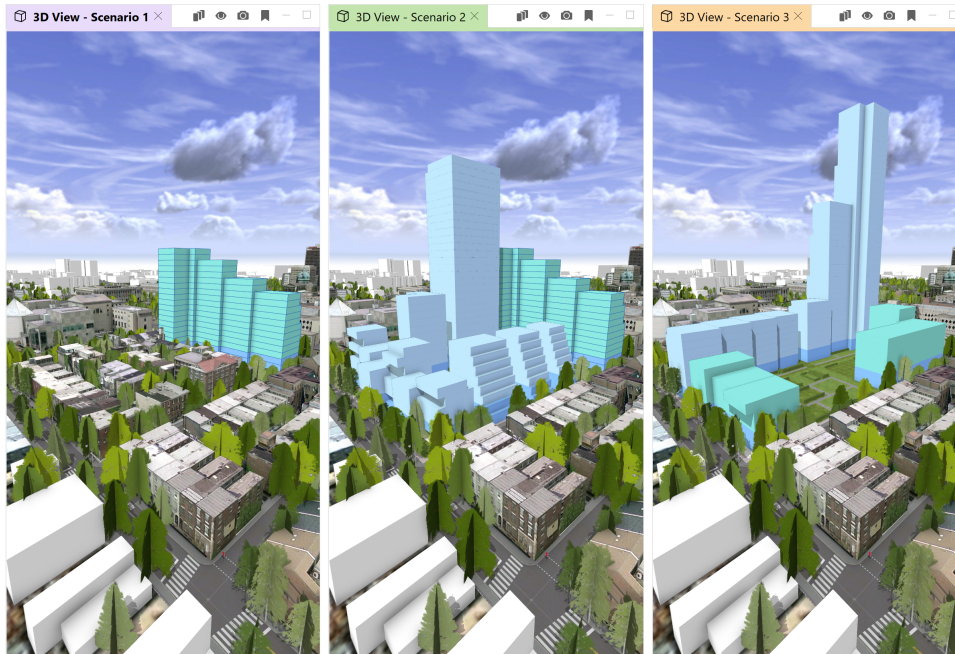
- Report or Group of Reports
- Scenario
- All Scene or Selected Objects



Report	Sum	Percent Total	Percent	Avg	Max	Min	N	NaNs	Color
GFABByClass.A	39.84	0.3650	0.1029	19.92	23.54	16.30	2.000	0.000	#00a651
GFABByClass.B	25527.2	84.49	65.96	55.13	95.54	40.84	463.0	0.000	#4cb848
GFABByClass.C	3961.5	14.42	10.24	50.15	85.85	42.78	79.00	0.000	#bed62f
GFABByClass.E	9172.1	0.7299	23.70	2293.0	2699.2	1074.5	4.000	0.000	#fdb813

# Scenarios

Scenarios allow you to create multiple designs within a single scene and then compare them. Scenarios can be displayed side by side in different views for visual comparison. [Dashboards](#) are also scenario aware and are able to show the values for multiple scenarios simultaneously, allowing you to compare the performance of designs against each other.



3D Viewports side by side with different scenarios


Scenarios consist of layers, containing objects such as buildings, streets, or terrain, that only display when the scenario is active. You can make custom changes that affect only a specific scenario or you can make global changes that can be applied across all scenarios. They provide a way to store multiple design alternatives in a single scene. Furthermore, working with different scenarios is supported by [Unreal Engine](#), [360 VR experience](#), and [ArcGIS Urban](#).

## Create scenarios

In the **Scene Editor**, you can create and manage scenarios.

### Add a new scenario

To add a new scenario, do the following.

1. Click the **Add new scenario** button  to create a scenario.
2. Provide the **Name**, **ID**, and **Color**.
3. Click **OK**.

### Manage scenarios in Scene Editor

Manage scenarios on the **Scene Editor** toolbar. Change the order, edit properties, and remove scenarios. When you right-click a scenario, you get the following options:

<b>Duplicate</b>	See <a href="#">Duplicate</a> .
<b>Mirror</b>	See <a href="#">Mirror</a> .
<b>Edit</b>	Change the scenario name, ID, and color.
<b>Delete</b>	Remove the scenario.
<b>Move right</b>	Move the scenario to the right.
<b>Move left</b>	Move the scenario to the left.

 **Tip:**

You can also change the order of scenarios by dragging them left or right with the mouse.

Duplicate or mirror a scenario

You can use **Scene Editor** to duplicate and mirror existing scenarios. Duplicating a scenario creates a unique copy of that scenario that is independent of the original scenario. Mirroring a scenario creates a scenario that is linked to the original scenario and contains all its layers.

*Duplicate*

To duplicate a scenario, do the following:

1. In **Scene Editor** , right-click a scenario to duplicate.
2. Click **Duplicate**
3. Provide a **Name**, **ID**, and **Color**.
4. Click **OK**.  
A scenario is created in which all objects and layers are copies with unique IDs.

*Mirror*

To mirror a scenario, do the following:

1. In **Scene Editor** , right-click a scenario to mirror.
2. Click **Mirror**
3. Provide a **Name**, **ID**, and **Color**.
4. Click **OK**.  
Each scenario is identical and a clone of the other. Any edits made to a layer in one scenario are applied equally to the other.

Add objects and layers to scenarios

To add object and layers to a new or existing scenario, do one of the following:



- Select objects in the scene and copy them to a scenario.
- Select default objects or layers in the [Scene Editor](#) and copy them to a scenario. Default objects are the original objects in layers, such as terrain, streets, and buildings, that make up the scene.
- Copy objects or layers from one scenario to another.

Click a scenario to make it the active scenario in the [Scene Editor](#). Click the **Default Objects** button  to turn the default objects layers on and off.

 **Tip:**

Similarly to the **Duplicate** action, when you copy objects or layers to a scenario, you create a unique copy of the objects and layers that is independent of the original scenario membership.

## Change the active scenario in the Viewport

The active scenario is the scenario that is selected in the [Scene Editor](#). The Default Objects layers will be visible by default. In each **Viewport** window, you can select which scenario is active in the [Viewport](#). Click **Scenarios**  > **Active Scenario** in the **Viewport** to display the active scenario set in the **Scene Editor**. Or click **Scenarios**  and select a specific scenario to make it active.

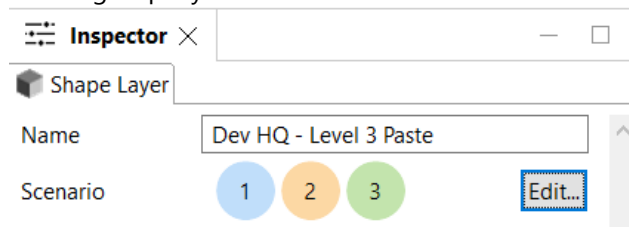
 **Tip:**

- The Viewport window displays the name and color of the scenario.
- You can choose how to display scenarios in the **Viewport** windows through the [Windows main menu](#).

## Manage scenarios in the Inspector

You can manage scenario group layers in the **Inspector** by changing the object type of the group layer from default object to scenario object; and you can select which scenarios contain the group layer.



1. Click a group layer in the **Scene Editor**.





2. Click **Edit** to open the **Edit Scenario Membership** dialog box, which indicates the membership status of the group layer.
  - a. Under **Object Type**, choose **Default Object** or **Scenario Object**.
  - b. Under **Scenarios**, choose the scenarios of which the group layer is a member.

When you add the membership of a layer to another scenario, it has the same effect as when you mirror a scenario. If you remove a default object layer from a scenario, [Inspector](#) indicates that a scenario has been removed from the default object layer with an asterisk \*.

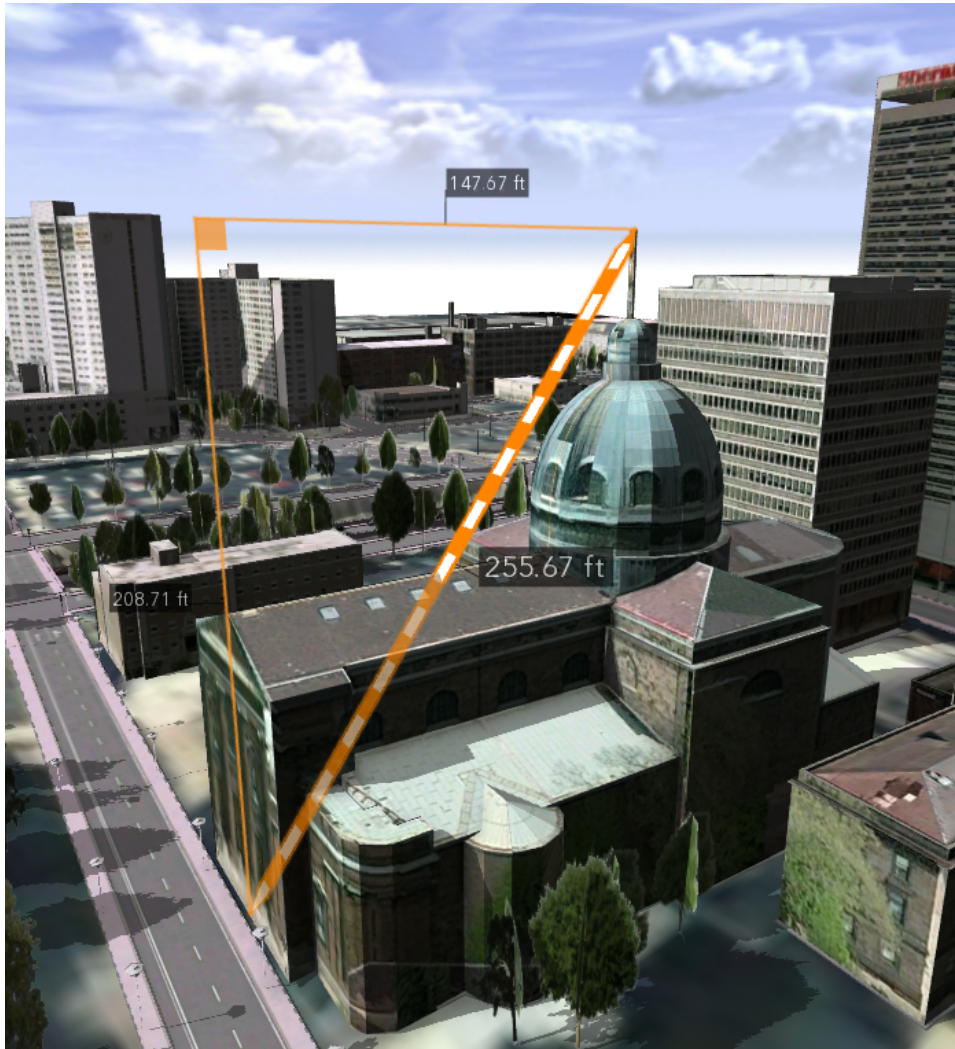
# Measure tools

You can measure distance with the **Measure Distance** tool  and areas and paths with the **Measure Area and Path** tool .

## Measure distance

To measure distance, click the **Measure Distance** tool  (M+D) to open the **Tool Options** window . You can also click **Analysis > Measure Distance** in the main menu.

Click in the scene to place the starting point. Click again to place the endpoint.



Line of sight, vertical, and horizontal distance are measured. Note: The line of sight shines through the building.



## Measure Distance tool options



The **Measure Distance** tool options ✂ include the following:

✂ <b>Measure Distance</b>	
<b>Diagonal distance</b>	Diagonal distance between the points.
<b>Horizontal distance</b>	Horizontal distance between the points.
<b>Vertical distance</b>	Vertical distance between the points.
<b>Show laserlines</b>	Turn laser lines on and off. The laser lines project the current height of the mouse onto the surrounding geometry.

## Measure area and path

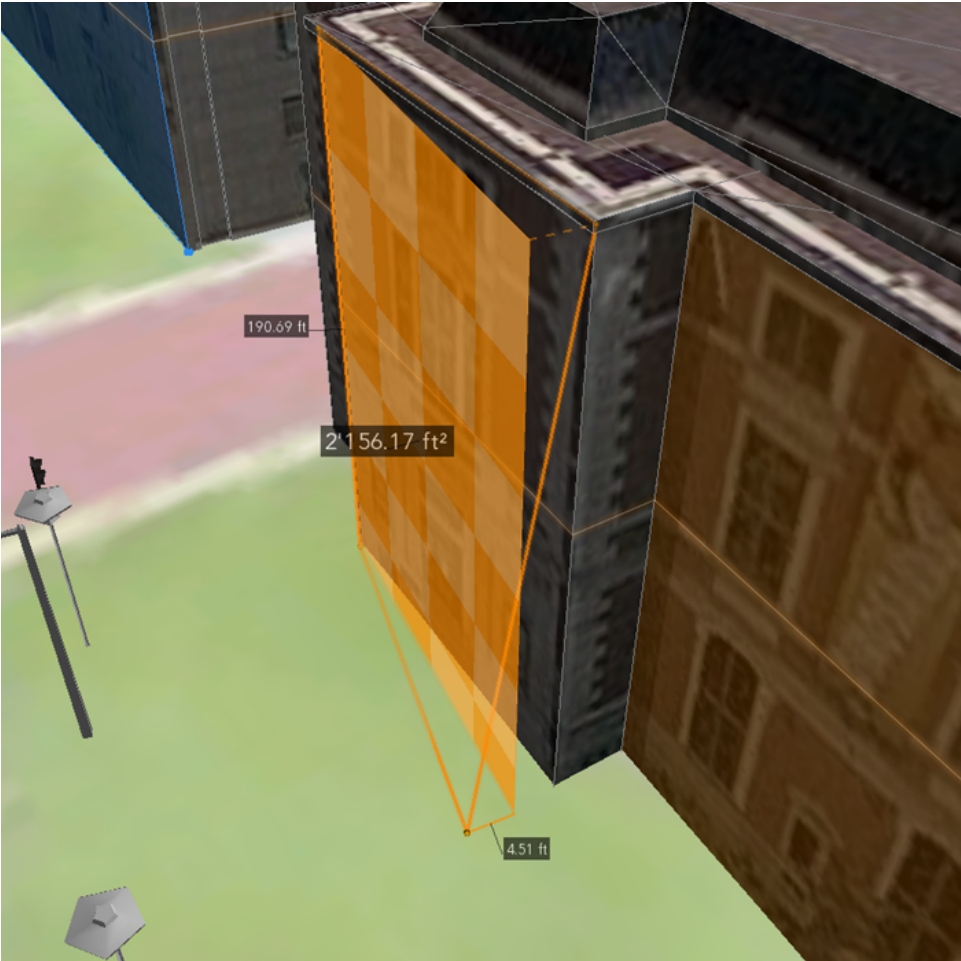
To measure area and path, click the **Measure Distance** tool  and click the **Measure Area and Path** tool  (M+A) in the **Tool Options** window ✂. You can also click **Analysis > Measure Area and Path** in the main menu.

