ArcGIS CityEngine 2022.0

Released: June 2022

Online CityEngine Help

Table of Contents

ArcGIS Urban

ArcGIS 360 VR

Preferences and Shortcuts

CityEngine help

The CityEngine 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.

Additional online resources

You can find additional CityEngine resources available online that are not included in this PDF document. If you are new to CityEngine, check out About CityEngine or the Essential skills tutorial. Ensure you are online to view these topics.

You can gain further knowledge and insight into CityEngine by visiting the online pages below:

- 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 main menu.

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

Note:
There are links throughout this PDF document to other pages that are available offline. Additionally, there are links to online CityEngine resources, such as those mentioned above.

UI Basics

UI basics overview

The CityEngine user interface (UI) has several windows that can be repositioned and resized to suit your unique needs. Below is a typical layout in CityEngine:

CityEngine UI overview

1	Scene Editor—Scene, layer, object and scenario management.
2	CGA Editor—Editing CGA rules.
3	Navigator—Manage and preview files in the workspace.
4	Viewport—Single and multiple perspective cameras and scenarios.
5	Inspector—Detailed view and editing of selected objects and scenarios.
6	Tool Options—Manage tool options when using tools.
7	Dashboards—Overview of main attributes by scenario.

Additional windows are:

- Log—CityEngine messages
- Console—CGA output
- Problems—CGA compiler errors and warnings
- Progress—Progress reporting of long-running CityEngine operation
- Model Hierarchy—Inspect Models generated with CGA

Navigator

Use the **Navigator** to work with your local workspace, projects, and files as well as browse ArcGIS Online or on-premise 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:

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

Search and filtering files

You can search for files in the active space by entering text in the search box and/or filter for a specific file types 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** 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 double click them.

Preview 3D models

The 3D preview can be tumbled and zoomed using the same shortcuts as for the **Viewport**. 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 on 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 **Navigator** or from the main menu.

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

1. Locate the file in **Navigator**.
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**.
- CGA rules open in the **CGA Editor**.
- Python modules open in the python editor.
- Other files open in the system editor.

Drag and drop

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

- **CGA rule files and packages**—Select a set of shapes and/or models in the 3D viewport, drag a CGA file and drop it 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** and choose what layers to import in the popup 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 appear in a new layer.
- **File Geodatabase**—A dialog opens that lets you decide what parts of the file geodatabase you want to import.
- **Images**—A dialog opens that lets you decide how to import the image.

Note:
Drag and drop works for single elements only.

Scene Editor

The **Scene Editor** is the central place where you manage your 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, 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** 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 may rearrange layers by dragging them to the desired position.

Create a group layer

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

To create a new **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 and drop them in the new group layer.

Layer states

To work with layers effectively, you can toggle their visibility and edit status and assign different colors to make editing in the viewport easier. The states are represented by the squares next to the **Scene Editor**.

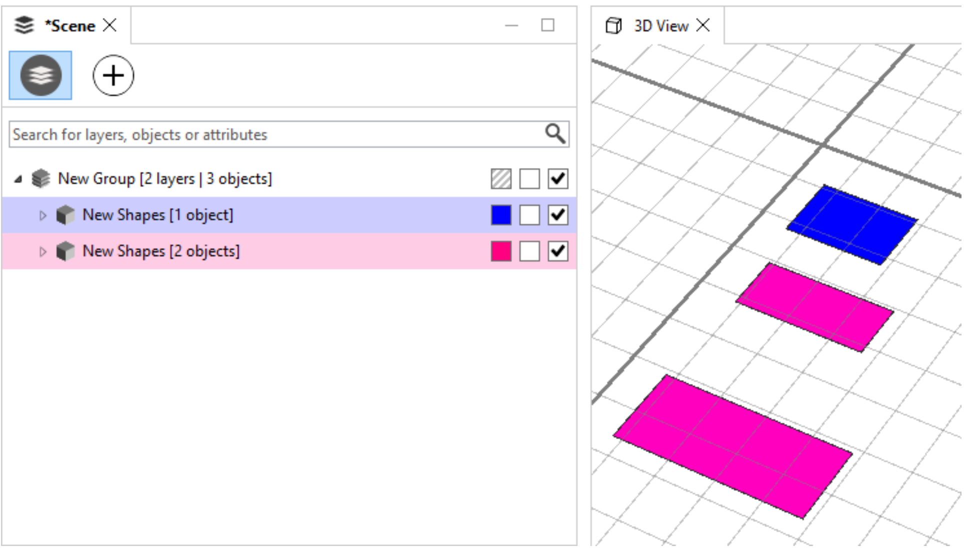
Set the layer color

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** and in the **Inspector**. When a layer doesn't have a color assigned to it, the color box displays grey-striped diagonal lines.

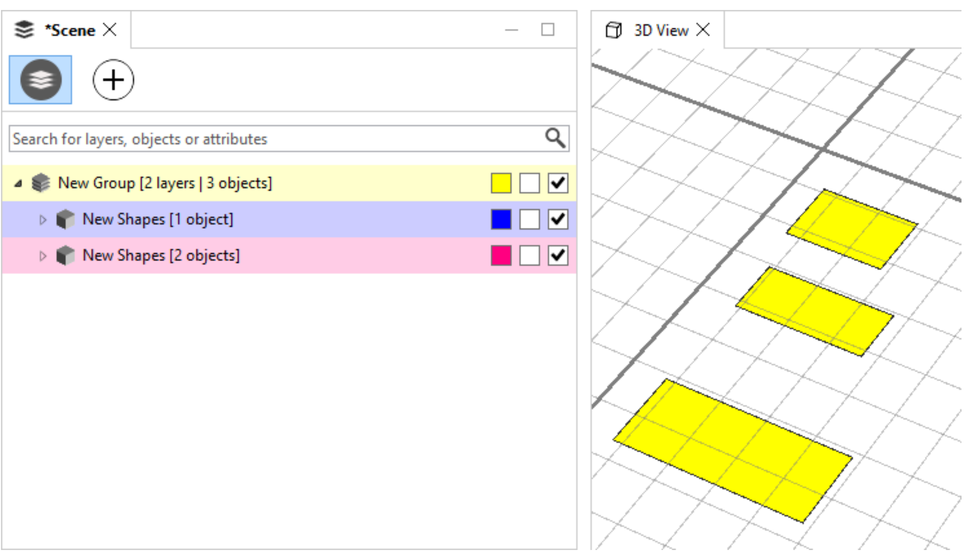
Layer colors and hierarchies

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**. In the **Scene Editor**, 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

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



Two colored layers with inherited colors

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 and in the **Inspector**.

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

Toggle visibility

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 any viewports. Invisible objects can't be selected. You can access the visibility in the **Scene Editor** and in the **Inspector**.

Note:
- When a group is "invisible", 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 / 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 on its name twice, using with the context menu ("Rename" entry), or by changing the name field in the **Inspector**.
Framing elements	Double click any element to frame the element in all viewports.
Multi selection	Select multiple layers/groups and to change visibility/lock/color state for all the selected items.
Using the Search Field to select and filter objects	The search field allows the user to search for objects with name match, attribute match (name and value). The result of the search is automatically selected and framed in the **Viewport**. The result of the search action is displayed in the status bar. In the search field on the top of the scene window you can type a wildcard expression (e.g. `"Lot*"`) to select matching objects. • `Shape*` — Matches scene objects whose names start with `Shape` • `"Shape 12"` — Matches scene objects whose names are exactly `Shape 12` • `"Shape* *12"` — Matches scene objects whose names start with `Shape` AND end with `12` **Note:** Whitespace characters in a search query denote a logical OR expression. To match object names with spaces, put your search query in double quotes ""

Importing and exporting layers	• In the case where 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 ".obj" file (each group corresponds one shape) or a ".dxf" file or similar. Afterwards copy it into the data folder of your project and import it by clicking **File** > **Import** > **CityEngine Layers** from the main menu and choose your file type or through the **Navigator** context menu on the file. Afterwards you can apply CGA shape grammar rules in the usual way on these imported shapes. • Any object from the scene window can be exported to a new scene by choosing **File** > **Export** > **Export Selected Objects as .cej** > To import layers from the exported ".cej" file into a new scene, select **File** > **Import** > **CityEngine Layers** from the main menu.

Viewport

The 3D **Viewport** is your main interaction tool 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 new viewport.

To learn more on how to navigate in 3D, transform scene objects, and work with bookmarks, see scenes.

Viewing modes and display settings

The 3D **Viewport** 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**.

Layer type visibility

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

Item	Function	Shortcut
Isolate Selection	Isolate the current selection, hide unselected objects. **Note:** Terrains and Analyses layers will still be visible even if not selected.	Press `I`
Map Layers	Toggles attribute map layer visibility for this viewport.	Press `F9`
Graph Networks	Toggles graph network visibility for this viewport.	Press `F10`
Shapes	Toggles shape visibility for this viewport.	Press `F11`
Models	Toggles model visibility for this viewport.	Press `F12`
Analyses	Toggles analysis visibility for this viewport.	

Camera and viewport settings

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

Item	Function	Shortcut
10mm fisheye lens (121° FOV)	Type of lens and FOV (field of view).	
18mm ultra-wide lens (90° FOV)	Type of lens and FOV (field of view).	
24mm wide-angle lens (73° FOV)	Type of lens and FOV (field of view).	
35mm standard lens (54° FOV)	Type of lens and FOV (field of view).	
50mm standard lens (39° FOV)	Type of lens and FOV (field of view).	
70mm telephoto lens (28° FOV)	Type of lens and FOV (field of view).	
135mm telephoto lens (15° FOV)	Type of lens and FOV (field of view).	