To measure, click in the scene to place the starting point. Add additional points to calculate the length of the path or click the starting point or press **Enter** to close the path. Press **Ctrl+Z** to undo the last measure point.

The measurement points are not required to be part of the same plane. If they are not in the same plane, the points are projected onto a horizontal or vertical plane at an average height or depth automatically.

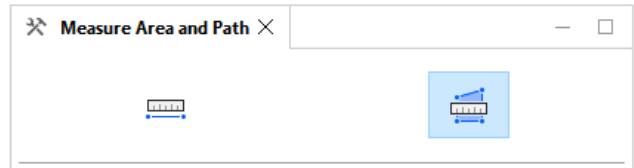
### **Note:**


- In this case, the displayed surface measure relates to the projected area, while the displayed distance always refers to the perimeter defined by the measurement points.
- The projected plane is positioned so that the distance to the measurement points is minimized. The measurement point with the largest distance to the plane is marked with a label displaying the distance.




The measurement points are not coplanar and are therefore projected onto a vertical plane. At 4.51 feet, the lower right measurement point is the most distant from the plane.

Measure Area and Path tool options




The **Measure Area and Path** tool options  include the following:

 <b>Measure Area and Path</b>	
<b>Perimeter</b>	Perimeter length of the polygon.
<b>Area (projected)</b>	Area of polygon.
<b>Show laserlines</b>	Turn laser lines on and off. The laser lines project the current height of the mouse onto the surrounding geometry.

Measure tool considerations

The following list describes measure tool functionality:

- When placing measurement points, the pointer snaps to vertices, edges, faces, and guides.
- The measurement tools allow for parallel and perpendicular [snapping](#).
- Only one measurement can be displayed in a scene at a time. When starting a new measurement, the current one disappears.
- Measurements disappear after switching to a different tool. This also applies for the navigation tools in the toolbar. The navigation shortcuts can be used between placing measurement points and after finishing a measurement.
- To reset the tool, press **Esc**.
- To change the units, click in the [Viewport](#) and go to **View Settings > View Coordinate System**. The change takes effect with the next measurement you start.
- To adjust a completed measurement, hover over the measurement points. Use the arrows to move the measurement point along the axes, or use the orange ball to move freely and snap to vertices, edges, faces, or guides.
- You can copy values from any of the fields in the **Tool Options** window .

# Visibility analysis




# Visibility analysis

Visibility analysis tools highlight surfaces and structures that are visible or hidden from a user-defined observer position.

To open the visibility analysis tools, click the **Viewshed Creation** tool . You can switch the active tool in the **Tool Options** window .


You can also open the tools from the main menu by doing one of the following:

- Click **Analysis > Viewshed Creation**.
- Click **Analysis > View Dome Creation**.
- Click **Analysis > View Corridor Creation**.

 <b>Viewshed Creation</b>	Calculates a <b>viewshed</b> that determines the visibility from a camera-like observer for a limited field of view.
 <b>View Dome Creation</b>	Calculates a <b>view dome</b> that has the same functionality as the <b>Viewshed Creation</b> tool but provides a 360° field of view.
 <b>View Corridor Creation</b>	Creates a protected <b>view corridor</b> where any geometry visible in the corridor is highlighted.

By using these tools, you can create **Analyses objects** that can be saved in your CityEngine scene. To learn more about Analyses objects in the **Inspector** and **Scene Editor**, see the **Manage analysis layers** section below. The visibility analysis tools snap to buildings and terrains to facilitate the positioning of the observer point and the point of interest.

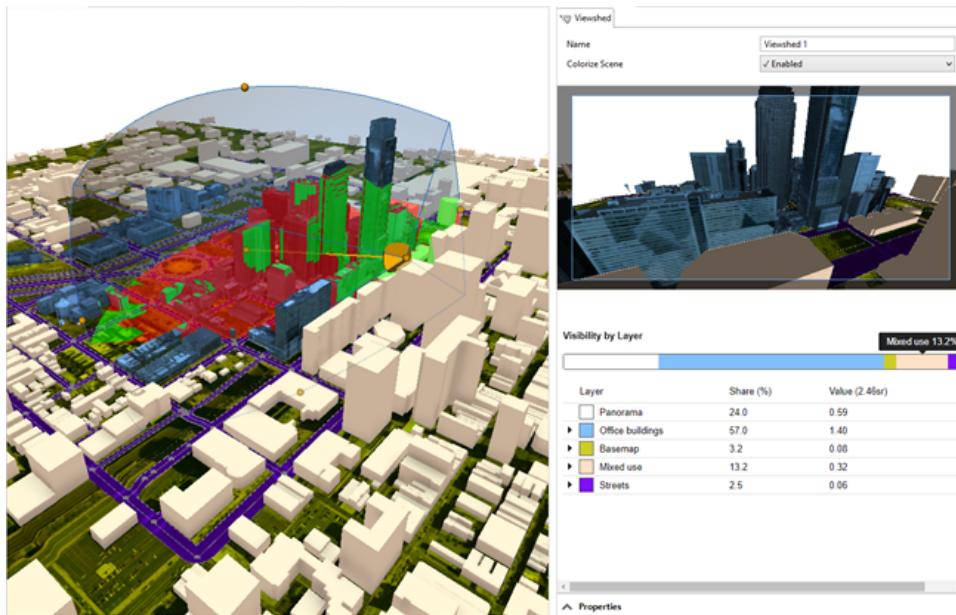
## Common features

- All three tools are represented as scene objects that belong to an **Analysis Layer**.
- Analysis tools are persistent and therefore are saved with the scene.
- In case the geometry in the scene or the analysis tool itself is edited, the recalculation is triggered automatically for all visibility objects.
- The visual representation in the **Viewport** provides handles to edit the position of the observer as well as view direction and field of view.
- You can edit the **Analyses** objects with the **transform tools**.
- The **Inspector** provides a perspective view rendered from the observer position.
- The **Visibility settings** tool  allows you to show or hide all analysis objects.

## Visibility by layer

The **Viewshed Creation** and **View Dome Creation** tools provide information on how much each scene layer contributes to the geometry visibility in the specified field of view. This information is presented as a horizontal bar chart and as a table in the **Inspector** when the **Analyses** object is selected. The table provides the information as percentages of the specified field of view and as solid angles expressed as steradians(sr).

Besides the layers as defined in the **Scene Editor**, an additional category Panorama is shown. Panorama applies to all nongeometry areas in the field of view including the sky and areas below the horizon not covered by terrain or geometry.



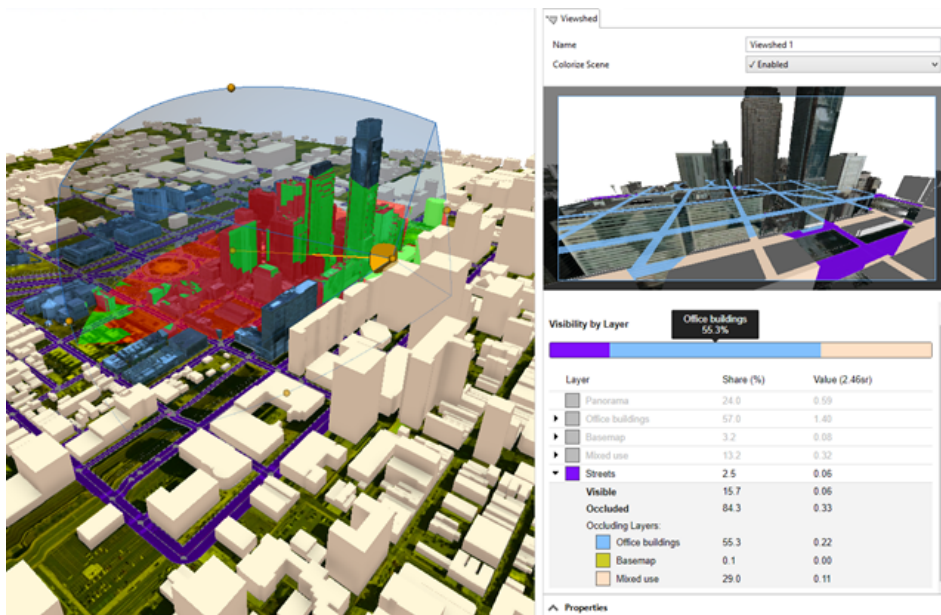
Viewshed in the Viewport with colored geometry and the Inspector with the bar chart and the table view of the statistics are shown.

#### **Note:**

Only geometry inside the viewshed or view dome body is taken into account for the calculations. Geometry farther away than the view distance is not colored in the [Viewport](#), not visible in the [Inspector](#) view and also not taken into account when the visibility statistics are calculated. These directions are therefore counted as Panorama.

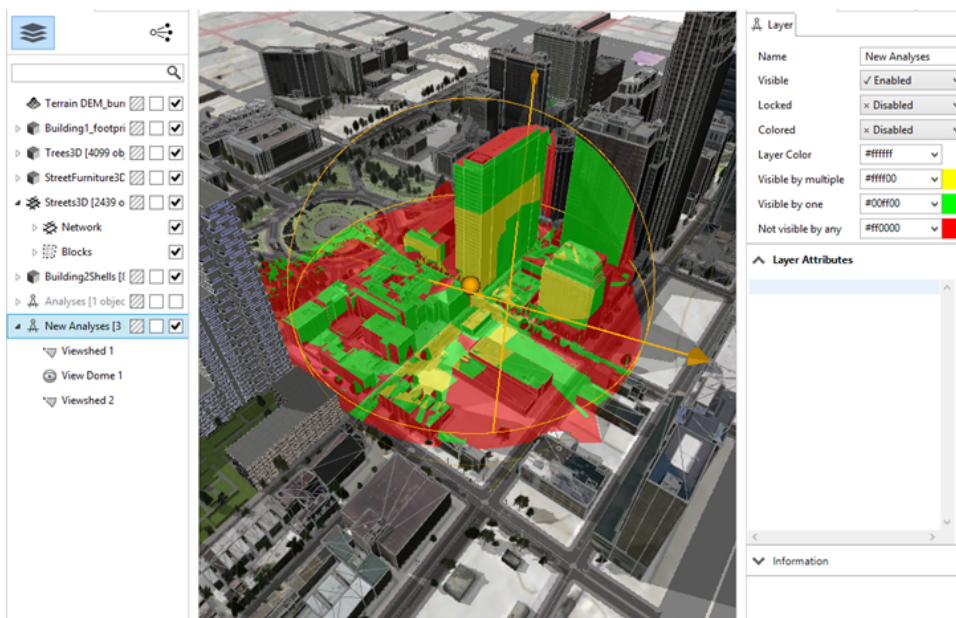
For further analysis, you can expand a layer to see how much of it is visible or occluded within the viewshed or view dome. Furthermore, you get a list of layers that contribute to the occlusion of the expanded layer including percentages and sr values. In the image below, 15.7 percent of the Streets layer is visible and 84.3 percent is occluded by other layers. Additionally, you can see the Office buildings layer is the main occluder of the Streets layer with a share of 55.3 percent.

You can use visibility analysis in [scenarios](#) as well.



View Dome: expanded Layer Streets reveals that about half of its geometry in the field of view is covered by the Buildings layer.

## Manage analysis layers



Two Viewsheds overlapping with a View Dome. Note the Inspector settings for the Analysis layer.

By putting overlapping viewsheds and view domes into the same analysis layer, the coloring of the geometry is cumulated. You can specify the color scheme in the [Inspector](#) of the Analysis Layer as follows:


- **Visible by multiple**—This color is applied to geometry that is visible from more than one observer point.
- **Visible by one**—This color is applied if the geometry is visible from one observer point only.
- **Not visible by any**—This color is applied if the geometry is not visible by any observer point. This color is also used to highlight geometry that violates a view corridor.




**Tip:**

It is recommended that you put view corridor objects in a separate **Analysis Layer**, as they cannot interact with the viewshed or view dome objects.

# Viewsheds


Viewsheds are visualizations of what is visible from a given point and are often used in urban planning. Use the **Viewshed Creation** tool  to determine the visible area from an observer point to a point of interest. The tool colors areas that are visible (by default in green) or occluded (by default in red) from the observer. The viewshed object you created can be managed in the [Scene Editor](#).

You can open the **Viewshed Creation** tool  the following ways:

- Click the **Viewshed Creation** tool .
- Click **Analysis > Viewshed Creation** in the main menu.

## Create a viewshed

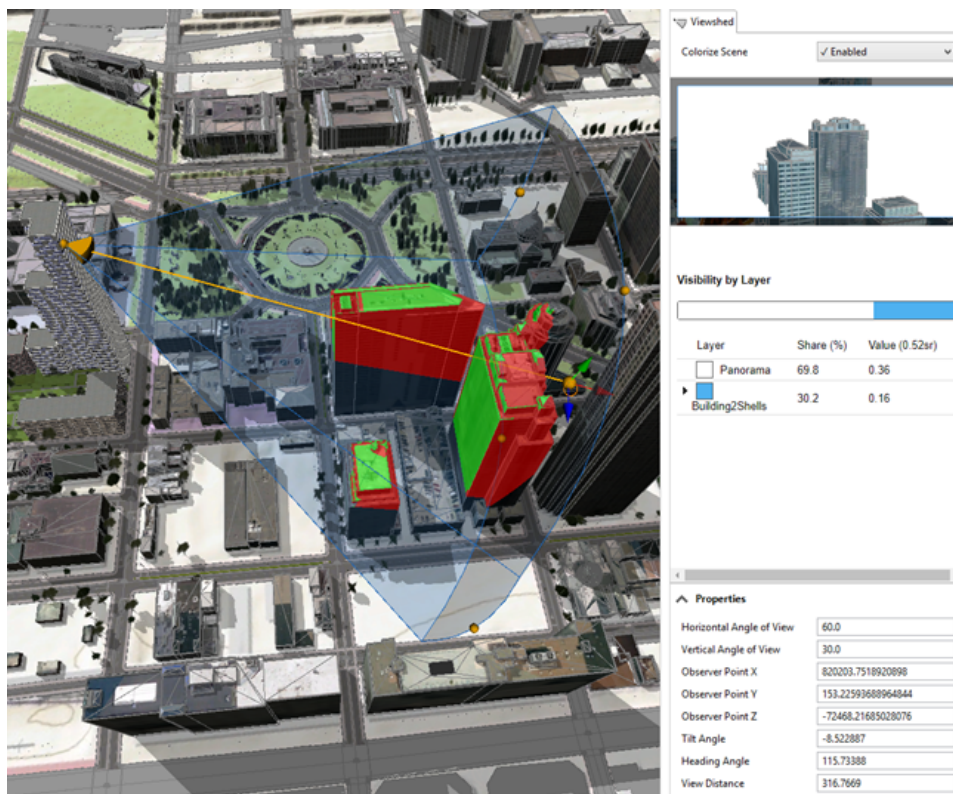
To create a viewshed, do the following:

1. Click a point in the [Viewport](#) to create an observer point.  
This starts the **Viewshed Creation** tool , which dynamically displays the viewshed as you move.