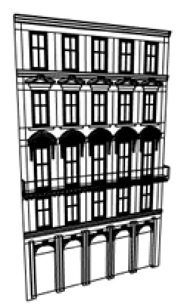
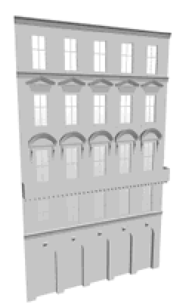
Item	Function	Shortcut
Parallel projection view	Type of lens and FOV (field of view).	Press `P`
Wireframe	Renders the scene contents as the Wireframe of the face borders. **Tip:** To see the wireframe of terrains, select them in the **Scene editor** and enable the **Wireframe** option in the **Inspector**.	Press `4`
Shaded	Renders the scene contents with colors but with no textures. **Note:** This also means that no opacity maps are applied and some otherwise transparent parts are rendered opaque.	Press `5`
Textured	Renders the scene contents with colors and textures.	Press `6`
Wireframe on Shaded/Textured	Enable/disable an overlay like the wireframe render mode but on top of the rendered elements.	Press `7`
Shadows	Turns shadows on or off. **Note:** Note that enabling shadows on large models may considerably affect rendering performance.	Press `8`
Ambient Occlusion	Enable/disable ambient occlusion rendering (ambient occlusion).	Press `9`
On-camera light	Toggles the sun position between the specified location and straight behind the camera.	Press L
Single-Sided Lighting	• Toggles between single- and double-sided lighting. • With single-sided lighting faces are rendered the same no matter whether you look from the front or the back side. • With double-sided lighting faces are rendered as though they were a physical wall which looks different on both sides.	
Backface Culling	Enable/disable backface culling. With backface culling enabled, only faces facing the camera are rendered.	
Terrain Masking	Enable/disable masking of multiple terrains in overlapping terrain regions.	
Information Display	Toggles the information display. The information display provides statistics for the current scene such as number of objects and polygon count.	Press `D,D`

Item	Function	Shortcut
Axes	Toggles rendering of the axis visualization in the bottom left corner. **Note:** The coordinate system can be switched in **View Coordinate System**.	Press `D,A`
Compass	Toggles rendering of the Compass in the bottom right corner.	Press `D,C`
Grid	Toggles rendering of the Grid at elevation 0 for reference.	Press `D,G`
Bookmarks gizmos	Toggles visibility. Toggles rendering of small cameras for the Bookmarks locations.	
Handles	Toggles handles visibility.	
View Coordinate System	Choose a Coordinate System for the Navigation Display and the axis visualization	

Note:
Press `D` to open the shortcut table for display of Information, Axes, Compass, and Grid

Render modes

The **Viewport** 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`.	
Shaded — All polygons are filled with the object color. No texturing takes place. The shaded mode can be activated by selecting the **Shaded** button or pressing `5`	

Textured — All polygons are filled with textures and object colors. The textured mode can be activated by selecting the **Textured** button or pressing `6`.	
Wireframe on Shaded/Textured — Overlay wireframe independent of the viewing mode by enabling **Wireframe on Shaded/Textured** in the **View Settings** menu or pressing `7`.	

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 also Viewport preferences.

Viewport context menu

Depending on the selection, the context menu contains different entries for different objectives.

Frame (F)	Frame the selection (or the whole scene if selection is empty).
Select Objects in Same Layer	All objects in the same layer(s) are selected.
Select Objects with Same Rule File	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 toggle the full screen mode for this **Viewport**.

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 non-unique names again.

Inspector

The **Inspector** is the main tool for viewing and modifying attributes of objects, such as shapes, map layers, and models. Depending on the type of object selected, the **Inspector** adapts its user interface to provide full access to the object's attributes. You can open the **Inspector** by clicking **Window** > **Inspector** in the main-menu, or by pressing Alt+`I`.

The **Inspector** not only supports editing of single objects but also 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 non-unique with the "?" symbol. The **Inspector** shows the attributes of the lead object when multiple objects are selected. The lead object is always the most recent individually selected object.

For map layers, the **Inspector** lets you change the map files, modify the bounds, and adjust the display offset (how much the rendering of the map is displaced regarding the actual map values). In addition, an overlay color and alpha value for the map can be specified. See Map Layers for more information.

Inspector parameters

Depending on the type of the selected objects in the Viewport (Shape, Segment, Lot, Node, Sidewalk, Block) there are different parameter options available:

See Block, Node, Segment and sidewalk parameters for more information.

Inspector Sources

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

Default	The default value used. For rule attributes, this is the initial attribute value. For parameters, this is the algorithm specific default.
User	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.
Object	The value is taken from the corresponding object attribute. The value is displayed in italic, and marked with `(Object)`.
Shape	A rule attribute can use the value from the parent shape. For example a street shape may sample the street segment's `streetWidth`. The value is displayed in italic, and marked with `(Shape)`.
Layer	The value is connected to a layer attribute. The value is displayed in italic, and marked with the source layer in brackets. See Layer Attributes for details.

Note:

- The easiest way to create connections and set sources is using the Connection Editor.
- During rule assignment, object attributes with names that match a rule attribute get connected automatically. During rule assignment, object attributes with names that match a rule attribute get connected automatically.
- Whenever you enter a value to a parameter or attribute, it will automatically change to user source.

Work with object attributes

To use the **Inspector** to add an attribute, do the following:

1. Click **Add new object attribute**.
2. Enter information for the **Attribute Name**, **Value**, and **Type**.
   City Engine has the following types of object attributes:
   - `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 entering the value of an attribute, the user can enter a "NaN" value for a float (Not a Number), or a "NULL" value for a String.
- When entering the value for attributes arrays, separate the values with a `,` (comma). For example :
  - `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]`

Edit with the List Editor

You can use the **List Editor** to edit array attributes in the **Object Attributes** section.

1. Click **Edit List...** from the drop-down menu.
2. Modify the values by editing the second column cells.
   You can replicate the elements, add, duplicate, and delete elements using the toolbar.
3. Click **OK**.
   The new values are added to the data model and scene.

Edit with the Table Editor

If an object has many attribute arrays which follow the `prefix_...` syntax, you can edit the attribute values from the drop-down menu and select **Edit Table...** for each of the attributes.

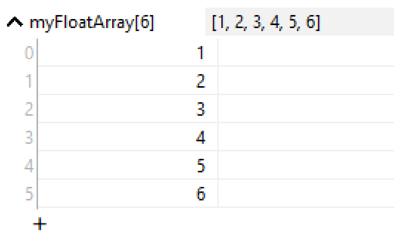
To edit the multiple array attributes, do the following:

1. Click **Edit List...** from 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**.

Work with array attributes

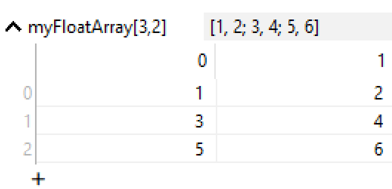
CityEngine supports float, string, and bool arrays. The **Inspector** displays arrays for rule and object attributes and allows for editing.

1. Click the button to expand the array attributes.
   Array indices are displayed in grey.

2. Click the **+** icon to add a new row to the end of the array.
   The added element is set to the default value for each type (i.e. `0` for floats, `""` for strings, `false` for bools).
3. Click on an element to edit it.
4. Right-click on a row to delete or insert rows.

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



2D arrays displayed in table

Map attributes with Connection Editor

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

To edit 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**.
3. Set the attribute connection.
4. Click **OK**.

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

<table>
<tr><td>Object attribute</td><td>Connect your attribute to an object attribute. This option is only available if:
the scene object has an object attribute with a matching name.
the type of the object attribute matches the required type of the attribute.</td></tr>
<tr><td>Shape parameter</td><td>Connect your attribute to the parent shape parameter. This option is only available if:
it is a rule attribute
the rule file is attached to a shape of an intersection, street, or block (a lot)
the attribute has the same name as a shape parameter
the type of the shape parameter matches the required type of the attribute</td></tr>
<tr><td>Layer Attribute</td><td>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.
Each layer will provide a list of its available layer attributes, consisting of the following:
Channel attributes for Map Layers: the color channels (red, blue, green, alpha, brightness, ...) of the layers image. Marked as <code>(Map Channel)</code>.
Object attributes: Object attributes of objects in the source layer. Marked as <code>(Object attribute)</code>.
Other attributes: Existing expressions or mappings. Marked as <code>(Layer attribute)</code> or <code>(Expression)</code>.
Note:
If a layer attribute with the same name as your attribute already exists the new attribute might override the existing one.</td></tr>
<tr><td>Reset attributes</td><td>You can reset a specific attribute through the Rule default option in the attribute drop-down menu. Depending on the attribute's type this resets to the default value of the algorithm (e.g. Street Shape creation) or to the default value from the rule file (rule attribute). Additional options are available as well from the attribute context menu (open by right click) :
Reset user attributes — Reset all user-set attributes of this rule file to its default (Rule) values.
Reset all attributes — Reset all user-set and mapped attributes of this rule file to its default (Rule) values.</td></tr>
</table>

Map object attributes with Layer Attribute

Every layer can have an arbitrary set of layer attributes defined. Whereas, map layers normally use their image data as 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 its scene objects to attributes with different names, or to objects on other layers.

Note:
For simple cases, use 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 object attributes with matching name within the layer that the object attribute was created.