2. Drag the tool to set the point of interest.

3. Click again to anchor the **Point of Interest** property in the viewshed.


The viewshed is added to the [Scene Editor](#) and is automatically selected. You can directly adjust its properties, such as the **Angle of View** and **View Distance** parameters, in the [Viewport](#) using handles or in the [Inspector](#) using input fields.



## Tool options



The **Viewshed Creation** tool options  include the following:

 <b>Viewshed Creation</b>	
<b>Horizontal Angle of View (°)</b>	The value determines the <a href="#">horizontal angle of view</a> for the next view corridor placed. Once placed, you can <a href="#">adjust the viewshed</a> .
<b>Vertical Angle of View (°)</b>	The value determines the <a href="#">vertical angle of view</a> for the next view corridor placed. Once placed, you can <a href="#">adjust the viewshed</a> .

## Adjust the viewshed

You can adjust the viewshed in the following ways:

- In the [Viewport](#), you can drag any of the orange handles to modify the **Observer point**, **Point of Interest**, or **Angle of View** properties.
- In the [Inspector](#), you can edit any of the properties.

## Properties

The following is a list of the **Viewshed Creation** tool properties in the [Inspector](#):

<b>Horizontal Angle of View</b>	Horizontal angle of view, or field of view, from the observer
<b>Vertical Angle of View</b>	Vertical angle of view, or field of view, from the observer
<b>Observer Point X</b>	X-coordinate of the observer
<b>Observer Point Y</b>	Y-coordinate of the observer
<b>Observer Point Z</b>	Z-coordinate of the observer
<b>Tilt Angle</b>	Camera view angle from -85 to 85 degrees
<b>Heading Angle</b>	Camera view angle from -360 to 360 degrees
<b>View Distance</b>	Distance between the observer and the point of interest


### **Tip:**

The **Colorize Scene** property in the [Inspector](#) allows you to choose whether to display the colored visible or hidden areas in the [Viewport](#). The preview shows you the view from the observer point with the current properties, enabling an intuitive placement of the viewshed.




## Analyze by layer

The [Inspector](#) displays visibility statistics about the current viewshed and breaks the visibility down by layer. See [Visibility by layer](#) for details.

# View domes

The **View Dome Creation** tool  allows you to see the visibility from a given point in a 360-degree view. The **View Dome Creation** tool colors the areas that are visible (by default in green) or occluded (by default in red) from the observer. The view dome object you created can then be managed in the **Scene Editor**.

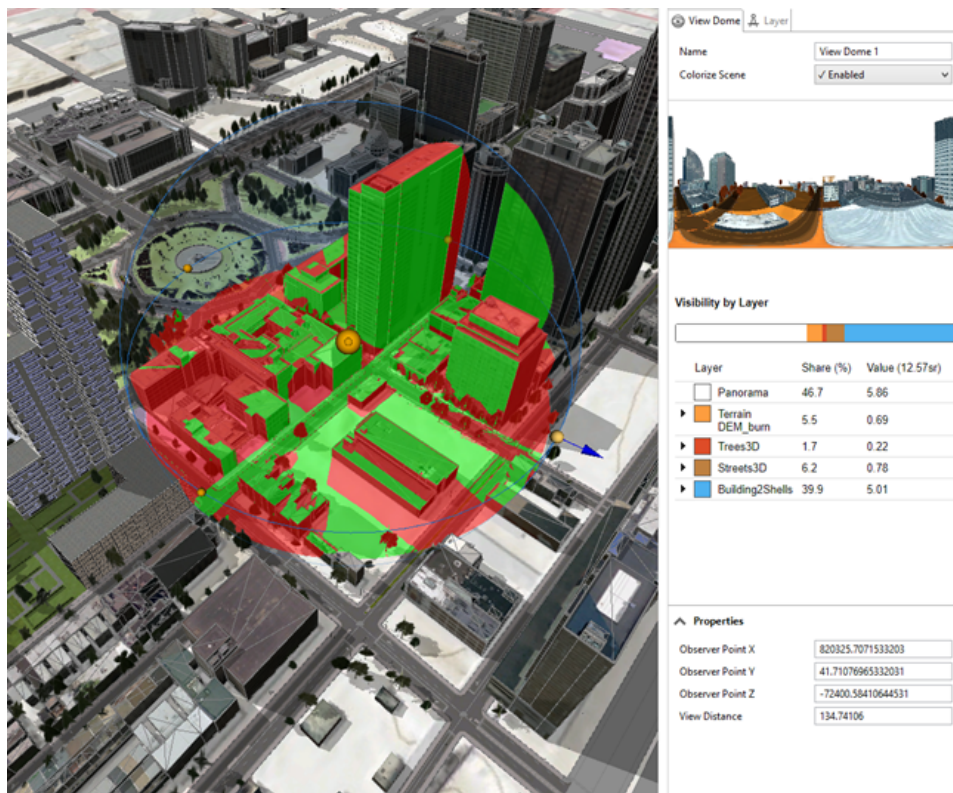
You can open the **View Dome Creation** tool  the following ways:

- Click the **Viewshed** tool  and click the **View Dome Creation** tool  in the **Tool Options** window .
- Click **Analysis > View Dome Creation** in the main menu.

## Create a view dome

1. Click a point in the **Viewport** to create an observer point.
2. Drag the **View Dome Creation** tool to create the visibility analysis.

The view dome is added to the **Scene Editor** and is automatically selected. You can directly adjust its properties, such as the view distance in the **Viewport**, using handles or the **Inspector** using input fields.



## Adjust the view dome

The **Colorize Scene** property allows you to choose whether to display the colored visible or hidden areas of the view dome in viewports. The preview shows you the 360-degree view (using an equirectangular projection) from the current observer point.

You can adjust the view dome in the following ways:

- In the **Viewport**, you can click and drag any of the orange sphere handles in the view dome to modify the **Observer Point** or the **View Distance**.
- In the **Inspector**, you can edit any of the properties.

### Properties

The following is a list of the view dome properties in the **Inspector**:

Observer Point X	The x-coordinate of the observer
Observer Point Y	The y-coordinate of the observer
Observer Point Z	The z-coordinate of the observer
View Distance	The distance between the observer and the point of interest

 **Tip:**

The **Colorize Scene** property in the **Inspector** allows you to choose whether to display the colored visible or hidden areas of the view dome in the **Viewport**. The preview shows you the view from the **Observer Point** with the current properties allowing you to accurately place the view dome.



### Analyze by layer

The **Inspector** displays visibility statistics about the current view dome and breaks the visibility down by layer. See [Visibility by layer](#) for details.

## View corridors


View corridors protect established views from a given observer against building development obstruction. For example, there are multiple view corridors in the city of Seattle that keep the Space Needle clear of buildings. The use of view corridors is a well-known and accepted practice to keep views of landmarks from being blocked or compromised.

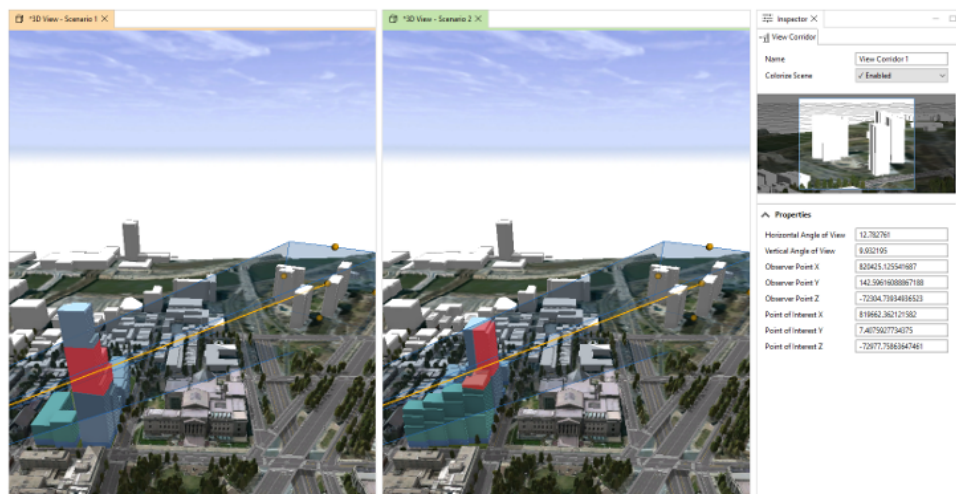
You can open the **View Corridor Creation** tool  the following ways:


- Click the **Viewshed** tool  and click the **View Corridor Creation** tool  in the **Tool Options** window .
- Click **Analysis > View Corridor Creation** in the main menu.

### Create a view corridor


To create a view corridor, do the following:

1. Click a point in the **Viewport** to create an observer point.  
This starts the **View Corridor Creation** tool , which dynamically displays the view corridor as you move.
2. Drag the tool to set the point of interest.
3. Click again to anchor the **Point of Interest** property in the view corridor.  
The view corridor is added to the **Scene Editor** and is automatically selected. You can directly adjust its properties, such as the **Angle of View** and **View Distance** parameters, in the **Viewport** using the handles or in the **Inspector** using input fields.




All parts of scenario objects that lie in the view corridor and are visible from the observer point are colored with a chosen color (the default is red). To see the Scenario A objects in a view corridor, you can activate Scenario A with the **Scenario** tool  in the **Viewport** tool. See [Scenarios](#) for more information.


#### **Tip:**

A view corridor is designed to analyze [scenarios](#). It is recommended that you create scenarios before creating objects, because only buildings in a scenario are colored by the **View Corridor Creation** tool .

## Tool options



The **View Corridor Creation** tool options  include the following:

 <b>View Corridor Creation</b>	
<b>Horizontal Angle of View (°)</b>	The value determines the <a href="#">horizontal angle of view</a> for the next view corridor that is placed. Once placed, you can <a href="#">adjust the view corridor</a> .
<b>Vertical Angle of View (°)</b>	The value determines the <a href="#">vertical angle of view</a> for the next view corridor that is placed. Once placed, you can <a href="#">adjust the view corridor</a> .

## Adjust the view corridor

You can adjust the view corridor in the following ways:

- In the [Viewport](#), you can drag any of the orange handles to modify the **Observer point**, **Point of Interest**, or **Angle of View** properties.
- In the [Viewport](#), you can edit any of the properties.

## Properties

The following is a list of the **View Corridor Creation** properties in the [Inspector](#):

<b>Horizontal Angle of View</b>	Horizontal angle of view, or field of view, from the observer
<b>Vertical Angle of View</b>	Vertical angle of view, or field of view, from the observer
<b>Observer Point X</b>	X-coordinate of the observer
<b>Observer Point Y</b>	Y-coordinate of the observer
<b>Observer Point Z</b>	Z-coordinate of the observer
<b>Point of interest X</b>	X-coordinate of the point of interest
<b>Point of interest Y</b>	Y-coordinate of the point of interest
<b>Point of interest Z</b>	Z-coordinate of the point of interest

### **Tip:**

The **Colorize Scene** property in the [Inspector](#) allows you to choose whether to color the scenario objects that are in the view corridor. The preview helps the placement of the view corridor and shows which active scenario objects alter the protected view.

# Online and Enterprise

# ArcGIS Online and ArcGIS Enterprise

[ArcGIS Online](#) and [ArcGIS Enterprise](#) are online platforms to share maps and geographic information. You can share various types of geographic information: maps, map layers, features, editing templates (using layer packages), imagery, and analytic results. Once you sign in to ArcGIS Online, you can use your own personal online workspace in the cloud, where you can share data packages as well as search for and find others' content, which you can use across the ArcGIS platform.

## Sign in

[Sign in](#) to your organization to get access to your content, group content, and your organization's content in your project. You can sign in to additional portals to use their content as well.

## Configure redirect URIs

You can [configure redirect URIs](#) on an ArcGIS Enterprise portal

## Get map data

[Get map data](#) from ArcGIS Online to add basemaps to a CityEngine scene. You can choose from several types of basemaps and select an area to download into a scene.

## Sync feature layers

[Sync feature layers](#) by editing ArcGIS Online polygon feature layers in CityEngine and synchronizing the updates to ArcGIS Online.

## Share data from CityEngine

In CityEngine, you can [share data](#) from web scenes, Esri scene layer packages, tile packages, rule packages and 360 VR Experiences to ArcGIS Online and ArcGIS Enterprise. CityEngine guides you through the process of preparing, packaging, and sharing the information.

# Sign in

Sign in to your organization for access to your content, group content, and your organization's content in your project. You can sign in to additional portals to use their content as well.

You can sign in to ArcGIS Online and ArcGIS Enterprise by either signing in from the main menu or from the sign-in menu on the CityEngine toolbar. If you don't have an ArcGIS Online account, see [Get new account](#) to create a public account.



## Note:

To access **All Portal** items in the **Navigator** and share your data with your organization and the public, sign in to a **Portal**.


## Main menu

To sign in from the main menu, do the following:

1. Start CityEngine.
2. Click **File > Sign in**.
3. Enter your username and password.  
The username is case sensitive; enter it exactly the same as when you created the account.
4. Click **Sign in**.  
Check **Sign me in automatically** to have CityEngine remember your credentials and sign in to the **Portal** automatically; this option is checked by default.

## Sign in menu


To sign in and manage your portals, do the following:

1. Click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click **Sign in** to open the **ArcGIS Sign In for CityEngine** dialog box.
3. Enter your username and password.  
The username is case sensitive; enter it exactly the same as when you created the account.
4. Click **Sign in**.  
Check **Sign me in automatically** to have CityEngine remember your credentials and sign in to the **Portal** automatically; this option is checked by default.

## Manage portals

You can work with content from many portals. To access another portal's content, you must first add a connection to the portal.

To add a portal connection, do the following:

1. Click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click **Manage Portals**.  
The **Portals** dialog box under **Preferences** appears.

3. Click **Add new portal** to open the **Add Portal** dialog box.
4. Enter the URL of the portal.
5. Click **OK**.
6. Enter your username and password.
7. Click **Sign In**.

You can add multiple portals following the same steps above.


## Set the active portal


You can sign in to multiple portals but only one is the active portal to which you connect to ArcGIS Online or ArcGIS Enterprise. To make a portal active, on the **Preferences** dialog box, do the following:

1. Right-click the portal and select **Set as active portal** to make the portal active.
2. Click **OK**.

After you add the portal or portals, remembers your connected portals the next time you start the application.

## Switch the active portal

You can also switch the active portal from the **Sign in** CityEngine menu  **Not signed in** on the toolbar. To switch the active portal, do the following:


1. Click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click **Switch active portal**.  
This opens a drop-down list of all available portals.
3. Click the portal you want to make active.

### **Note:**


You may need to sign in to the portal you selected.

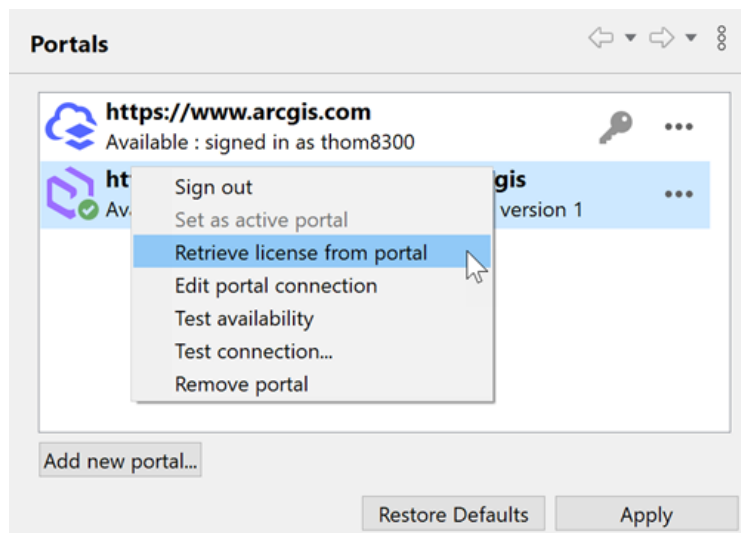
## Change your licensing portal

You may need to change the licensing portal. This is the portal that validates your Named User License (NUL) and allows you to use CityEngine. For example, you need to do this if your CityEngine license is moved from an ArcGIS Online organization to an ArcGIS Enterprise organization.

In the **Portals** dialog box under **Preferences**, the key icon  indicates whether <https://www.arcgis.com> or another portal that you've previously added is the active licensing portal.

To change the licensing portal, ensure the [portal is added](#) and complete the following steps:

1. Click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click **Manage Portals**.  
The **Portals** dialog box under **Preferences** appears.
3. Right-click the portal and click **Retrieve license from portal**.




The **ArcGIS Sign In** dialog box appears.

4. Enter your username and password.
5. Click **Sign In**.

## Sign in with a social login

If your ArcGIS Online account is configured to login using a social login (GitHub, Facebook, Google, or Apple), you can also do this from CityEngine, however, the login is done in an external web browser.


1. To open the **ArcGIS Sign In for CityEngine** dialog box, click **File > Sign In** in the main menu or click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click the GitHub, Facebook, Google, or Apple button in the sign-in dialog box to open an external web browser.
3. Provide your GitHub, Facebook, Google, or Apple credentials in the external web browser.  
If the login in the external browser was successful, you can close the browser tab and return to CityEngine.

### **Note:**

When using a social login method from CityEngine, the authentication code is sent unencrypted from the external browser to CityEngine through the loopback network interface (127.0.0.1 or ::1). However, the authentication code will not leave your computer and additional means to protect the authorization code (PKCE) are implemented.

## Sign in using a web browser

If your account is configured for some special sign-in type, such as, IWA, SAML, OpenID Connect, or PKI, signing in from the CityEngine sign-in dialog may not work. If this happens, you can try to sign in using an external web browser.

1. To open the **ArcGIS Sign In for CityEngine** dialog box, click **File > Sign In** in the main menu or click the **Sign in** menu  **Not signed in** on the toolbar.
2. Click **Sign in using browser** at the bottom of the sign-in dialog to open an external web browser.
3. Provide your credentials in the external web browser.

If the login in the external browser was successful you can close the browser tab and return to CityEngine.



**Note:**

To do this on an ArcGIS Enterprise Portal, you first need to [configure the CityEngine redirect URIs on this portal](#). When using this sign-in method from CityEngine, the authentication code is sent unencrypted from the system browser to CityEngine through the loopback network interface (127.0.0.1 or ::1). However, the authentication code will not leave your computer and additional means to protect the authorization code (PKCE) are implemented.

# Configure the CityEngine redirect URIs on an ArcGIS Enterprise portal

Portal administrators can create or update the registration information needed to sign in on their portal when using a browser.

## For ArcGIS Enterprise 11.1 and later

1. Open the Portal Administrator Directory in a web browser (<https://<domain>/<webadaptorname>/portaladmin>).
2. Click **Login** and sign in as the portal administrator user.
3. Click **Security** > **OAuth** > **Update App Info**.
4. In the text box, type the following lines (include the braces):

```
{
  "appId":"cityenginedesktop",
  "redirectURIs":[
    "urn:ietf:wg:oauth:2.0:oob",
    "http://127.0.0.1:50212/auth",
    "http://127.0.0.1:51883/auth",
    "http://127.0.0.1:52677/auth",
    "http://127.0.0.1:53664/auth",
    "http://127.0.0.1:54177/auth",
    "http://127.0.0.1:56272/auth",
    "http://127.0.0.1:57736/auth",
    "http://127.0.0.1:59916/auth",
    "http://127.0.0.1:62333/auth",
    "http://127.0.0.1:64615/auth",
    "http://[::1]:50212/auth",
    "http://[::1]:51883/auth",
    "http://[::1]:52677/auth",
    "http://[::1]:53664/auth",
    "http://[::1]:54177/auth",
    "http://[::1]:56272/auth",
    "http://[::1]:57736/auth",
    "http://[::1]:59916/auth",
    "http://[::1]:62333/auth",
    "http://[::1]:64615/auth"
  ]
}
```

5. If you are using ArcGIS Enterprise 11.4 and later, these versions support NUL sign in using a browser. To sign in, you need to add CityEngine as the appId with the same redirects in to the text box:

```
{
  "appId":"cityenginedesktop",
  [...],
  "appId":"cityengine",
  "redirectURIs":[
    "urn:ietf:wg:oauth:2.0:oob",
    "http://127.0.0.1:50212/auth",
    "http://127.0.0.1:51883/auth",
    "http://127.0.0.1:52677/auth",
    "http://127.0.0.1:53664/auth",
    "http://127.0.0.1:54177/auth",
    "http://127.0.0.1:56272/auth",

```

```

"http://127.0.0.1:57736/auth",
"http://127.0.0.1:59916/auth",
"http://127.0.0.1:62333/auth",
"http://127.0.0.1:64615/auth",
"http://[::1]:50212/auth",
"http://[::1]:51883/auth",
"http://[::1]:52677/auth",
"http://[::1]:53664/auth",
"http://[::1]:54177/auth",
"http://[::1]:56272/auth",
"http://[::1]:57736/auth",
"http://[::1]:59916/auth",
"http://[::1]:62333/auth",
"http://[::1]:64615/auth"
]
}

```

6. Click **Update App**.

## For ArcGIS Enterprise 11.0 and earlier

1. Sign in to the portal as the portal administrator user.
2. Click **Content**.
3. Click **New item**.
4. Click **Application**.
5. Choose **Other application** and click **Next**.
6. Enter CityEngine into the title and click **Save**.
7. On the item page, click **Settings** next to the **Overview** tab.
8. Click **Registered Info** at the bottom.
9. Copy the string just below **App ID**.
10. Open a new browser tab and enter the Portal Administrator Directory URL  
(<https://<domain>/<webadaptorname>/portaladmin>).
11. Click **Login** and sign in as the portal administrator user.
12. Click **Security** > **OAuth** > **Update App Info** or **Change App Info**.
13. Enter the string copied in step 9 as the Current App ID and type cityenginedesktop for the **New App ID**.
14. Click **Change App ID**.
15. Click **Update App Info**.
16. In the text box, type the following lines (include the braces):

```

{
  "appId": "cityenginedesktop",
  "redirectURIs": [
    "urn:ietf:wg:oauth:2.0:oob",
    "http://127.0.0.1:50212/auth",
    "http://127.0.0.1:51883/auth",
    "http://127.0.0.1:52677/auth",

```

```

    "http://127.0.0.1:53664/auth",
    "http://127.0.0.1:54177/auth",
    "http://127.0.0.1:56272/auth",
    "http://127.0.0.1:57736/auth",
    "http://127.0.0.1:59916/auth",
    "http://127.0.0.1:62333/auth",
    "http://127.0.0.1:64615/auth",
    "http://[::1]:50212/auth",
    "http://[::1]:51883/auth",
    "http://[::1]:52677/auth",
    "http://[::1]:53664/auth",
    "http://[::1]:54177/auth",
    "http://[::1]:56272/auth",
    "http://[::1]:57736/auth",
    "http://[::1]:59916/auth",
    "http://[::1]:62333/auth",
    "http://[::1]:64615/auth"
  ]
}

```

17. Click **Update App**.

 **Note:**

When using this sign-in method from CityEngine, the authentication code is sent unencrypted from the system browser to CityEngine through the loopback network interface (127.0.0.1 or ::1). However, the authentication code will not leave your computer, and additional means to protect the authorization code (PKCE) are implemented.

# Get map data (ArcGIS Online/ArcGIS Enterprise)

[Get map data](#) from ArcGIS Online and ArcGIS Enterprise to add data to a new or existing CityEngine scene.

# Sync feature layers


You can add, update, and manage polygon feature layers hosted by ArcGIS Online. CityEngine allows you to do the following:

- Create a polygon feature layer
- Modify an existing feature layer

## Create and publish a polygon feature layer

You can create a polygon feature layer in ArcGIS Online or ArcGIS Enterprise directly in CityEngine.

1. Create a shape layer with the [shape creation](#) tools.
2. Right-click the shape layer in **Scene Editor** and select **Create Feature Layer**.  
The **Create new Feature Layer** dialog box appears.
3. Fill in the necessary fields.
4. Click **OK**.

The feature layer (hosted) icon  now displays, indicating that a new polygon feature layer has been published to ArcGIS Online.

### **Note:**

- Once the polygon feature layer is published to ArcGIS Online, the layer schema cannot be changed, such as adding or deleting new attribute fields.
- Creating a feature layer is not possible with an ArcGIS Enterprise version earlier than 6.1. **Create Feature Layer** is disabled when the active portal is an ArcGIS Enterprise version earlier than 6.1.


## Modify a hosted polygon feature layer

You can make edits to an existing hosted polygon feature layer by adding the polygon layer to CityEngine, making changes, and synchronizing the layer with the server.

### Add a polygon layer

To add a hosted polygon feature layer, do the following:

1. Open **Navigators**.
2. Right-click the polygon feature layer you want to edit.
3. Click **Import**.  
The **Import feature layer** dialog box appears.
4. Select which layers you want to import from the feature layer.
5. If you want to import only polygons that are inside a specific area, choose **Use spatial envelope**. Select the area by resizing and moving the rectangle on the map or entering the coordinates.
  - The extent of all the imported polygon features must be less than 500 kilometers.
  - If a spatial envelope is used, only polygons inside the selected area will be synchronized.
6. Click **Finish**.

CityEngine will add the selected polygon feature layers to the **Scene Editor** and indicate that the layers are ready for synchronization with the feature layer (hosted) icon . If you don't see the polygon layers in the **Scene Editor** after import, see the **Log** for further details.

## Edit a polygon layer

You can edit the features of a polygon layer inside CityEngine with any of the CityEngine tools, such as the **Move**, **Scale**, or **Regular** shape creation tools.

Make any of the following edits to polygon geometry in the layer:

- Move polygons and vertices along the x-, y-, and z-axes.
- Rotate.
- Reshape.
- Add and delete.

To edit the polygon attributes, do the following:

1. Select a polygon feature.
2. Open the **Inspector**.
3. Click **Object Attributes**.
4. Make edits to the attributes in the layer.

## Sync a polygon layer

Sync the local edits you made in CityEngine to the hosted polygon layer.

1. Open the **Scene Editor**.
2. Right-click the polygon layer.
3. Click **Synchronize**.

The polygon layer is updated on the server. When you sync or create a polygon feature layer, CityEngine automatically validates the geometry using [validation rules and operations for shapes](#), such as storing points counterclockwise or ensuring normals are facing up.

### **Note:**

- CityEngine can only synchronize Short and Long Integers, Double, Float, and Text type fields. Any attribute fields added or deleted will not update when you synchronize the polygon layer.
- With generated models based on an hosted polygon feature layer, you can still edit and synchronize the polygon layer with the generated models visible. Turn off the models in CityEngine to help perform these edits.

## Apply external updates

You can make external edits to a polygon layer, such as in ArcGIS Online or ArcGIS Pro, and apply the updates in CityEngine.

### Make external updates

You can make external edits to both the geometry and attributes, such as in ArcGIS Pro.

- Move polygons and vertices along the x-, y-, and z-axes.
- Rotate.
- Reshape.
- Add and delete.

In ArcGIS Online, you can make edits only to the attribute table of the polygon layer.

- Add fields to the attribute table.
- Edit existing fields in the attribute table.

#### **Note:**

You can only add fields outside of CityEngine, such as in ArcGIS Pro or ArcGIS Online. If you do create new attribute fields outside of CityEngine, you need to remove and re-add the polygon layer in CityEngine for the changes to take affect.

### Sync external edits

1. Open the **Scene Editor**.
2. Right-click the polygon layer.
3. Choose **Synchronize**.

The polygon layer is updated in CityEngine and matches the external updates.

### Local and server conflicts

There may be instances in which there are conflicting edits to the polygon layer in both CityEngine and on the server. In this case, CityEngine displays a **Synchronization Conflict** message that gives you the option to choose whether you want to keep the **Local** CityEngine or **Server** version. The version you select becomes the primary version in CityEngine and on the server.

### Managing polygon layer synchronization

For the synchronization feature to work, you need to ensure that the **Edit** and **Sync** settings are properly configured in ArcGIS Online or ArcGIS Enterprise.

1. Open the polygon feature layer item in ArcGIS Online or ArcGIS Enterprise.
2. Click **Settings**.
3. Under **Feature Layer (hosted) Settings**, check **Enable Editing** and **Enable Sync**.
4. Click **Save**.

#### **Note:**

You can also configure the Edit and Sync settings when you publish the polygon feature layer from ArcGIS Pro.

# Share data

In CityEngine, you can share maps and geographic information. CityEngine guides you through the process of preparing, packaging, and sharing the information online.

## Share file

To share a file, do the following:

1. Click **File** > **Share as** in the main menu.
2. Browse to the file.
3. Click **Open**.

The package dialog box guides you through the sharing process.



### Tip:

You can also right-click the file in the **Navigators** and select **Share as**.

You can share to ArcGIS Online or your organization the following types of files:

- [Esri scene layer package \(slpk\)](#)
- [Rule packages \(rpk\)](#)
- [Tile packages \(tpk\)](#)
- [CityEngine 360 VR Experiences \(3vr\)](#)

## Share data on a different portal

By default, CityEngine targets the main ArcGIS Online portal at [www.arcgis.com](http://www.arcgis.com). If you belong to an organization on ArcGIS Online, or if you want to share your data to ArcGIS Enterprise, define a different portal URL in [CityEngine network preferences](#).