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
```

streetWidth parameter dependent on other attributes

`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**.

Reports

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

Reports

Report	N	%	Sum	%	Avg	Min	Max	NaNs
FAR	77	100.00	14.71	100....	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....	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....	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

Reports table in the Inspector

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 `Retail` or `Office`, 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

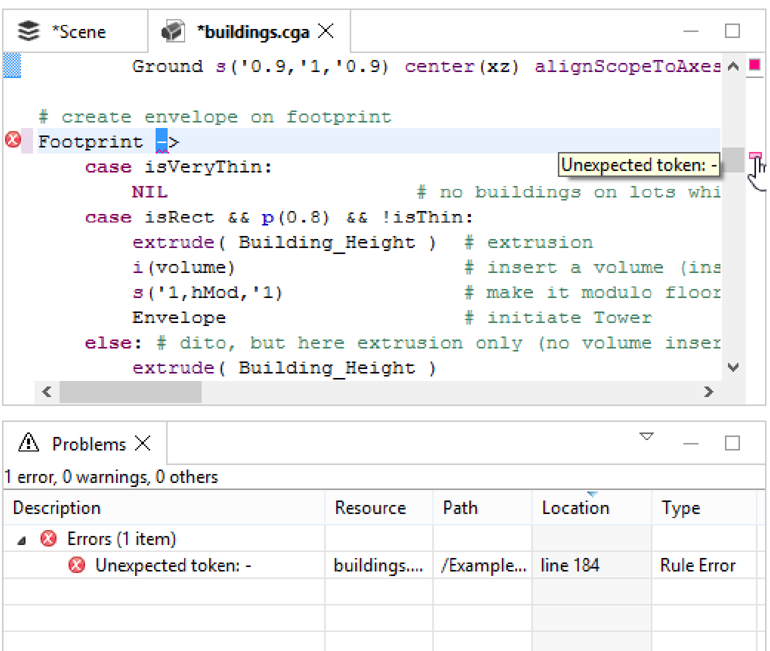
In the **CGA Editor**, you can write, modify, and save CGA rules. To edit a CGA rule, do the following:

1. Open the CGA rule file.
   - Double click the rule file in the **Navigator**.
   - Click **Rule File** in the **Inspector** 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 use **File** > **Save** from the menu.

Note:

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

Manage rule errors and warnings



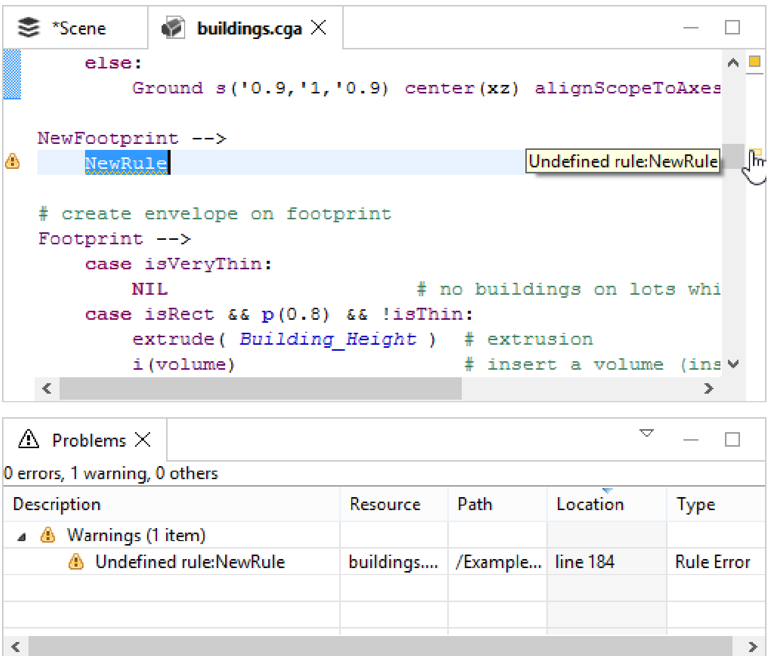
Rule file with syntax error

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

Note:
The position of errors are indicated as small red boxes next to the scrollbar on the right. More detailed information about the error can be found in the **Problems view** or by hovering over the red indicators or error markers in the **CGA Editor**.

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

The **CGA Editor** also issues warnings:

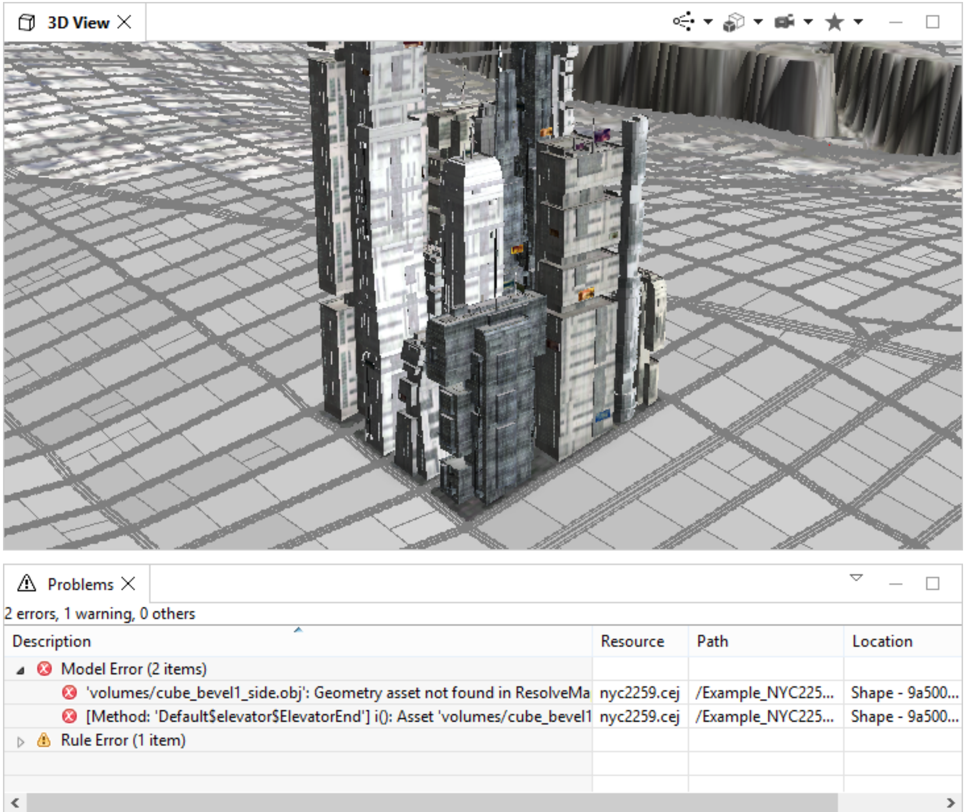


Rule file not defined warning

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

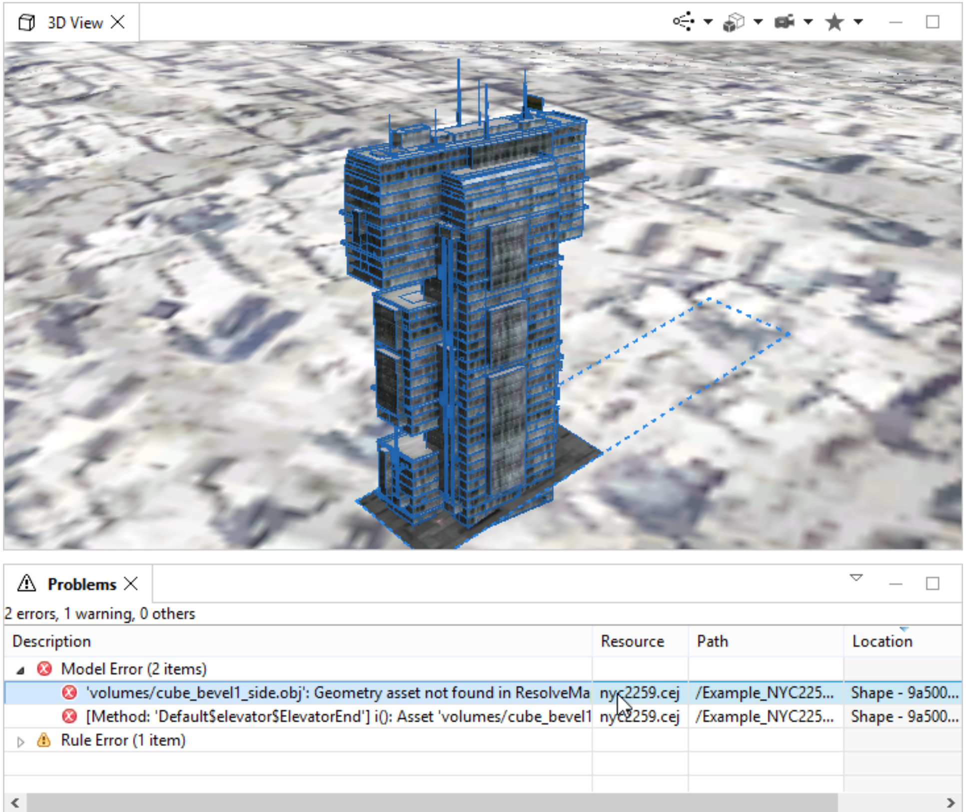
Problems view

You can open the **Problems View** using **Window** > **Show Problems** . On top of Rule Errors (static compile errors), the **Problems View** also shows Model Errors (dynamic runtime errors), i.e. problems encountered during generation of a model. Such errors and warnings depend on the rule as well as on the initial shape (i.e. its geometry and attributes such as the seed etc.). The **Problems View** is a great aid in finding and resolving such problems. In the example below a number of buildings were generated and an "asset not found" error was reported.



Model error

To find the according model or shape, double-click on 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.



Asset not found error

Configuration

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

- Right-click in the **Problems View** and select **Broup By** > **Type** to group the errors by their type (i.e. separate Rule Errors and Model Errors):
- Click **Location** to sort the errors by Location (i.e. 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** features automatic code completion. At any position in the CGA code, you can press `Ctrl + Space` and a window pops up with a number of suggestions which match the current context. Use the cursor keys or the mouse to choose one.

Important shortcuts

Very important shortcuts for working with the **CGA Editor** include:

- `Ctrl+S` —save the file (changes must be saved before generation; files with changes are marked with an asterix * in the tab)
- `Ctrl+G` —generate (the selected objects, i.e. shapes or models)
- `Ctrl+F5` —re-generate all models

- `Ctrl+F` —opens "find / search-replace" dialog
- `Ctrl+L` —opens "go to line" dialog
- `Ctrl+Shift+L` —shows all shortcuts

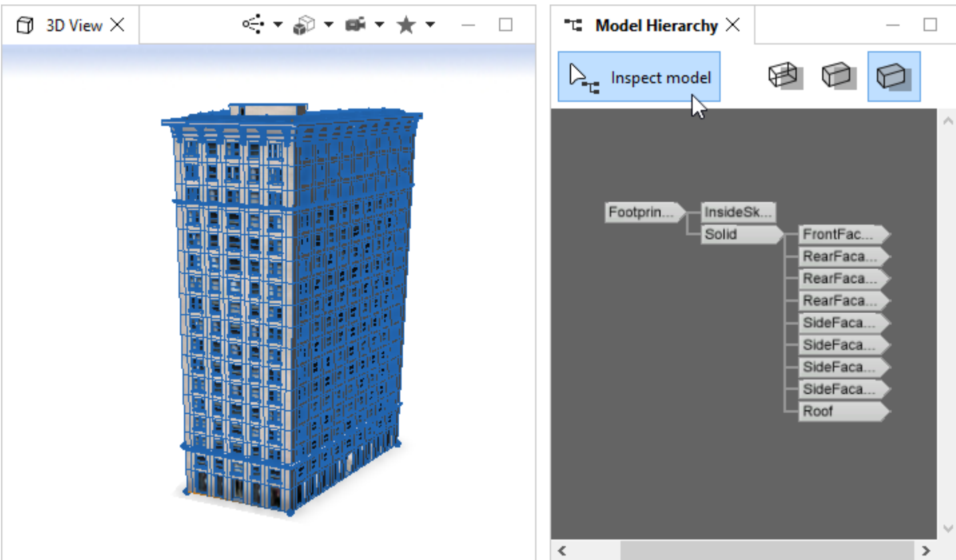
Note:
These shortcuts only work if the **CGA Editor** is the current view (i.e. its tab is highlighted).

Model Hierarchy

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

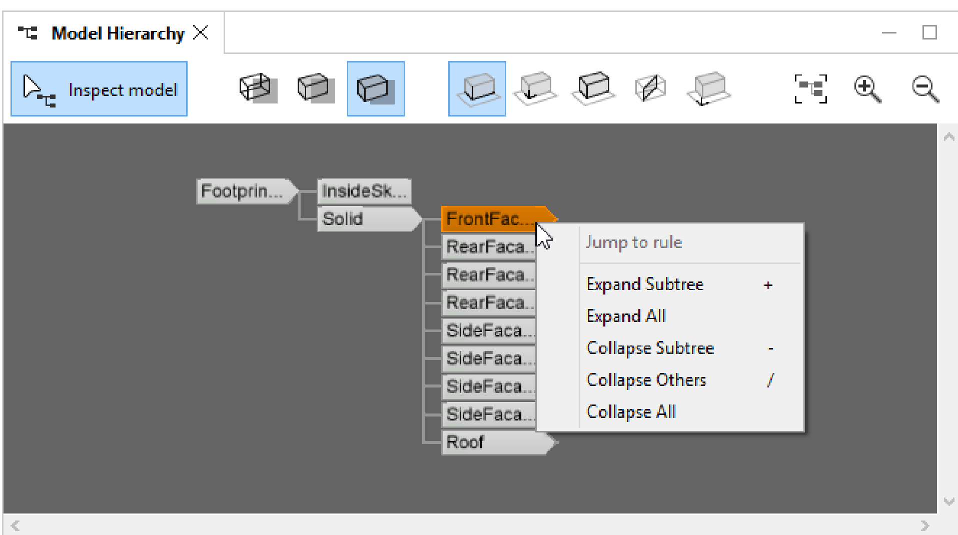
Inspect model

Generate a model, select it and click the **Inspect Model** tool in the toolbar of the **Model Hierarchy**:



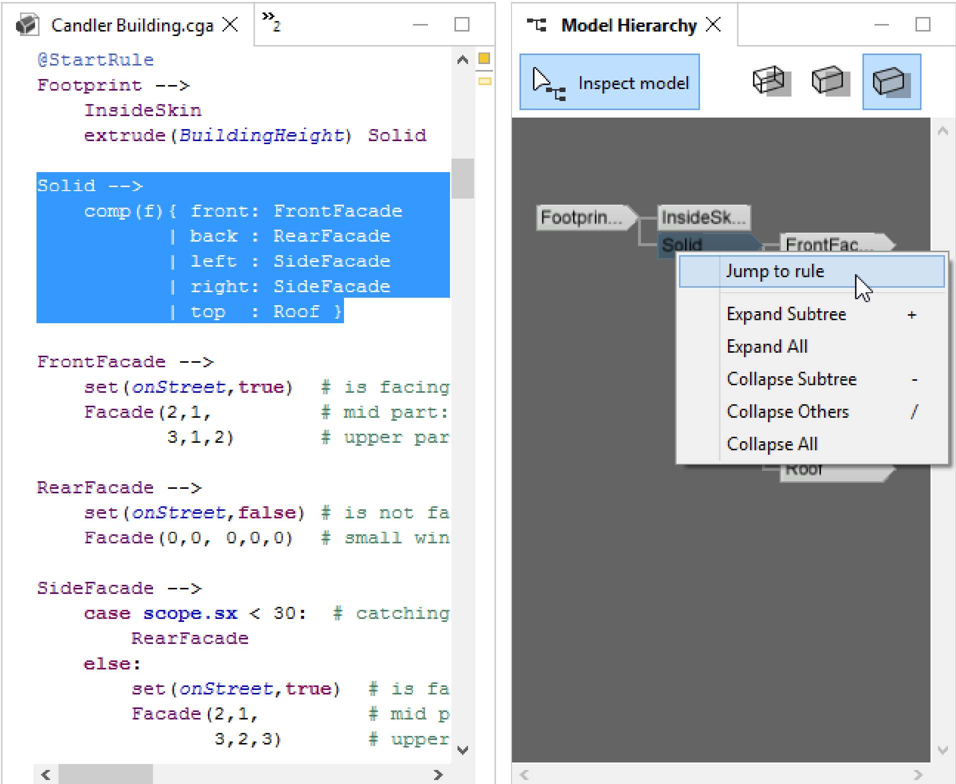
A generated model switched to the edit model; the hierarchy appears in the Model Hierarchy Explorer

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



The right-click menu expands and collapses model hierarchy nodes

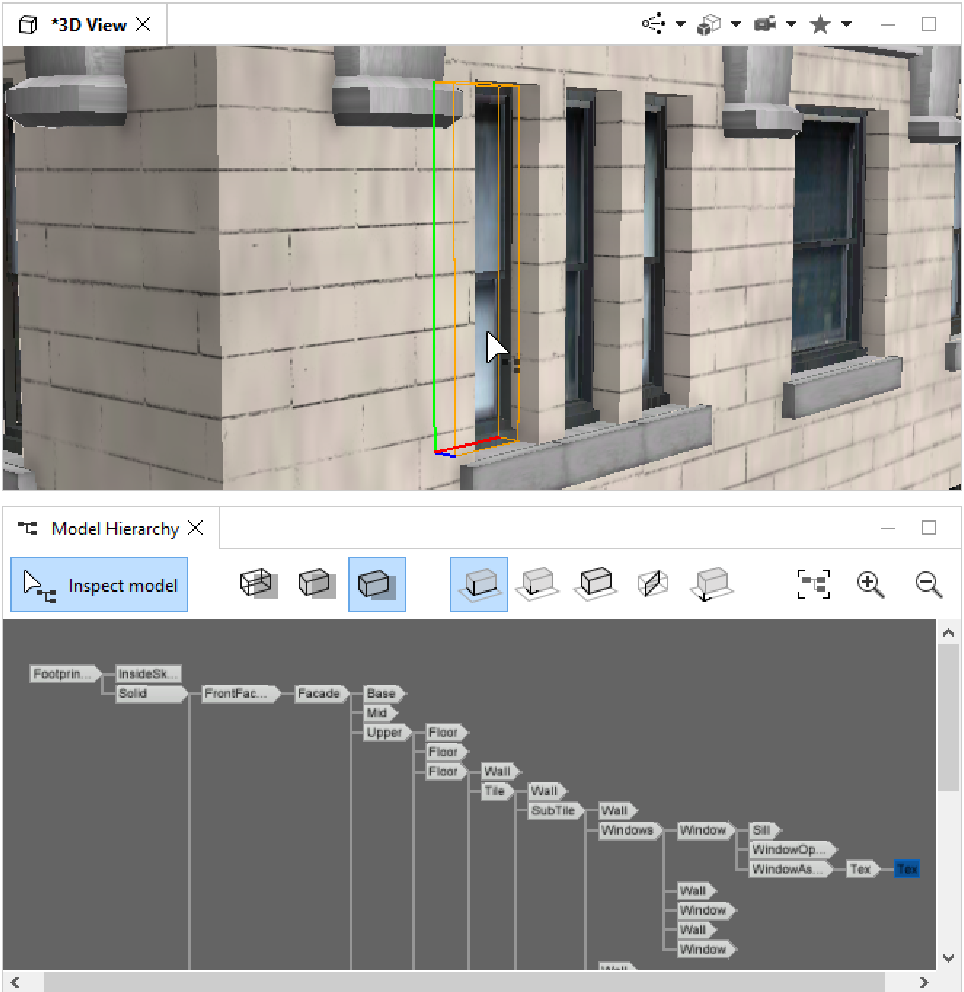
The shape tree of a specific building is defined by the associated rule file and the initial shape. Double-click on a node or choose **Jump to rule** in the context menu to jump to the respective rule in the rule file.