# ArcGIS Urban


# ArcGIS Urban integration

Using ArcGIS Urban in ArcGIS CityEngine allows you to do the following:

- Import ArcGIS Urban plans (spaces, existing buildings, zoning and overlay boundaries, and [scenarios](#)) in ArcGIS CityEngine.
- Edit the imported parcel layers and save the changes to ArcGIS Urban.
- Publish CityEngine models as scene layers to ArcGIS Online and link them to a scenario in an ArcGIS Urban plan.

## Import a plan

To import a plan from ArcGIS Urban to CityEngine, complete the following steps:

1. Set the [active portal](#).
  2. Sign into the portal.
  3. Open the [Navigator](#) window and click the ArcGIS Urban tab .
- A list of available ArcGIS Urban models containing plans and projects displays.

4. Right-click a plan and select **Import as new Scene**.
  - A CityEngine project is created.
  - All scenarios, with spaces, parcels, existing buildings, zoning and overlay layers are imported into a scene.
  - A terrain layer with the basemap and satellite imagery map is imported.



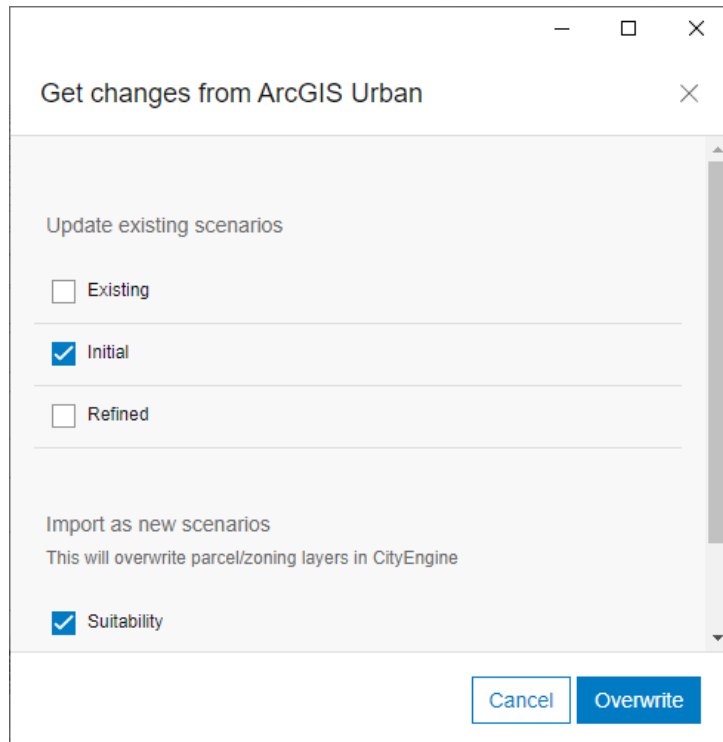
### Note:

- The scene is named after the imported plan.
- The scene is stored in a CityEngine project named after the ArcGIS Urban model configured in the ArcGIS Urban settings. If the CityEngine project doesn't exist, it is created.

## Get changes from ArcGIS Urban

After making changes to zoning regulations and parcel developments in ArcGIS Urban, you can overwrite selected scenarios in the imported scene with those updates. Additionally, you can import new scenarios from ArcGIS Urban to the imported scene.

1. Click **ArcGIS Urban > Get changes from Urban** in the main menu to open the **Get changes from ArcGIS Urban** dialog box:



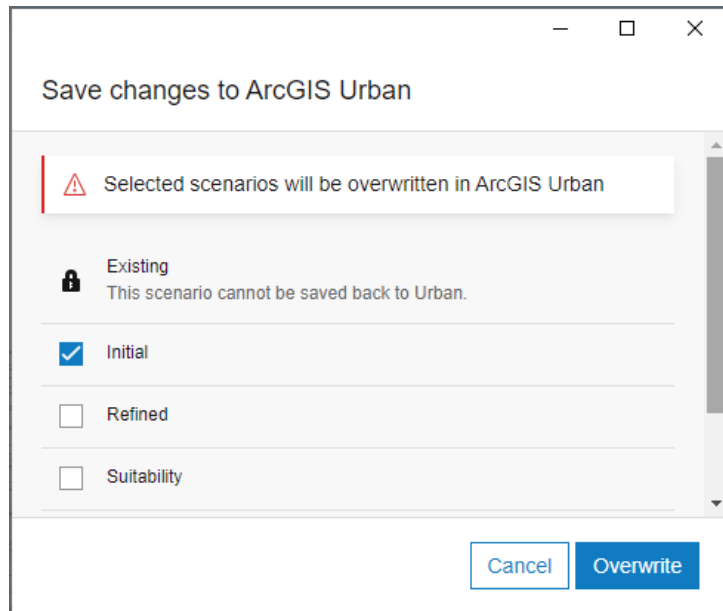
2. Click the scenario to overwrite.
3. Click **Overwrite**.

## Save changes to ArcGIS Urban

After you make changes to the parcel shapes and their parcel attributes in the imported scene, you can save those changes to ArcGIS Urban.

To save the changes to ArcGIS Urban, complete the following steps:

1. Click **ArcGIS Urban > Save changes to Urban** in the main menu to open the **Save changes to ArcGIS Urban** dialog box:



Save changes to ArcGIS Urban dialog box

2. Click the scenario to overwrite.
3. Click **Overwrite**.
4. In the ArcGIS Urban web application, refresh your browser to reload the the parcel changes.
5. Toggle to the **Zoning** menu, scroll down, and click **Reapply all building types** to recreate the spaces.

#### **Note:**

Changes to spaces are not saved to ArcGIS Urban and can only be directly edited in the ArcGIS Urban web application. See [Develop a parcel](#) for more information.

## Publish ArcGIS Urban scene layers

You can publish CityEngine models as scene layers and link them as external layers to an ArcGIS Urban plan scenario as design context. The scene layers appear in ArcGIS Urban when you switch scenarios in the **Projects** details view.

1. In the **Scene Editor** window, activate the scenario for which you want to publish models.
2. Select the models you want to publish.
  - You can publish generated models and static models.
  - Shapes are not exported.
3. Click **ArcGIS Urban > Publish selected models to Scene Layer** in the main menu.  
The models are published to a scene layer and can be linked to the scenario.

**Note:**

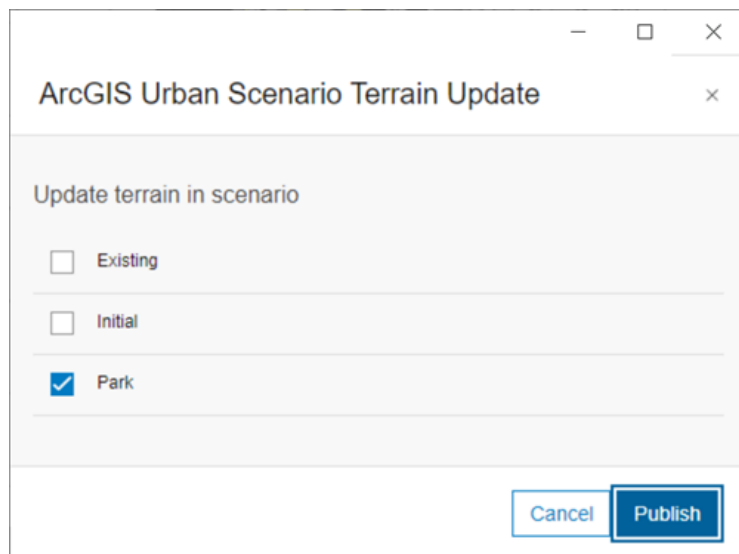
- If the scene layer is linked to the scenario, you can choose to replace it.
- When publishing is complete in CityEngine, the scene layer may not be immediately visible in ArcGIS Urban because the publishing is still running on ArcGIS Online.
- To see how the scene layer looks in ArcGIS Urban, click **ArcGIS Urban > Open in web browser** in the main menu. This opens ArcGIS Urban in your browser and selects the imported plan or project. You must sign in to ArcGIS Urban.

## Publish terrain to a plan scenario

You can select a terrain layer in the **Scene Editor** window and publish it to an ArcGIS Urban plan scenario.

1. In the **Scene Editor** window, click the terrain layer you want to publish.
2. Click **ArcGIS Urban > Publish selected Terrain to Scenario** in the main menu.

The **ArcGIS Urban Scenario Terrain Update** dialog box appears:



3. Select the scenarios to which you want to add the modified terrain.
  4. Click **Publish**.
- The terrain is published to a tile layer service and gets linked to the scenarios.

**Note:**

- When publishing is complete in CityEngine, the tile layer may not be immediately visible in ArcGIS Urban because the publishing is still running on ArcGIS Online.
- To see how the terrain layer looks in ArcGIS Urban, click **ArcGIS Urban > Open in web browser** in the main menu. This opens ArcGIS Urban in your browser and selects the imported plan or project. You must sign in to ArcGIS Urban.

## Existing buildings

The default layers of the imported ArcGIS Urban plan contain buildings outside of its study area as design context. The default Existing scenario contains an Existing Buildings layer showing the existing conditions. All other

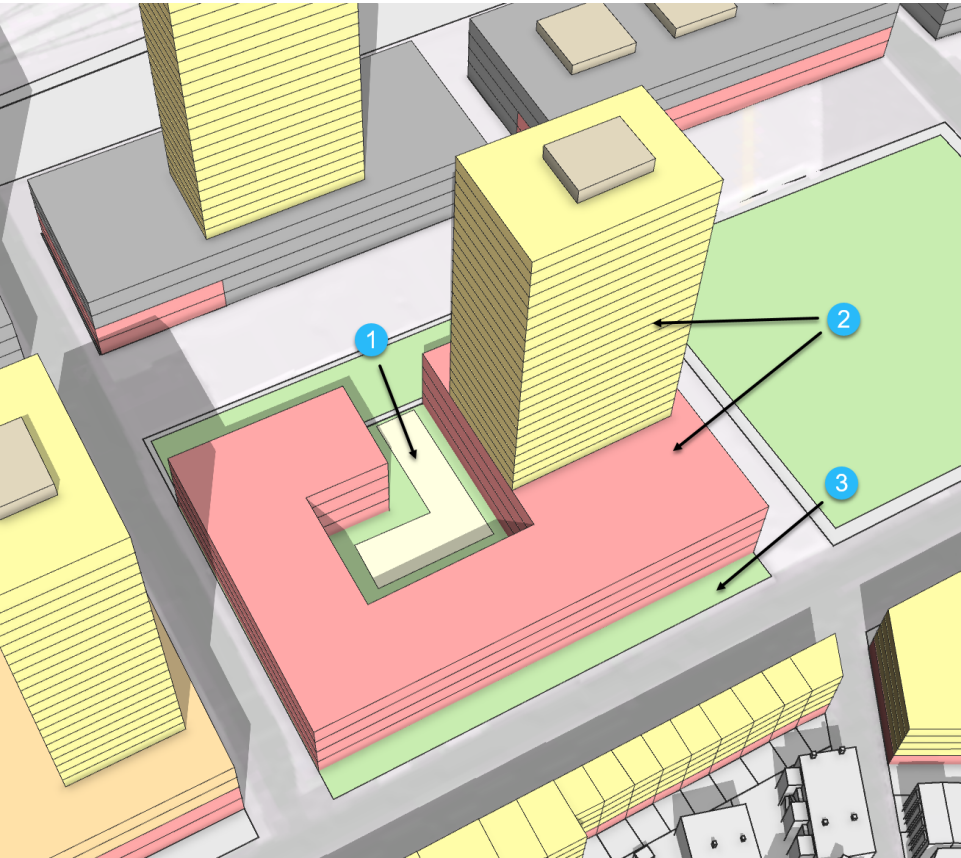
scenarios contain a copy of the **Existing Buildings** layer, in which the buildings are removed on demolished or redeveloped parcels, and inside demolished areas defined in ArcGIS Urban.

 **Note:**

Existing buildings can be imported into CityEngine if the **Existing Building** scene Layer has an accompanying feature layer. When a feature layer is not available, CityEngine suggests to download Open Street Map building footprints.

## Spaces

The scenarios have a **Spaces** group layer with shape layers containing the spaces. By default, a rule from the **ESRI.lib** project gets assigned that visualizes the spaces colored by space-use type. The following image shows the types of spaces:



1	Surface space
2	Building spaces
3	Ground space

### Building spaces

Building spaces are represented as floor volumes extruded from a floor plate shape. A shape layer with the **Building** label contains all floor plates of a building.

## Surface spaces

Surface spaces are represented as planar shapes aligned to the parcel. A shape layer with the **Surfaces** label contains all the surfaces of a parcel.

## Ground spaces

Parcels can have a space-use type assigned as default parcel ground use. In this case, a shape derived from the **Ground Space** gets added to the parcel's **Surfaces** layer.



### Note:

- To select all spaces of a building, right-click a building space and select **Objects in Same Layer**.
- See [Develop a parcel](#) in ArcGIS Urban for more information on spaces.

## Building envelopes

The building envelopes of new developments are visualized in the **Zoning Envelopes** layer. The layer resides in the **Zoning** group layer. Use the visibility settings of the layer to hide the layer. To update the envelope with any changes in the zoning regulations, click **ArcGIS Urban > Get changes from ArcGIS Urban** in the main menu.

## Generate future buildings

In CityEngine, you can generate future buildings of developed parcels similar to that in ArcGIS Urban. You need to assign to your parcels the latest ArcGIS Urban rule located in the `/ESRI.lib/rules/urban` project folder.

## Parcels Inspector

When you select a parcel, the **Inspector** window displays the **Parcels Attributes** section instead of **Object Attributes**. At the top, there is a dedicated **Zoning** section that indicates the zoning designation as defined in ArcGIS Urban and allows you to define overrides for effective regulations.

### Override zoning attributes

You can override zoning attributes for the selected parcel by entering the new value in the text box. An overridden zoning attribute is indicated by the attribute value shown in the bold font. To reset an overridden attribute, click the drop-down menu and select **Remove override**.

### Change building types

To change the building type for the selected parcel, go to the **Parcels Attributes** section and click the **Building Type** drop-down menu and select the new type from the list. Ensure **Development Type** is set to **Set Building Type**.



### Note:

You cannot edit the list of building types or the parameters of a building type in CityEngine. To change building types, edit them in ArcGIS Urban and synchronize them with your CityEngine scene. This updates the list of building types in the CityEngine scene.

# 360 VR Experience

# Export 360 VR Experiences from CityEngine

A 360 VR Experience (.3vr) is a file format to publish and consume panoramic images of CityEngine scenes and web scenes. These panoramic images are consumable in browsers (users look around with mouse), mobiles (users look around with gyro/touch), and virtual reality (VR) headsets (users look around by turning their head).

At its core, the CityEngine 360 VR Experience exporter is an automated way to take a series of viewport snapshots based on camera bookmarks. These snapshots are combined into panoramic images (one for each bookmark). The exported 360 VR Experience can then be consumed either locally or through the cloud after uploading.

You can also do the following with 360 VR Experiences:

- [View 360 VR Experiences](#) on desktop, mobile devices, and VR.
- [Create and publish 360 VR Experiences](#) from web scenes directly in a web browser.

## 360 VR Experience export

The exporter converts CityEngine [scenarios](#) or layer compositions into 360 VR Experience scenarios; turning CityEngine [camera bookmarks](#) in to 360 VR Experience viewpoints. It then writes everything into a single 360 VR Experience (.3vr) file.


The screenshot shows the 'Export 360 VR Experience' dialog box. The title bar says 'Export 360 VR Experience'. The dialog contains the following fields and controls:

- Output Path:** A text field with the path 'C:\Users\matt7894\Documents\CityEngine\Default Workspace\Example' and a 'Browse...' button.
- Experience Name:** A text field with the value 'Philadelphia'.
- Alternatives:** A dropdown menu showing 'Layer Compositions'.
- Render Settings:** A dropdown menu showing 'Default Settings'.
- Bookmarked Views:** A section with four checkboxes: 'North Facing Facades', 'West Facing Facades', 'South Facing Facades', and 'Park'. All are checked.
- Layer Compositions:** A section with a list box containing '> [x] Layer Composition 1'.
- Layer Composition Operations:** A row of buttons: 'Add', 'Duplicate', 'Remove', 'Rename', and 'Reset'.
- Navigation:** At the bottom, a row of buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

To create a 360 VR Experience, do the following:

1. Open the CityEngine scene.
2. Setup bookmarks and give them meaningful names.
3. Optionally, enable the camera **Bookmark gizmos** view option in **View settings** > **Bookmark gizmos** to visualize the position of your camera viewpoints.

- 4. Open the 360 VR Experience export wizard by clicking **File > Export > CityEngine > Export 360 VR Experience**.
- 5. Adjust the export settings.

<b>Output Path</b>	Specify where to save the 360 VR Experience (.3vr) file. If you plan to upload it to <a href="#">ArcGIS Online</a> , a subdirectory of the current CityEngine project is recommended.
<b>Experience Name</b>	The title of the panoramic image.
<b>Alternatives</b>	<ul style="list-style-type: none"><li>• <b>None</b>—Offer no alternatives. The exporter will use the current layer and object visibility and take a panoramic image for each bookmark.</li><li>• <b>Scenarios</b>—Use the scene <a href="#">scenarios</a> as 360 VR Experience scenarios.</li><li>• <b>Layer Compositions</b>—Specify the composition of scene layers—which scene layers to include in an 360 VR Experience scenario. The default state is initialized with the layer visibility. Use the Layer Composition Operations for modification.</li></ul>
<b>Render Settings</b>	Choose the render settings to use when exporting the panoramas. <ul style="list-style-type: none"><li>• <b>Default Settings</b>—Choose the default render settings.</li><li>• <b>{Viewport}</b>—Choose a viewport to export its render settings defined in the View settings .</li></ul>
<b>Bookmarked Views</b>	Choose which camera bookmarks to include. This will decide the number of unique points-of-view available in the panoramic image.

- 6. Click **Finish**.  
The 360 VR Experience can now be [viewed locally](#) or uploaded to the cloud, such as ArcGIS Enterprise or ArcGIS Online.


## Share 360 VR Experience to ArcGIS Online or ArcGIS Enterprise

 **Note:**

Although you can share 360 VR Experience items to ArcGIS Online and ArcGIS Enterprise, they can currently only be viewed in ArcGIS Online. See [View 360 VR Experiences stored in ArcGIS Enterprise](#) for workarounds to this limitation.

To share a 360 VR Experience, do the following:

- 1. Export the 360 VR Experience to a folder in your current CityEngine project, such as the \model\ folder.
- 2. [Sign in](#) to ArcGIS Enterprise or ArcGIS Online.
- 3. In the **Navigator** window, right-click the 360 VR Experience (.3vr) file and choose **Share as...**
- 4. Choose option **Upload package to my ArcGIS Online or Portal account** and type a name for the experience.
- 5. Fill in the required fields for **Item Description**.
- 6. Choose the access permission of the item in the **Sharing** tab.

7. Click **Analyze** to validate your 360 VR Experience for any errors or issues.  
You must validate and resolve all errors before you can save it to disk or share it to ArcGIS Online or ArcGIS Enterprise. If any issues are discovered, an error will be reported. You have to fix the error before you can continue.
8. Click **Share** to upload your web scene package to ArcGIS Online or ArcGIS Enterprise.
9. Click **My Content**  in **Navigator** to see your uploaded 360 VR Experience.

## Preview 360 VR Experiences

To preview a 360 VR Experience locally in your browser, do the following:

1. Export the 360 VR Experience to a folder in your current CityEngine project , such as the \model1\ folder.
2. In **Navigator**, right-click the 360 VR Experience (.3vr) file and click **Open in browser**.  
The 360 VR Experience opens in your default browser.

# Preferences and Shortcuts

# Appearance preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

To change the appearance preferences such as color themes and font set, click **Preferences > Appearance**.

## Choose appearance theme

Under **CityEngine Theme**, specify the active theme for the UI.

- **Light**—The default appearance.
- **Dark**—The UI such as toolbars, menus, and windows are dark. This presents a more modern look and can help reduce strain on the eyes.

Under **Visible tabs on overflow**, check the **Show most recently used tabs** check box to display the last tabs used.

## Change font set

Under **Appearance > Fonts**, you can change the font set.

# General preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The following preferences can be changed on the **General** preferences page:

- **Always run in background**—Turn this option on to perform long-running operations in the background that don't prevent you from doing other work.
- **Keep next/previous editor, view and perspectives dialog open**—If this option is turned on, the editor and view cycle dialog boxes will remain open when their activation key is released. Normally, the dialog box closes as soon as the key combination is released.
- **Show heap status**—Turn this option on to display an indicator showing information about current Java heap usage. A basic heap monitor in CityEngine reports Java and core heap usage.
- **Open mode**—You can select one of the following methods for opening resources:
  - **Double click**—Clicking a resource will select it and double-clicking it will open it in an editor.
  - **Single click (Select on hover)**—Hovering over the resource will select it and clicking it once will open it in an editor.
  - **Single click (Open when using arrow keys)**—Selecting a resource with the arrow keys will open it in an editor.



## Note:

Depending on which view has focus, selecting and opening a resource may have different behavior.

# Editors preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Editors** preferences under **General > Editors**.

You can change the following preferences on the **Editors** preferences page:

- **Size of recently opened files list**—Each file that is opened in an editor is stored in a list of recently used files in the File menu. This option controls the number of files that is displayed in that list.
- **Show multiple editor tabs**—Specify whether to show multiple editor tabs. If this is off, editor workbooks have one large tab and all non-visible editors are accessible only from the drop-down menu.
- **Show Heap Status**—Turn this option on to display an indicator showing information about current Java heap usage. A basic heap monitor in CityEngine reports Java and core heap usage.
- **Allow in-place system editors**—Specify if OLE in-place editing will be used on the Windows platform.
- **Restore editor state on startup**—Specify if editors should be restored on the next launch of CityEngine.
- **Close editors automatically**—Specify whether to reuse editors in CityEngine. If this is on, you may specify the number of editors to use before they are recycled (the default is 8). You can also specify if a prompt dialog box will open or if a new editor will open when all editors are dirty (have unsaved changes). Once it is turned on, the Pin Editor action is added to the toolbar and editor tab menu. Pinned editors are not recycled.

# Keys preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The function of the keyboard can be extensively customized in CityEngine using the **Keys** preferences page. In CityEngine, key strokes and key sequences are assigned to invoke particular commands.

You can set **Keys** preferences under **General > Keys**.

## Key strokes, sequences, and bindings

A key stroke is the pressing of a key on the keyboard while, optionally, holding down **Ctrl**, **Alt**, or **Shift**.. For example, holding down **Ctrl** while pressing **A** produces the key stroke **Ctrl+A**. The pressing of the modifier keys themselves does not constitute a key stroke.

A key sequence is one or more key strokes. Traditionally, Emacs assigned two or three key stroke key sequences to particular commands. For example, the normal key sequence assigned to Close All in Emacs is **Ctrl+X Ctrl+C**. While Eclipse supports key sequences of arbitrary lengths, it is recommended that keyboard shortcuts be four key strokes in length (or less).

A key binding is the assignment of a key sequence to a command.

## Schemes

A scheme is a set of bindings. CityEngine includes two schemes:

- Procedural CityEngine (default)
- Autodesk Maya
- Autodesk Revit
- Google SketchUp
- McNeel Rhino
- Autodesk 3ds Max
- Blender
- Autodesk Autocad
- Graphisoft ArchiCAD
- Maxon Cinema4D
- Emacs (do not use)
- Default (do not use)

The Procedural CityEngine scheme contains a general set of bindings, in many cases recognizable as traditional key sequences for well-known commands. For instance, **Ctrl+A** is assigned to Select All, and **Ctrl+S** is assigned to Save.

Choose the scheme you are most comfortable with by changing the **Scheme** setting on the **Keys** preferences page.

## Contexts

Key bindings can vary based on the current context.

For instance, the active part may be a CGA shape grammar editor in which a different set of key sequence assignments may be more appropriate than if the active part was a 3D viewport. As a specific example, typically **X**, **Y**, and **Z** are assigned to normal typing actions in a context such as CGA shape grammar editing, while **X**, **Y**, or **Z** is assigned to axis alignment in a 3D viewport. This context is usually determined by the active window, but it can be influenced by the active dialog box as well. If the active window does not choose a particular context, the active context is set to In Windows.

CityEngine includes a number of contexts, for example:

- In Dialogs and Windows
- In Windows (extends In Dialogs and Windows)
- In Dialogs (extends In Dialogs and Windows)
- Editing Text (extends In Windows)
- In Viewport
- In Console

#### **Note:**

It is not recommended to promote a key binding to a context that it extends. For example, it is not recommended that you move an Editing Text key binding to the In Dialogs and Windows context. This may have unexpected results.

Some key bindings work in dialog boxes. Those key bindings are assigned to the In Dialogs and Windows context. One example of such a key binding is the key binding for cut. You can change these key bindings. For example, you can set **Ctrl+X** to cut in dialog boxes, but **Ctrl+W** to cut in windows.

## Platform and locale

Key bindings also vary by platform and locale. In Chinese locales (zh), **Alt+Slash** is assigned to Content Assist, instead of the usual **Ctrl+Spacebar**.

The current platform and locale is determined when CityEngine starts and does not vary over the lifetime of a running CityEngine.

## Customizing key bindings

With multi-stroke key sequences, schemes, and contexts, there are a lot of things to keep in mind when customizing key bindings. All key customization is done on the **Keys** preferences page.

For example, you want to bind **Ctrl+5** to the About command. By default, the **Keys** preference page shows you all possible key bindings. You can see the About command listed in the Help category. You can bind the command by putting focus in the Binding text box and pressing **Ctrl** and **5** as if you were running the command.

When you type **Ctrl+5**, you have created a binding for About. The column on the right will indicate that this is a user binding by displaying a U. If there was a conflict with another key, this column would also display a C. The binding will be in the default context, In Windows. You can now use the **When** combination box to change the key binding context (for example, to move this binding to Editing Text).

To add a second key binding to About, you can use the **Copy Command** button to create a second command entry for you to bind another key to. To delete a binding, you can either use the **Remove Binding** button or give focus to the **Binding** text box and press **Backspace**.

## Conflict resolution

There are only a finite number of common key strokes available to assign to a multitude of commands. Scheme, context, platform, and locale all partition key sequence assignments into domains where they don't conflict with one another.

If the user sets a key binding and creates a conflict, the conflicting bindings will be displayed in the conflicts list. This can be used to switch between conflicting key bindings so that they can be changed.

Resolve these conflicts by assigning the key sequence to one of the commands or remove it from the other.

# Miscellaneous preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Miscellaneous** preferences under **General > Miscellaneous**.

The following preferences can be changed on the **Miscellaneous** preferences page:

- **Exit CityEngine after [minutes]**—CityEngine features an automatic exit functionality. CityEngine can automatically exit after a given number of minutes of inactivity. Before exiting, all open files will be saved. Type 0 minutes to disable automatic exit. Automatic exit can be useful for installations in which you usually do not close applications and thus do not return licenses held by these applications to the license server. Closing the application returns the licenses held to the license server and allows other users to start the application.
- **Enable immediate mode**—In immediate mode, procedural models are generated automatically if the underlying initial shape or its attributes change.

## 3D mouse preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The **3D Mouse** preferences page allows you to configure CityEngine-related settings of your Logitech/3DConnexion 3D mouse. The top of the preferences page shows CityEngine user interface commands. You can assign any of these commands to the hardware buttons on your 3D mouse. To assign a command to a button, select the command from the list and press the button on the device to assign the command to. The list of commands may be filtered by typing the desired filter into the text field above the list of commands.

You can set **3D Mouse** preferences under **Navigation Devices > 3D Mouse**.

Note, that on Windows platforms, the settings made in the Logitech/3DConnexion system control panel are bypassed. To configure your 3D mouse for CityEngine, use the built-in 3D mouse preferences.

3D mouse support is not available in all CityEngine versions.

You can restore the default command bindings for your specific device by clicking the **Restore Defaults** button.

### Settings

You can change the following settings:

- **Overall Speed**—The overall speed (or sensitivity) of the 3D mouse. Change this if you think your device responds too quick or slow.
- **Reverse all Axes**—You may prefer to have the movements in helicopter or camera mode aligned with the mouse. You may reverse the axes with this option.
- **Zoom direction**—By default, zoom is mapped to moving the cap forward and backward. You may prefer zoom on the up and down axis. You may select your preferred behavior with this option.
- **Navigation mode**—The preferred navigation mode (see below).
- **Rotate**—Enable rotation. If this option is not checked, the device will have no effect when you rotate the cap.
- **Pan/Zoom**—Enable pan/zoom. If this option is not checked, the device will have no effect when you move the cap.