Part of a CGA rule file and the hierarchy of the generated model

The pictures above show snippets from the `Candler` rule file. The rule file and the model hierarchy structures match (e.g. 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/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**:

Inspect model	Activate inspect model mode.
Transparent model	Set the drawing mode of the selected model to transparent.
Translucent model	Set the drawing mode of the selected model to translucent.
Opaque model	Set the drawing mode of the selected model to opaque.
Show scope	The selected shape nodes' scopes are drawn (x-axis red, y-axis green, z-axis blue, other scope edges orange).
Show pivot	The selected shape nodes' pivots are drawn (x-axis red, y-axis green, z-axis blue).
Show geometry	The selected shape nodes' geometry is drawn.
Show trim planes	The selected shape nodes' trim planes are drawn.
Show origin of model	Tthe 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 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** 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**.
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**.
3. Load the selected shape or image into the **Facade Wizard** by clicking the **New Facade** button in the **Facade Wizard** toolbar.
4. Click the **New Facade from Image** 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 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** option .
    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** creates CGA code "one way" only, i.e., you cannot "load" a facade with associated rules. However, the created CGA code can easily be edited using the **CGA Editor**.

Additional considerations

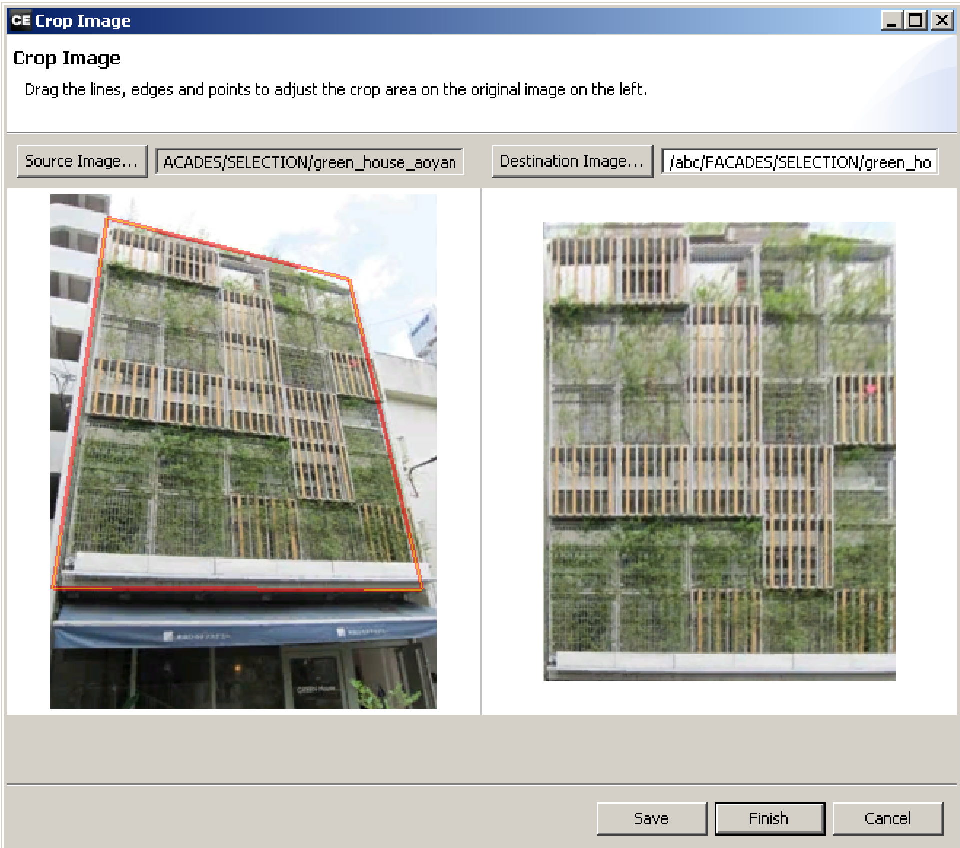
Flexible vs. rigid splits	Normally, if you have multiple splits along an axis (X or Y), the Facade Wizard will assign flexible splits automatically, so a created facade can adapt to different shapes. However, in some cases, it is necessary to manually define split types. Move with the mouse over a region 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 will ask for an approximate initial width. When loading a shape, the dimensions will be taken from the actual shape size. **Note:** A more precise width or height can be specified at any time using the context menu's "Set Region Width / Height..." entry.
Select region	Using the context menu's "Select Region" 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 `Shift` when adding splits, and the split will snap to previous locations. The snapping history is kept when loading new facades, making snapping very 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**.

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 in order 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 **Destination Image...** path

Perspective frame manipulation

- 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:

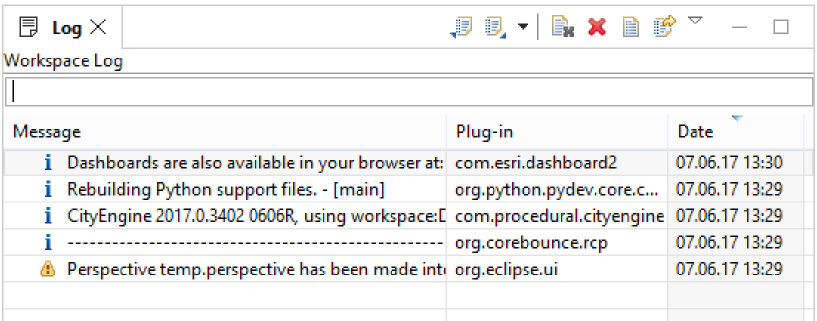
- Zoom: Mouse wheel.
- Pan: Move mouse while pressing middle mouse button and `ALT`.

Status windows

CityEngine includes additional windows to give you important information about your scene, such as progress status of operations, log records, or errors.

Log

The **Log** window shows the log records of the CityEngine. You can open the **Log** by selecting **Window** > **Log** from the main menu. Log records are created by various parts of the 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** view is especially useful in tracking down strange or erroneous behavior.



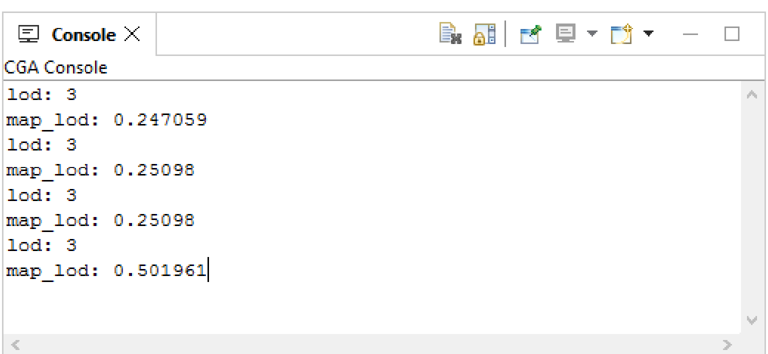
Log window

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 different consoles used when working with CityEngine. The top toolbar button switches between different 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

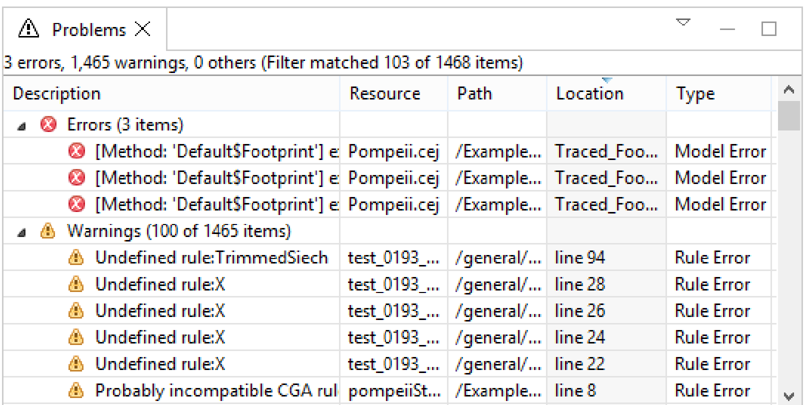
The Python Output console is the default output console for python statements that use a print() command. This console is available once a Python print output is produced.

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



Problems window

In the views settings of the **Problems** view (triangle in top toolbar), change the grouping to by **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 on the red **Stop** button on the right-hand side of the operation. You can open the progress view by selecting **Window** > **Progress** from the main menu.

Progress window

Note:

You can always 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.

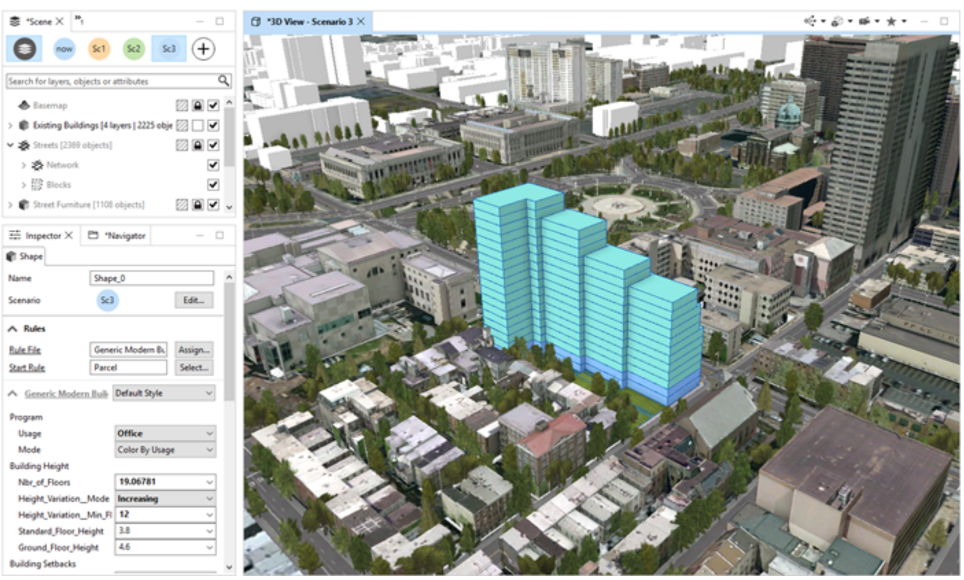
Default

The **Default** layout has one big perspective view with editors open to allow for scene overview and management tasks like import / export.

Default layout

Compact

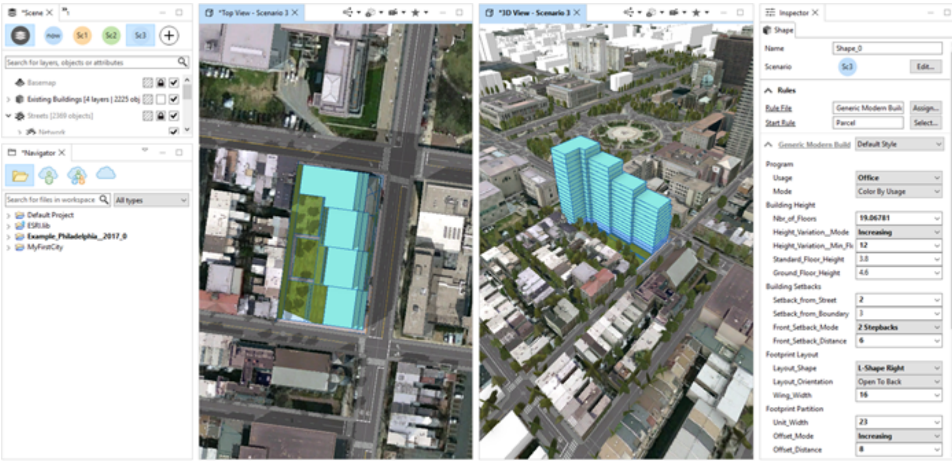
The **Compact** layout has one big perspective view and some editors open. This makes it suitable for presentations.



Compact window

Top & 3D View

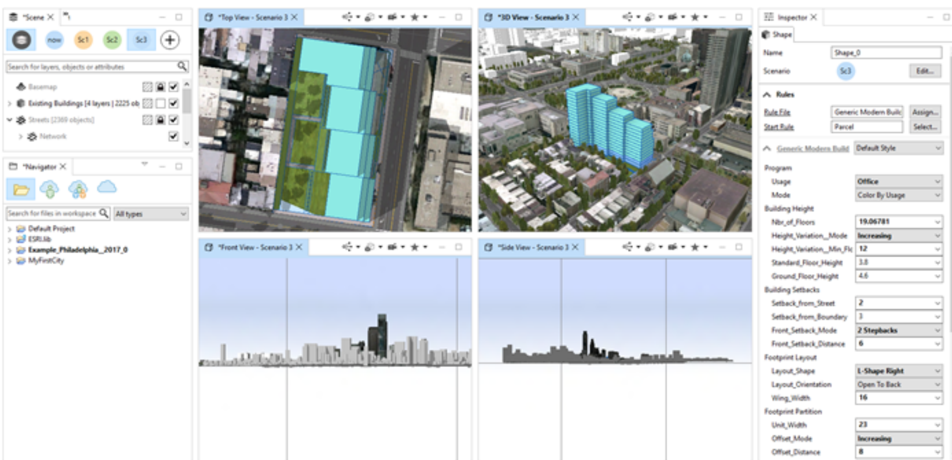
The **Top & 3D View** layout has a perspective view and a top view with space for editors on both sides.



Top & 3D View layout

Top, Front, Side & 3D View

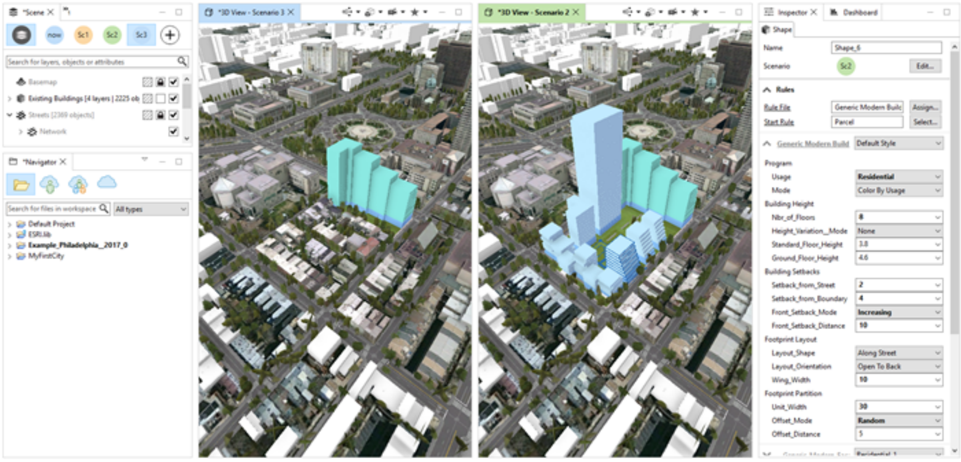
The **Top, Front, Side & 3D View** layout is the classical CAD layout with a perspective, top, front, and side view.



Top, Front, Side & 3D View

2 Scenarios Side-by-side

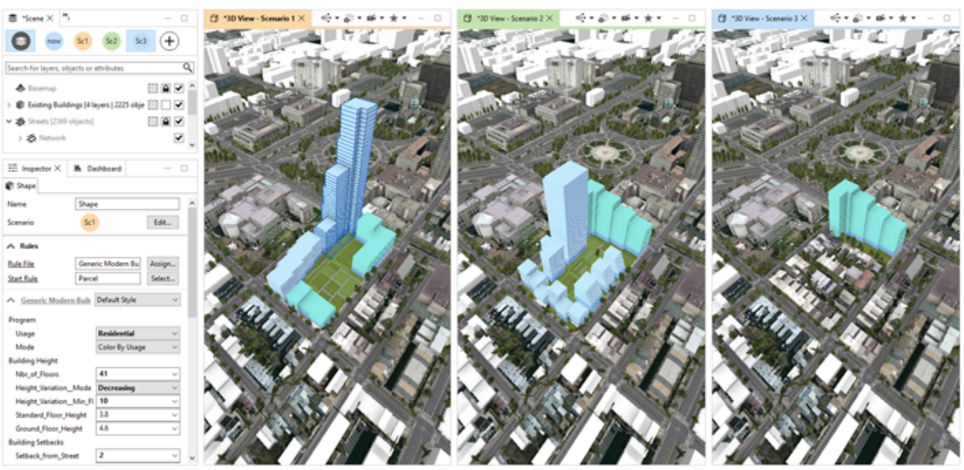
The **2 Scenarios Side-by-side** layout has two perspective views with different scenarios assigned to it. This layout is especially suitable to compare scenarios.



2 Scenarios Side-by-side

3 Scenarios Side-by-side

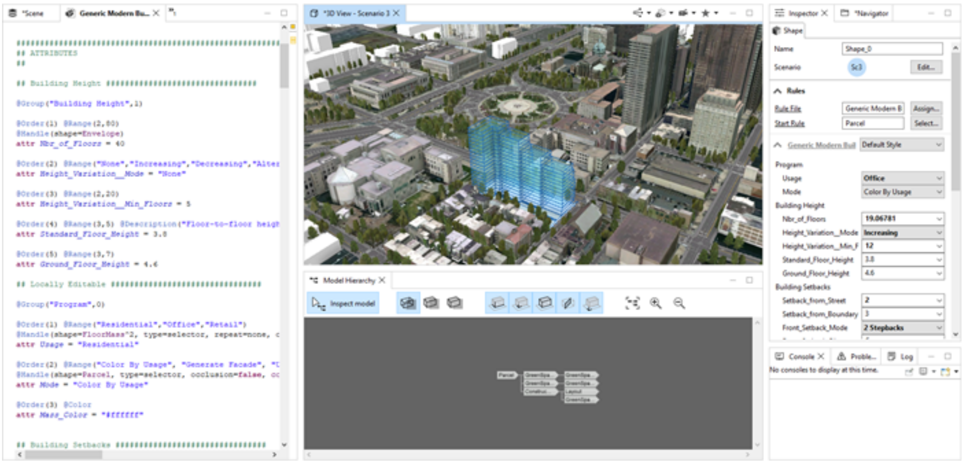
The **3 Scenarios Side-by-side** layout offers three perspective views for different scenarios.



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

Multiple Viewport Windows

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

The following list 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**. 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 overview

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 work efficiently.

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 filesystem where all your projects are stored. This makes it is very easy 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 filesystem 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.

Creating a new workspace

To create a new 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 current workspace.

- **Working Sets** — Inherit user-defined working sets from current workspace.

Working 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 on 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 to a given scene or group of scenes are organized in a project. A project is a collection of folders for the following different 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 like **Viewport** snapshots are stored here.
- **maps**: This folder contains the image maps used by the map layers. For example, a height- 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 on 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 to 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 opens, do the following:

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

Importing 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 among the following import options for importing:

- **Select root directory** — Root directory in the file system to start scanning for projects to import. Type in the full path or Browse to select the path on the file system.
- **Select archive file** — Archive (zip) file to scan for projects to import. Type in the full path or Browse to select the archive on 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 to always turn on the option **Copy projects into workspace**. Exceptions are limited storage space or collaborative work environments.

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

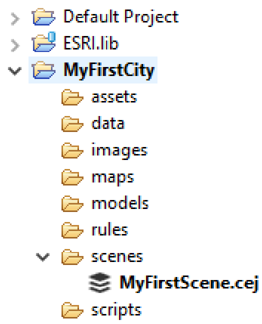
Create a new scene

To create a new 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 opens, 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 a one by clicking **Choose**.
6. Click **Finish**.
   The scene is added to the project in **Navigator**.



Scene 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 needs to 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 CGA Editor window is active.
2. To save the CGA rule file, you can do any 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 to make a copy of the `ESRI.lib` CGA rule file.

Import a file