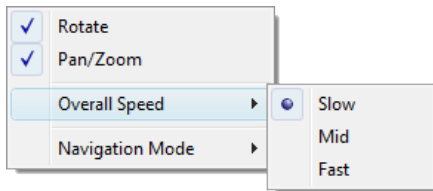
### Navigation modes

The following are navigation modes:

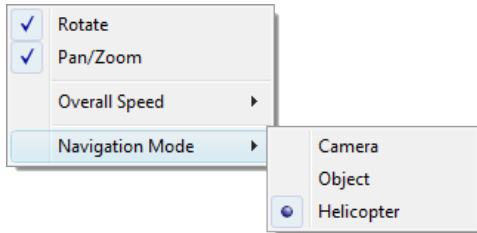
- Camera mode navigation allows you to visually move around in the scene you are observing. This turns in the direction that the cap on the 3D mouse moves and causes the objects displayed to move in the opposite direction. In camera mode, the center of rotation is at the camera position.
- The object mode navigation allows you to virtually hold an object in your hand. The center of rotation is the current point of interest, for example, set to the center of the selection by framing.
- The helicopter mode simulates a helicopter control mechanism. The device's pan axes control the movement in a plane parallel to the world's xz-plane irrespective of the applied tilt. The device's y-axis is used directly to control the height above the world's xz-plane. In this navigation mode, pulling the devices cap up causes the height above world's xz-plane to increase, increasing the distance of the view point above the plane. Similarly, pressing the cap down causes the view point to get closer to the plane. Not only are the device's y-translation values applied directly to the world's up-axis, the same is true for the devices spin values. These rotations act as if the device's and the world's up-axis were coincidental.

## 3D Mouse menu

The following images show **3D Mouse** menu settings.



*Overall speed settings are shown.*



*The Navigation mode settings are shown.*

By default, button 1 on the 3D mouse is assigned to the **3D Mouse** menu. This menu allows you to lock translation or rotational movements as well as change the speed and navigation mode quickly.

# Mouse preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The **Mouse** preferences allow you to select specific mouse schemes. CityEngine defines mouse schemes for major 3D applications. The schemes are read-only, unless you select the Custom scheme, which allows you to define the mouse actions for 3D navigation and editing.

You can set **Mouse** preferences under **Navigation Devices > Mouse**.



## Note:

On Linux platforms, it is recommended that you use the CityEngine Linux scheme. This scheme uses **Ctrl** as the default modifier key and does not interfere with window manager operations that commonly use the **Alt** modifier.

# Touch preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The **Touch** preferences allow you to fine-tune the touch navigation in CityEngine.

You can set **Touch** preferences under **Navigation Devices > Touch**.

# Procedural Runtime preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The **Procedural Runtime** preferences page controls options with respect to rule derivation (model generation), display, rendering, occlusion, and miscellaneous engine arguments.

You can set **Procedural Runtime** preferences under **Navigation Devices > Procedural Runtime**.

## CGA Compiler

If **Write compiler output to console window** is enabled, the CGA compiler's activities are logged to a console.

## Generate

The **Generate** section contains the following settings:

- The maximum array size limits the number of elements in arrays. This avoids an accidental high memory consumption.
- The maximum derivation depth controls the maximum recursion level of rules (createShape), or the depth of the shape tree (model hierarchy), respectively.
- The maximum derivation width controls the breadth of the shape tree (model hierarchy).
- The maximum function call depth controls the maximum recursion level of function calls. This includes attributes.
- The disk cache size controls how much disk space is used to cache decoded textures between starts of CityEngine. Using the disk cache may reduce memory consumption during runtime because textures do not need to be decoded (and kept in memory) to find their metadata (such as aspect ratio).
- Use **Number of parallel generate threads** to set the number of threads to be used to execute one generate command.
- The extent of the trim planes can be controlled with trim plane size. Note that this is for computation only; the rendering size of the trim planes can be controlled in the display settings below.

## Occlusion and Context

The **Occlusion and Context** section contains the following settings:

- **Disable inter queries** and **Disable intra queries**—Disabling intra-shape tree or inter-shape tree (neighbors) queries can be useful for rule debugging.
- **Neighborhood distance for inter-occlusion queries**—All shapes within this distance of the bounding box of a shape are considered neighbors (occluders). This means their models must be derived for inter-occlusion queries. This property is stored per scene.
- **Neighborhood distance for inter-context queries**—All shapes within this distance of the bounding box of a shape are considered for labeled context queries. This means their models must be derived for inter-context queries. If set to 0, the distance is ignored and all scene shapes are considered. This property is stored per scene.
- **Maximum distance for occlusion**—Due to floating point limitations, occlusion queries use this threshold value.

## Display Options

The **Display Options** section contains the following settings:

- **Edges size**—Defines the display size (thickness) of edges.

- **Vertices size**—Defines the display size (diameter) of vertices.
- **Pivot size**—Defines the display size of pivots.
- **Pivot line with**—Defines the display line width of pivots.
- **Scope line with**—Defines the display line width of scopes.
- **Trim plane size**—Defines the display size of trim planes.

## Rendering

The **Rendering** section affects only generated models and contains the following settings:

- **Disable GL Mipmaps**—Some hardware has problems using mipmaps and texture compression together. If this is the case on your system, you can disable GL mipmaps.
- **Disable GL Texture Compression**—By default, textures are compressed for rendering. This significantly reduces the memory consumption (typically ratios between 1:6 and 1:4 are achieved, depending on the texture format) and speed up rendering. However, the texture quality is slightly reduced.
- **Max texture width/height**—Textures that are wider/higher than this value are rescaled to this value to save memory.
- **Matching profile**—Using this pop-up, you can control render performance versus memory consumption. For most use cases, **Balanced** is a good choice.
- **Force OpenGL Double Buffering**—On Windows, double buffering is unavailable because the operating system already takes care of smooth rendering. Use this option to force double buffering.

## License

Timeout for license server connection in milliseconds.

## Logging

Set the **Procedural runtime log level** value. If set to a level less than 6, Procedural Runtime logs errors, warnings and so on to a console.

# Viewport preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Viewport** preferences under **General > Viewport**.

The following are the Viewport preferences :

You can change the appearance and behavior of 3D viewports.

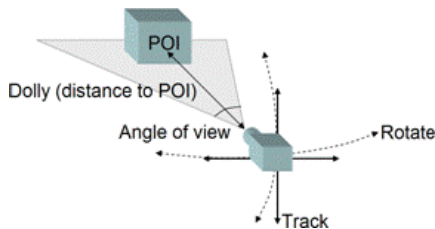
Color options	Set various colors for scene elements.
<b>Animation time [ms]</b>	Set time for camera animations (for example, when triggering a camera bookmark).
<b>Disable camera during camera navigation</b>	Speed up camera navigation on heavier scenes.
<b>Line width</b>	Set width for line rendering.

# Bookmarks preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Bookmarks** preferences under **General > Viewport > Bookmarks**.

A bookmark has the following options:



The Bookmark schematic is shown.

- **Perspective**—If selected, the camera will give a perspective correct image. If not selected, the camera will operate in orthogonal mode where parallel edges will also appear as parallel lines in the 3D viewport. You can switch between perspective and orthogonal mode by pressing **P** or selecting **Orthogonal View** from the **Focal Length** menu.
- **Angle of view**—The width of the field of view. Some predefined angles of view corresponding to specific focal lengths are accessible from the **Focal Length** menu.
- **Rotate/Translate**—These values position and orient the camera. You can type a world coordinate position for the camera as well as a specific rotation of the camera.
- **Distance to Point of Interest**—This value corresponds to the distance between the camera and the point of interest (POI). The POI is also the center of rotation, and thus the distance to the POI defines the radius of the camera rotation.

# Cameras preferences

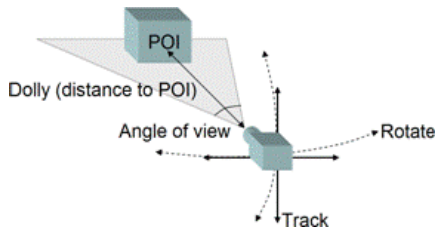
You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Cameras** preferences under **General > Viewport > Cameras**.

The following are the **Cameras** preferences:

## Configure cameras

Camera configurations have the following options:



*The Camera schematic is shown.*

- **Perspective**—If selected, the camera gives a perspective correct image. If not selected, the camera operates in orthogonal mode where parallel edges also appear as parallel lines in the 3D viewport. You can switch between perspective and orthogonal mode by pressing **P** or selecting **Orthogonal View** from the **Focal Length** menu.
- **Angle of view**—The width of the field of view. Additional predefined angles of view corresponding to specific focal lengths are accessible from the **Focal Length** menu.
- **Rotate/Translate**—These values position and orient the camera. You can enter a world coordinate position for the camera as well as a specific rotation of the camera.
- **Distance to Point of Interest (POI)**—This value corresponds to the distance between the camera and the POI. The POI is also the center of rotation and thus the distance to the POI defines the radius of the camera rotation.

# Light preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set **Bookmarks** preferences under **General > Viewport > Light**.

The **Light** preferences control how objects are lit in the **Viewport**:

<b>Sun position source</b>	Switches between Time and Month and Direct Solar Entry. For Time and Month to work, the <a href="#">Scene Coordinate System</a> must be set correctly.
<b>Time</b>	Time of sun position.
<b>Time Zone</b>	Time zone of sun position.
<b>Month</b>	Month of sun position.
<b>Solar elevation angle</b>	Altitude of the sun, the angle between the horizon and the center of the sun's disc.
<b>Solar azimuth angle</b>	Azimuth angle of the sun.
<b>Shadow quality</b>	Switches between Low, Medium, High, and Interactive.
<b>Solar intensity</b>	Scene light intensity.
<b>Ambient intensity</b>	Scene light ambient intensity.
<b>Shadow attenuation</b>	Attenuation of shadows (blend to black).
<b>Ambient occlusion attenuation</b>	Attenuation of screen space ambient occlusion (blend to black).
<b>Radius mode</b>	Switches between Manual and Interactive.
<b>Ambient occlusion radius</b>	Radius in meters of screen space ambient occlusion samples.
<b>Ambient occlusion samples</b>	Number of screen space ambient occlusion samples, or Interactive.

# Help preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

The following are **Help** preferences you can set for accessing help information:

On the **Help** preferences page, you can indicate how to display help information.

- Use external browsers—If an embedded web browser is supported on your system, the help window uses an embedded help browser to display help contents, whenever possible, and this option is available. Select it to force help to use external browsers. Use the **Web Browser** preferences page to select a browser to use.
- Open window context help—This option allows you to determine whether the window context help will be opened in a dynamic help view or in a pop-up.
- Open dialog context help—This option allows you to determine whether the dialog context help will be opened in a dynamic help section of help view or in a pop-up.
- Open help view documents—This option allows you to determine whether the documents selected in the help view will be opened in-place or in the editor area.

# Network preferences

You can set the CityEngine preferences by clicking **Edit** > **Preferences** in the main menu.

Change **Network** settings for your portal URL and proxy.

## ArcGIS Online or portal

By default, CityEngine connects to ArcGIS Online at [www.arcgis.com](http://www.arcgis.com). Set a URL to change the target portal for sign in and for sharing CityEngine web scenes and rule packages.

- For anPortal for ArcGIS on-premise installation—`https://webadaptor.domain.com/arcgis`
- For an ArcGIS Online organization: `http://jsapi.maps.arcgis.com`

## HTTP proxy

Define proxy settings for CityEngine network connections.

# Scene preferences

You can set the CityEngine preferences by clicking **Edit > Preferences** in the main menu.

You can set the following **Scene** preferences.

## Scene coordinate system

Set or redefine the scene coordinate system. See [Georeferencing](#).

Modifying the scene coordinate system will not reproject or relocate the data in the scene. It only redefines the internally stored reference system used for future data imports.

## Show handles

This preference enables or disables handles in this scene. Handles are shown on the currently selected model, if the rule file was created with handles. See [Handles](#).

# Shortcuts

Depending on your operating system, some of the default CityEngine shortcuts may be globally assigned to other actions. In this case, you can either change the operating system's defaults (where available) or change the CityEngine key bindings (see Keyboard preferences).

You can recall a list of active CityEngine shortcuts by pressing **Ctrl+Shift+L**.

Command	Binding	Category	When
Activate editor	<b>F12</b>	In windows	Window
Add comment block	<b>Ctrl+4</b>	Python editor scope	Python - Editor
Add memory block	<b>Ctrl+Alt+M</b>	In Memory view	Run/Debug
Add single comment block	<b>Ctrl+Shift+4</b>	Python editor scope	Python - Editor
Backward history	<b>Alt+Left</b>	In Windows	Navigate
Bookmark 0	<b>Numpad_0</b>	In 3D viewport	Viewport Control
Bookmark 1	<b>Numpad_1</b>	In 3D viewport	Viewport Control
Bookmark 2	<b>Numpad_2</b>	In 3D viewport	Viewport Control
Bookmark 3	<b>Numpad_3</b>	In 3D viewport	Viewport Control
Bookmark 4	<b>Numpad_4</b>	In 3D viewport	Viewport Control
Bookmark 5	<b>Numpad_5</b>	In 3D viewport	Viewport Control
Bookmark 6	<b>Numpad_6</b>	In 3D viewport	Viewport Control
Bookmark 7	<b>Numpad_7</b>	In 3D viewport	Viewport Control
Bookmark	<b>8 Numpad_8</b>	In 3D viewport	Viewport Control
Bookmark 9	<b>Numpad_9</b>	In 3D viewport	Viewport Control
Bounding boxes	<b>B</b>	In 3D viewport	Viewport Control
Cancel	<b>Esc</b>	In dialog boxes and windows	Engine Control
Close	<b>Ctrl+W</b>	In windows	File
Close	<b>Ctrl+F4</b>	In windows	File
Close All	<b>Ctrl+Shift+F4</b>	In windows	File
Close All	<b>Ctrl+Shift+W</b>	In windows	File
Close rendering	<b>Ctrl+W</b>	In Memory view	Run/Debug
Collapse	<b>Ctrl+Numpad_Subtract</b>	Editing text	Text Editing
Collapse	<b>Minus Sign</b>	In 3D viewport	Graph Commands

Command	Binding	Category	When
Collapse all	Ctrl+Shift+Numpad_Divide	In windows	Navigate
Collapse all	Ctrl+Shift+Numpad_Divide	Editing text	Text Editing
Collapse others	/	In 3D viewport	Graph Commands
Content assist	Ctrl+Spacebar	In dialog boxes and windows	Edit
Context information	Ctrl+Shift+Spacebar	In dialog boxes and windows	Edit
Copy	Ctrl+C	In dialog boxes and windows	Edit
Copy	Ctrl+Insert	In dialog boxes and windows	Edit
Copy lines	Ctrl+Alt+Down	Editing text	Text Editing
Cut	Shift+Delete	In dialog boxes and windows	Edit
Cut	Ctrl+X	In dialog boxes and windows	Edit
Debug last launched	F11	In windows	Run/Debug
Delete	Delete	In windows	Edit
Delete line	Ctrl+D	Editing text	Text Editing
Delete next word	Ctrl+Delete	Editing text	Text Editing
Delete previous word	Ctrl+Backspace	Editing text	Text Editing
Delete to end of line	Ctrl+Shift+Delete	Editing text	Text Editing
Deselect all	Ctrl+Shift+A	In 3D viewport	Selection Commands
Disable snapping	Shift	In 3D viewport	Edit
Duplicate lines	Ctrl+Alt+Up	Editing text	Text Editing
EOF	Ctrl+Z	In I/O console	Run/Debug
Expand	Plus Sign	In 3D viewport	Graph Commands
Expand	Ctrl+Numpad_Add	Editing text	Text Editing
Expand all	Ctrl+Shift+Numpad_Multiply	In windows	Navigate
Expand all	Ctrl+Numpad_Multiply	Editing text	Text Editing
Expand others	*	In 3D viewport	Graph Commands
Export	Ctrl+E	In windows	File
Extract local	Alt+Shift+L	Python editor scope	Python - Refactor