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

Import local files into project

To import files into your 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 would like to import. Recent directories that have been imported from are shown on the **From directory** field's combo box.
   - In the right pane check the individual 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 checkbox (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 easily 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 which 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 project

1. Click **File** > **Import**.
2. Click **Files into Existing Project** > **Archive File**.
   In the right pane check the individual 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 checkbox (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 in the full path or Browse to select the path on 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 which 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 simplest 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

In order 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 on 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 hierarchical folder structure in the file system as it exists in the workspace.
- **Create only selected directories** — Creates hierarchical structure in the file system only for selected folders.

Export a project to 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 **Browse** to select a path.
- **Overwrite existing files without warning** — Determines whether exporting a resource should silently overwrite a resource which 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 hierarchy (folder) structure in the file system as it exists in the workspace.

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

Scenes

Scenes overview

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** you may use the navigation tools to track, tumble, dolly, or look around. You can select the appropriate tool in the toolbar and click and drag the mouse in the **Viewport**. Alternatively, these actions can be applied any time using a modifier. The following is a table of helpful navigation actions:

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

You can click an object to select it or drag a selection rectangle and then press `F` to frame the selection. Afterwards 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 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 might 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 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.

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.

Tool options

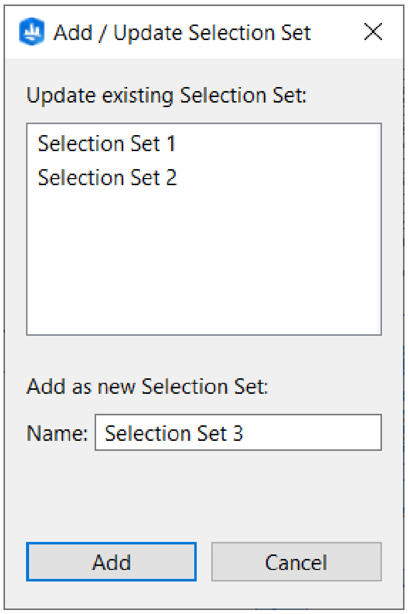
The **Selection Tool** includes the following options:

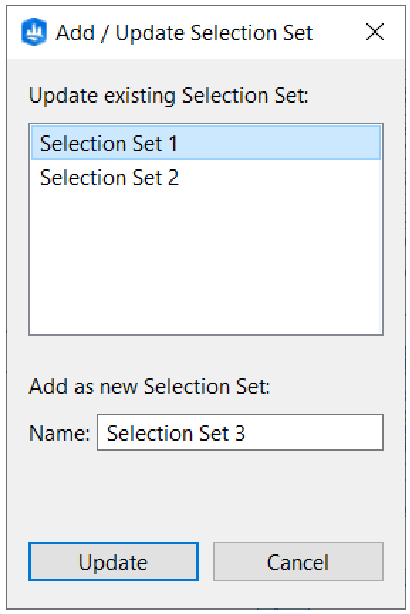
Selection Tool	
Mode	• Select • Add to selection (`Shift`) • Subtract from selection (`Shift+Ctrl`) • Invert selection (`Ctrl`) Press and hold the modifiers in the parentheses to change the mode.
Disable Highlight	Hides the selection highlighting while the object remains selected. This is useful when modifying CGA attributes in the Inspector.
Selection Sets	See the Selection sets 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.





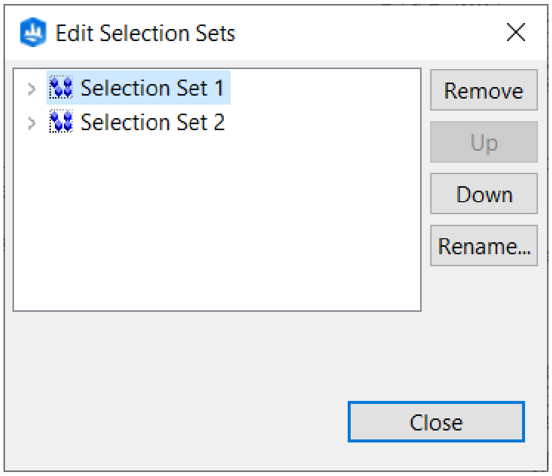
4. Click **Cancel** to close the window without adding or updating selections sets.

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 Viewport

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 offers a subset of selection methods that are described in the Select main menu.

Select in Scene Editor

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

Click	• On a layer/group layer: selects the layer, but does not select the scene objects inside. • On a scene object: selects individual object, deselects already selected objects.
Double-click	• On a layer/group layer: has no effect. • On a scene object: select and frame individual object.
`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 or in the Viewport.

Select All, Deselect All, Invert Selection	• In this case, "All" means scene objects that are not locked and with visibility = true. Map layers are not included. • Select all and deselect all can also be triggered using the shortcuts `Ctrl+A` and `Ctrl+Shift+A`. • Invert the selection of all selected objects.
Select Objects in Same Layer	All objects in the same layer or layers are selected.
Select Objects by Map Layer	See Selection with image maps for details.
Select Objects in Same Layer Group	All objects in the same layer group or groups are selected.
Select Objects of Same Type	Selects all objects with the same type or types as the current selection.
Select Objects of Same Group	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.
Select Objects with Same Rule File	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.
Select Continuous Graph Objects	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

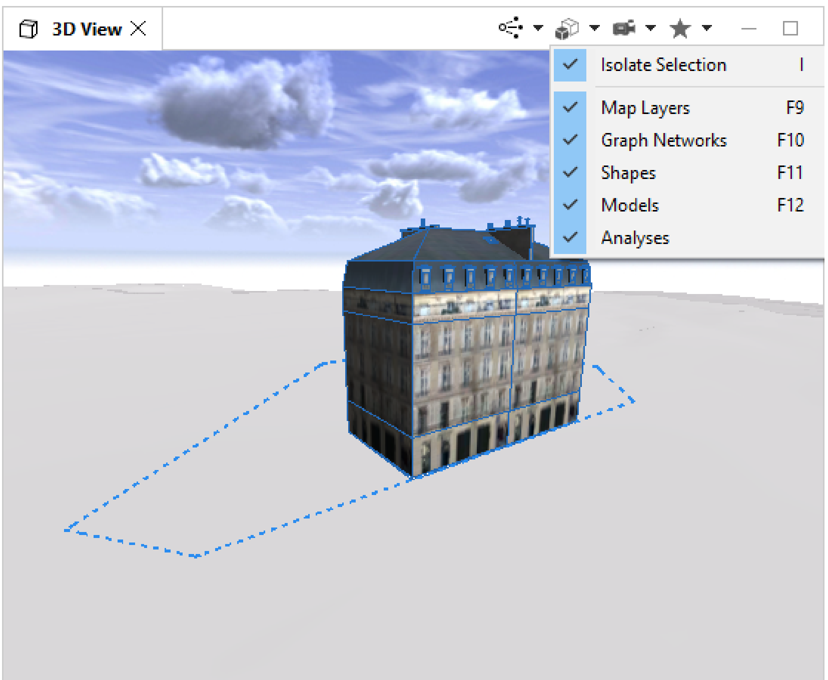
When you create multi selections, the highlighted color of the "lead selection" object varies from the color of the other selected objects. This is because the lead selection is used for operations where additional information is needed. For example, the subtract tool uses the lead selection to determine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

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.

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) 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**. 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 always align the camera along a specific axis by pressing `X`, `Y`, or `Z`. While `Y` only orients the Y-Axis in camera direction, `X` and `Z` also re-orient the Y-axis upwards. This, in combination with switching between orthographic and perspective view (P), allows you to quickly walk through multiple views of your scene.

See cameras in **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 new camera from the current view.
2. Click **Window** > **New Viewport** > **Manage Cameras** or **Edit** > **Preferences** > **General** > **Viewport** > **Cameras** to manage and edit cameras.
   This opens the Camera preferences menu. 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 (e.g. by rotation) affects the second viewport with the same camera, as well.

Configure cameras

Camera configurations have the following options:

- **Perspective Projection View** — Lets you 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 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 they were created from and can only be applied to this camera.

To create a new bookmark, do the following:

1. Adjust the camera as desired for the bookmark.
2. Click the **Bookmarks** menu.
3. Click **New Bookmark...**
   The first ten 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

To edit a bookmark, do the following:

1. Click the **Bookmarks** menu.
2. Click **Edit Bookmarks...**.
   You can also access **Edit Bookmarks...** by selecting **Edit** > **Preferences** > **General** > **Viewport** > **Bookmarks**.

In the **Bookmarks** manager, you can create new bookmarks, change the order of the bookmarks in the list, rename bookmarks, and update the camera properties.

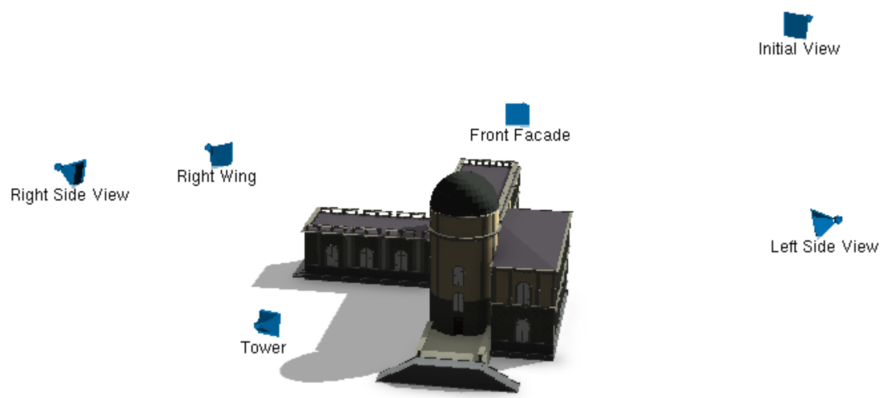
A bookmark has the same settings as a camera.

Tip:

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

Bookmarks visibility

Toggle 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.



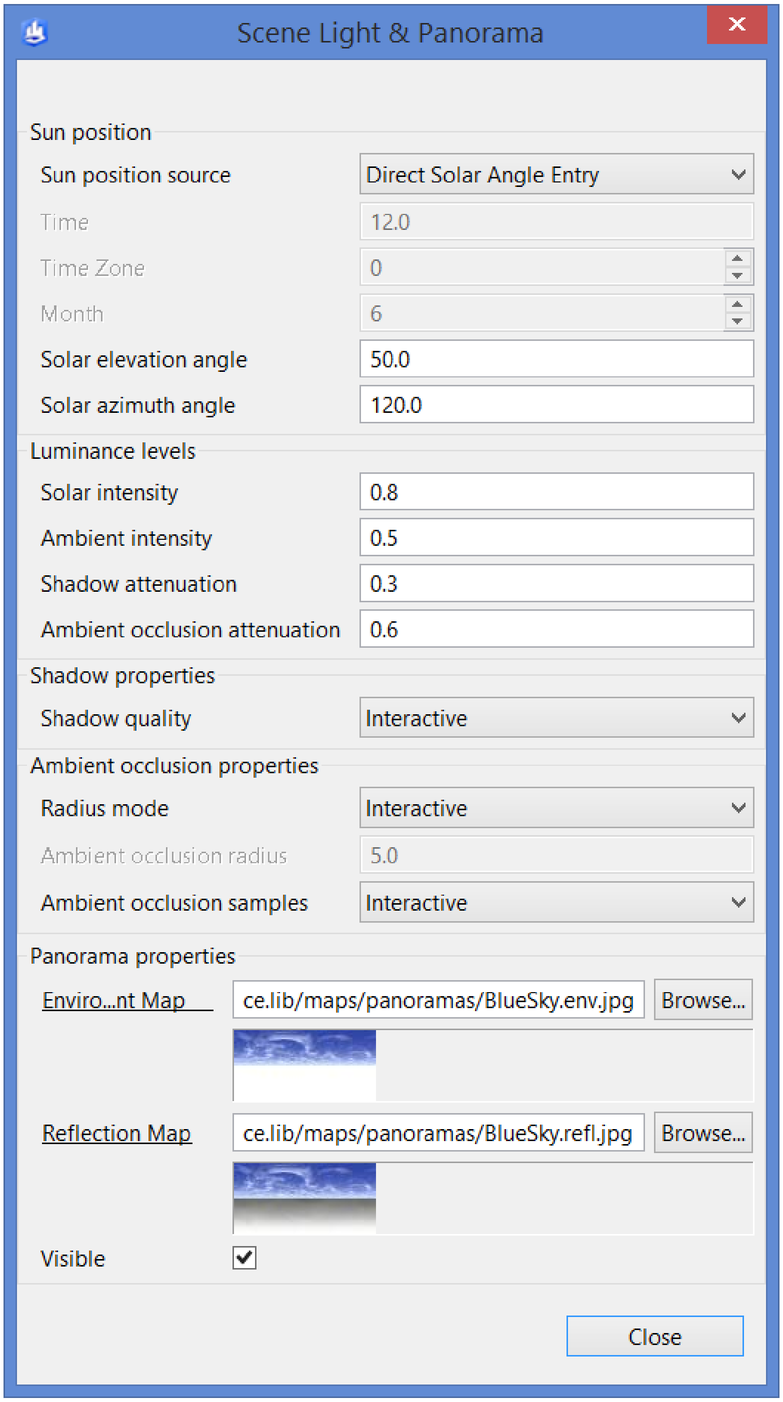
Bookmark positions with gizmos

Tip:
Turn on Bookmarks visibility when creating a CityEngine scene that will be exported to 360 VR Experience.

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



Scene light and panorama settings

Light

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

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

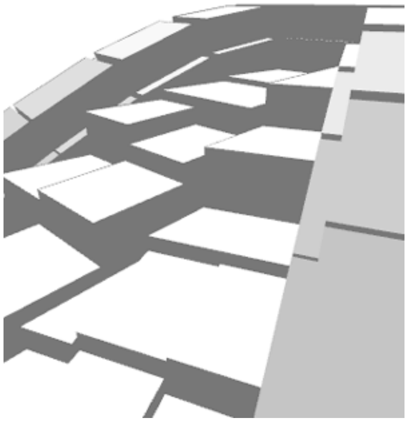
Interactive modes

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

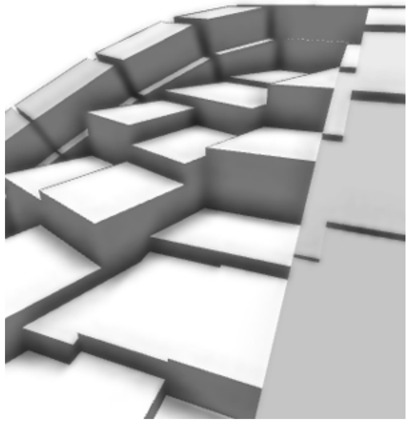
- When **Shadow quality** or **Ambient occlusion samples** are set to **Interactive**, the best visual quality is used when the camera stands still, while the quality is reduced based on performance during camera moves.
- When setting **Radius mode** to interactive, the ambient occlusion radius is automatically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

Ambient occlusion

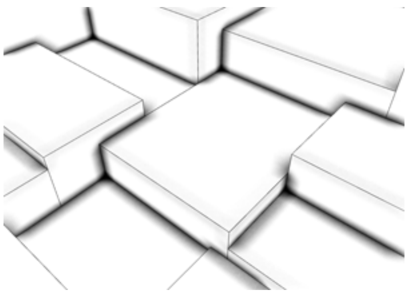