Command	Binding	Category	When
Extract local	Alt+Shift+T, L	Python editor scope	Python - Refactor
Extract method	Alt+Shift+T, E	Python editor scope	pepticcategor
Extract method	Alt+Shift+M	Python editor scope	pepticcategor
Find and replace	Ctrl+F	In windows	Edit
Find next	Ctrl+K	Editing text	Edit
Find previous	Ctrl+Shift+K	Editing text	Edit
Find references	Ctrl+Shift+G	In CGA Editor	Search
Find text in workspace	Ctrl+Alt+G	In windows	Search
First char	Home	Python editor scope	Python - Editor
Forward history	Alt+Right	In windows	Navigate
Frame all	A	In 3D viewport	Viewport Control
Frame selection	F	In 3D viewport	Viewport Control
Front or back view	Z	In 3D viewport	Viewport Control
Generate constructor using fields	Alt+Shift+T, C	Python editor scope	pepticcategor
Generate models	Ctrl+G	In dialog boxes and windows	Shape Commands
Generate properties	Alt+Shift+T, P	Python editor scope	pepticcategor
Go to address	Ctrl+G	In table memory rendering	Run/Debug
Go to line	Ctrl+L	Editing text	Navigate
Help contents	F1	In windows	Help
Home	H	In 3D viewport	Viewport Control
Incremental find	Ctrl+J	Editing text	Edit
Incremental find reverse	Ctrl+Shift+J	Editing text	Edit
Inline local	Alt+Shift+T, I	Python editor scope	Python - Refactor
Inline local	Alt+Shift+I	Python editor scope	Python - Refactor
Insert line above current line	Ctrl+Shift+Enter	Editing text	Text Editing
Insert line below current line	Shift+Enter	Editing text	Text Editing

Command	Binding	Category	When
Join lines	Ctrl+Alt+J	Editing text	Text Editing
Last edit location	Ctrl+Q	In windows	Navigate
Left and right side view	X	In 3D viewport	Viewport Control
Line end	End	Editing text	Text Editing
Line start	Home	Editing text	Text Editing
Lock and use current coordinate system	Period	In 3D viewport	Coordinate System Commands
Maximize active view or editor	Space	In 3D viewport	Window
Maximize active view or editor	Ctrl+M	In windows	Window
Move lines down	Alt+Down	Editing text	Text Editing
Move lines up	Alt+Up	Editing text	Text Editing
New	Ctrl+N	In windows	File
New menu	Alt+Shift+N	In windows	File
New rendering	Ctrl+N	In Memory view	Run/Debug
Next	Ctrl+Period	In windows	Navigate
Next editor	Ctrl+F6	In windows	Window
Next memory monitor	Ctrl+Alt+N	In Memory view	Run/Debug
Next method or class	Ctrl+Shift+Down	Python editor scope	Python - Editor
Next page	Alt+F7	In windows	Navigate
Next page of memory	Ctrl+Shift+Period	In table memory rendering	Run/Debug
Next subtab	Alt+PageDown	In dialog boxes and windows	Navigate
Next view	Ctrl+F7	In windows	Window
Next word	Ctrl+Right	Editing text	Text Editing
Offline action for scripting	Ctrl+2	Python editor scope	Python - Editor
Open	Ctrl+O	In windows	File
Open resource	Ctrl+Shift+R	In windows	Navigate
Open search dialog box	Ctrl+H	In windows	Search
Override or implement methods	Alt+Shift+T, O	Python editor scope	pepticcategory

Command	Binding	Category	When
Paste	Ctrl+V	In dialog boxes and windows	Edit
Paste	Shift+Insert	In dialog boxes and windows	Edit
Previous	Ctrl+Comma	In windows	Navigate
Previous editor	Ctrl+Shift+F6	In windows	Window
Previous method or class	Ctrl+Shift+Up	Python editor scope	Python - Editor
Previous page	Alt+Shift+F7	In windows	Navigate
Previous page of memory	Ctrl+Shift+Comma	In table memory rendering	Run/Debug
Previous subtab	Alt+PageUp	In dialog boxes and windows	Navigate
Previous view	Ctrl+Shift+F7	In windows	Window
Previous word	Ctrl+Left	Editing text	Text Editing
Print	Ctrl+P	In windows	File
Properties	Alt+Enter	In windows	File
Python collapse	Ctrl+Minus Sign	Python editor scope	Python - Editor
Python collapse all	Ctrl+9	Python editor scope	Python - Editor
Python comment	Ctrl+3	Python editor scope	Python - Editor
Python format code	Ctrl+Shift+F	Python editor scope	Python - Editor
Python go to definition	F3	Python editor scope	Python - Editor
Python organize imports	Ctrl+Shift+O	Python editor scope	Python - Editor
Python show class browser	Ctrl+Shift+T	Python editor scope	Python - Editor
Python show outline	Ctrl+O	Python editor scope	Python - Editor
Python toggle comment	Ctrl+Slash	Python editor scope	Python - Editor
Python uncollapse	Ctrl+Equal Sign	Python editor scope	Python - Editor
Python uncollapse all	Ctrl+0	Python editor scope	Python - Editor

Command	Binding	Category	When
Python uncomment	Ctrl+Shift+3	Python editor scope	Python - Editor
Python uncomment	Ctrl+Backslash	Python editor scope	Python - Editor
Quick access	Ctrl+3	In windows	Window
Quick diff toggle	Ctrl+Shift+Q	Editing text	Edit
Quick switch editor	Ctrl+E	In windows	Window
Redo	Ctrl+Y	In dialog boxes and windows	Edit
Refresh	F5	In windows	File
Regenerate all models	Ctrl+F5	In dialog boxes and windows	Shape Commands
Remove comment block	Ctrl+5	Python editor scope	Python - Editor
Rename	F2	In windows	File
Rename.	Alt+Shift+R	Python editor scope	Python - Refactor
Rename	Alt+Shift+T, R	Python editor scope	Python - Refactor
Rename	Alt+Shift+R	In CGA Editor	Refactoring
Reset structure	Ctrl+Shift+Numpad_Multiply	Editing text	Text Editing
Resume	F8	Debugging	Run/Debug
Run last launched	Ctrl+F11	In windows	Run/Debug
Run script	F9	Python editor scope	Python - Run
Run to line	Ctrl+R	Debugging	Run/Debug
Save	Ctrl+S	In windows	File
Save all	Ctrl+Shift+S	In windows	File
Save as bookmark 0	Ctrl+Numpad_0	In 3D viewport	Viewport Control
Save as bookmark 1	Ctrl+Numpad_1	In 3D viewport	Viewport Control
Save as bookmark 2	Ctrl+Numpad_2	In 3D viewport	Viewport Control
Save as bookmark 3	Ctrl+Numpad_3	In 3D viewport	Viewport Control
Save as bookmark 4	Ctrl+Numpad_4	In 3D viewport	Viewport Control
Save as bookmark 5	Ctrl+Numpad_5	In 3D viewport	Viewport Control
Save as bookmark 6	Ctrl+Numpad_6	In 3D viewport	Viewport Control

Command	Binding	Category	When
Save as bookmark 7	Ctrl+Numpad_7	In 3D viewport	Viewport Control
Save as bookmark 8	Ctrl+Numpad_8	In 3D viewport	Viewport Control
Save as bookmark 9	Ctrl+Numpad_9	In 3D viewport	Viewport Control
Select higher level	PageUp	Local edits	Local Edits
Select lower level	PageDown	Local edits	Local Edits
Select next pattern	End	Local edits	Local Edits
Select previous pattern	Home	Local edits	Local Edits
Script 0	Ctrl+Alt+Numpad_0	In windows	Scripting
Script 1	Ctrl+Alt+Numpad_1	In windows	Scripting
Script 2	Ctrl+Alt+Numpad_2	In windows	Scripting
Script 3	Ctrl+Alt+Numpad_3	In windows	Scripting
Script 4	Ctrl+Alt+Numpad_4	In windows	Scripting
Script 5	Ctrl+Alt+Numpad_5	In windows	Scripting
Script 6	Ctrl+Alt+Numpad_6	In windows	Scripting
Script 7	Ctrl+Alt+Numpad_7	In windows	Scripting
Script 8	Ctrl+Alt+Numpad_8	In windows	Scripting
Script 9	Ctrl+Alt+Numpad_9	In windows	Scripting
Scroll line down	Ctrl+Down	Editing text	Text Editing
Scroll line up	Ctrl+Up	Editing text	Text Editing
Select all	Ctrl+A	In dialog boxes and windows	Edit
Select line end	Shift+End	Editing text	Text Editing
Select line start	Shift+Home	Editing text	Text Editing
Select next word	Ctrl+Shift+Right	Editing text	Text Editing
Select previous word	Ctrl+Shift+Left	Editing text	Text Editing
Set tool (Tool: Create Segment Tool)	G	In 3D viewport	Tool Commands
Set tool (Tool: Circular Creation)	Shift+C	In 3D viewport	Tool Commands
Set tool (Tool: Curve Tool)	C	In 3D viewport	Tool Commands
Set tool (Tool: Local Edits)	O	In 3D viewport	Tool Commands
Set tool (Tool: Measure distance)	M,D	In 3D viewport	Tool Commands

Command	Binding	Category	When
Set tool (Tool: Measure area and path)	M, A	In 3D viewport	Tool Commands
Set tool (Tool: Polygonal Creation)	S	In 3D viewport	Tool Commands
Set tool (Tool: Rectangle Creation)	Shift+S	In 3D viewport	Tool Commands
Set tool (Tool: Selection Tool)	Q	In 3D viewport	Tool Commands
Set tool (Tool: Transform Move Tool)	W	In 3D viewport	Tool Commands
Set tool (Tool: Transform Rotate Tool)	R	In 3D viewport	Tool Commands
Set tool (Tool: Transform Scale Tool)	E	In 3D viewport	Tool Commands
Shaded render mode	5	In 3D viewport	Tool Commands
Shadows	8	In 3D viewport	Tool Commands
Show contributing plug-in	Alt+Shift+F3	In dialog boxes and windows	Window
Show definition	F3	In CGA Editor	Search
Show in	Alt+Shift+W	In windows	Navigate
Show inspector	Alt+I	In windows	Views commands
Show key assist	Ctrl+Shift+L	In dialog boxes and windows	Window
Show ruler context menu	Ctrl+F10	Editing text	Window
Show ToolTip description	F2	Editing text	Text Editing
Show view	Alt+Shift+Q, Q	In windows	Views
Show view (View: Breakpoints)	Alt+Shift+Q, B	In windows	Views
Show view (View: Console)	Alt+Shift+Q, C	In windows	Views
Show view (View: Error Log)	Alt+Shift+Q, L	In windows	Views
Show view (View: History)	Alt+Shift+Q, Z	In windows	Views
Show view (View: Outline)	Alt+Shift+Q, O	In windows	Views
Show view (View: Problems)	Alt+Shift+Q, X	In windows	Views
Show view (View: Search)	Alt+Shift+Q, S	In windows	Views
Show view (View: Synchronize)	Alt+Shift+Q, Y	In windows	Views
Show view (View: Variables)	Alt+Shift+Q, V	In windows	Views

Command	Binding	Category	When
Show view Menu	Ctrl+F10	In dialog boxes and windows	Window
Show or hide graph networks	F10	In 3D viewport	Uncategorized
Show or hide map layers	F9	In 3D viewport	Uncategorized
Show or hide models	F12	In 3D viewport	Uncategorized
Show or hide shapes	F11	In 3D viewport	Uncategorized
SSAO	9	In 3D viewport	Viewport Control
Step into	F5	Debugging	Run/Debug
Step over	F6	Debugging	Run/Debug
Step return	F7	Debugging	Run/Debug
Switch to editor	Ctrl+Shift+E	In windows	Window
Terminate	Ctrl+F2	Debugging	Run/Debug
Text end	Ctrl+End	Editing text	Text Editing
Text start	Ctrl+Home	Editing text	Text Editing
Textured render mode	6	In 3D viewport	Viewport Control
To lowercase	Ctrl+Shift+Y	Editing text	Text Editing
To uppercase	Ctrl+Shift+X	Editing text	Text Editing
Toggle axes	D, A	In 3D viewport	Viewport Control
Toggle block selection	Alt+Shift+A	Editing text	Edit
Toggle breakpoint	Ctrl+Shift+B	In windows	Run/Debug
Toggle compass	D, C	In 3D viewport	Viewport Control
Toggle current coordinate System	Comma	In 3D viewport	Coordinate System Commands
Toggle folding	Ctrl+Numpad_Divide	Editing text	Text Editing
Toggle grid	D, G	In 3D viewport	Viewport Control
Toggle information display	D, D	In 3D viewport	Viewport Control
Toggle information display	D, V	In 3D viewport	Viewport Control
Toggle insert mode	Ctrl+Shift+	Insert editing	Text Edit
Toggle isolate selection	I	In 3D viewport	Viewport Control
Toggle memory monitors pane	Ctrl+T	In Memory view	Run/Debug
Toggle overwrite	Insert	Editing text	Text Editing
Toggle perspective	P	In 3D viewport	Viewport Control

Command	Binding	Category	When
Toggle scene light	L	In 3D viewport	Viewport Control
Toggle wireframe on shaded or textured	7	In 3D viewport	Viewport Control
Top or bottom view	Y	In 3D viewport	Viewport Control
Undefined command	Ctrl+B	In windows	Unavailable Category
Undefined command	Ctrl+F8	In windows	Unavailable Category
Undefined command	Alt+Minus Sign	In windows	Unavailable Category
Undefined command	Ctrl+1	In dialog boxes and windows	Unavailable Category
Undefined command	Ctrl+Shift+F8	In windows	Unavailable Category
Undo	Ctrl+Z	In dialog boxes and windows	Edit
Update seed	Ctrl+Shift+G	In dialog boxes and windows	Shape Commands
Use step filters	Shift+F5	In windows	Run/Debug
Wireframe render mode	4	In 3D viewport	Viewport Control
Word completion	Alt+Slash	Editing	Text Edit
Zoom or dolly in	Ctrl+Equal Sign	In 3D viewport	Navigation Commands
Zoom or dolly in	Ctrl+Numpad_Add	In 3D viewport	Navigation Commands
Zoom or dolly out	Ctrl+Minus Sign	In 3D viewport	Navigation Commands
Zoom or dolly out	Ctrl+Numpad_Subtract	In 3D viewport	Navigation Commands