CityEngine ambient occlusion uses screen space ambient occlusion, a special real-time rendering technique that approximates ambient light occlusion. Ambient occlusion is a shading method used in 3D computer graphics which helps add realism by taking into account attenuation of light due to occlusion.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



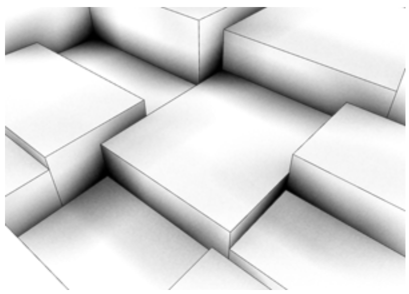
Solid shading without ambient occlusion



Ambient occlusion enabled



Low ambient occlusion radius



High ambient occlusion radius

Panorama

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

Environment Map	• Image to be used as environment map. • The environment map is used to texture the sky in your scene.
Reflection Map	• Image to be used as reflection map. • The reflection map is used on reflections on generated models that have reflection enabled in their material settings.
Visible	Show or hide the panorama.

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

Georeferencing

CityEngine provides different 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 the **View settings** > **Information display**.
2. This displays scene information, such as **Scene coordinate system**, grid size, and coordinates of the cursor.

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

1. Click **Window** > **Show Status Line**.
2. The **Status Line** displays in the bottom 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)**.

Setting 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**:

- Setting a coordinate cystem when georeferenced data is added to the scene.
- Setting a coordinate system when creating a new scene.
- Changing the **Scene Coordinate System** in the preferences.

Setting 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 correclty. 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 suggests a matching scene coordinate system.

Setting a coordinate system when creating a new scene

When creating a new 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...**.

Changing the Scene Coordinate System in the preferences

The **Scene Coordinate System** can be set or changed in the CityEngine preferences **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 to 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**.

CityEngine CS [meters]	CityEngine coordinates y-up, meters
Scene CS [feet]	Current Scene Coordinate System in feet
Scene CS [meters]	Current Scene Coordinate System in meters
UTM	Universal Transverse Mercator
MGRS	Military Grid Reference System
Long/Lat [decimal degrees]	Longitude / Latitude in decimal degrees
Long/Lat [degrees min sec]	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 changing the coordinate system option isn't available. See Setting the coordinate system for more information.
- The **Scene Coordinate System** in the **Status Line** window remains unchanged.

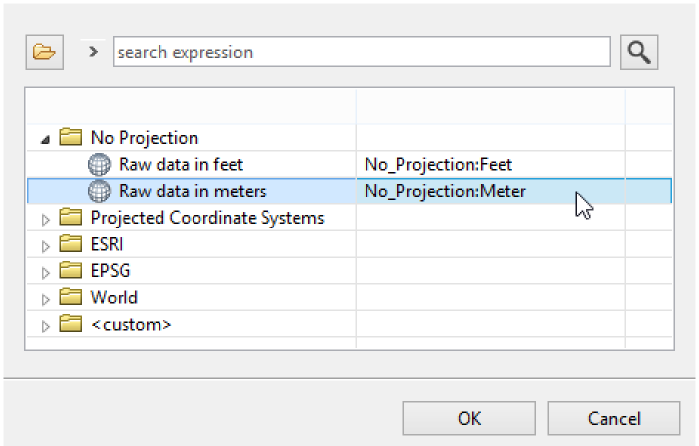
Coordinate System dialog

The **Coordinate System** dialog allows you to set the coordinate system when creating a new 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 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 option **No Projection** > **Raw Data in Meters** is a good choice.

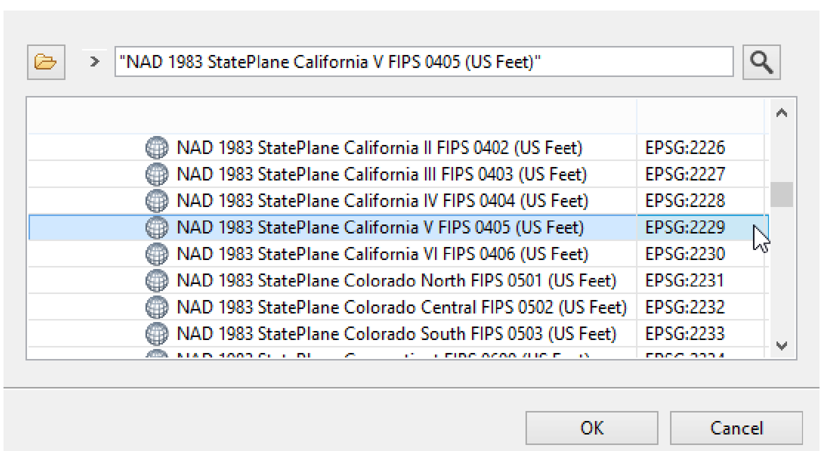


No projection in meters

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

Select data coordinate system

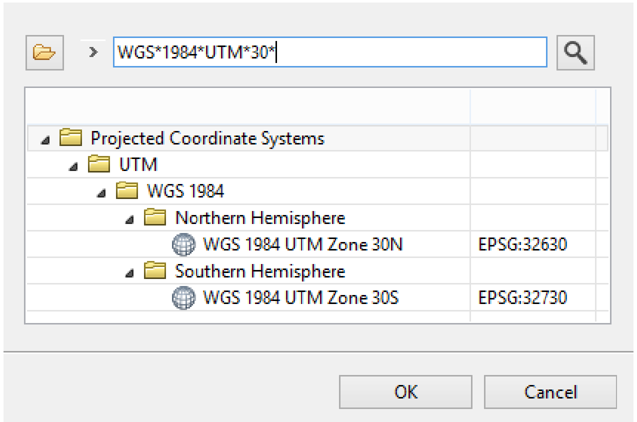
The data coordinate system pops up whenever georeferenced data is going to be imported, and no projection details are found with the data. Usually you will see a coordinate system suggested the **Scene Coordinate System** when this dialog appears, which is a good option.



Set the coordinate system

Using 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).

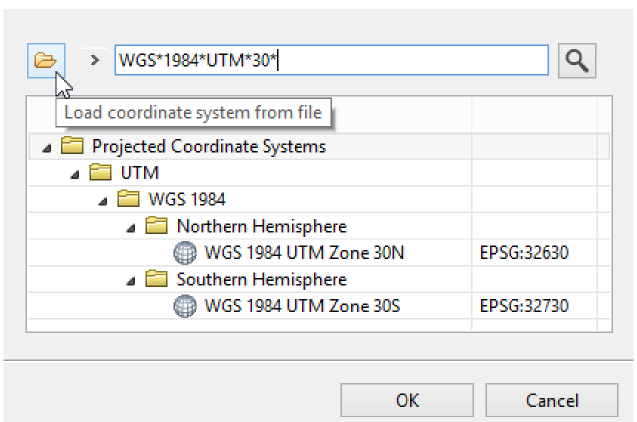


Search for coordinate system

Load 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 top left to bring up the file dialog, and browse to the `.prj` file.

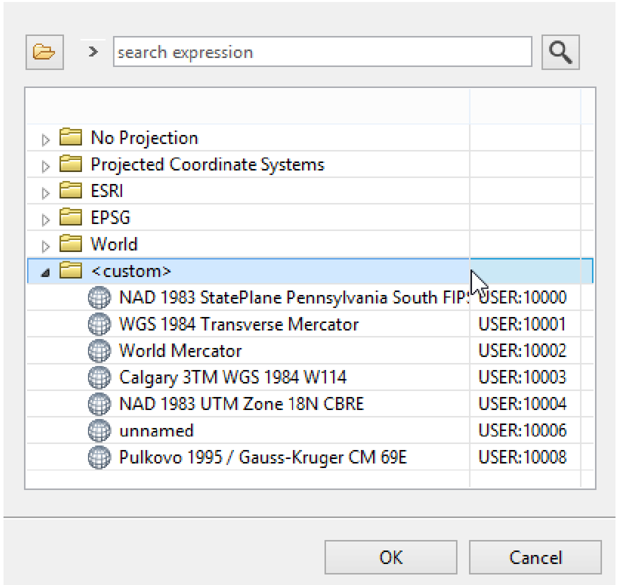
This feature also allows you to define your own custom coordinate system by creating your own `.prj` file with the desired parameters.



Load coordinate system from 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 authority code.



Custom coordinate system

Importing 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 which lets the user 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 lets the user set the **Scene Coordinate System**.

Shapefiles (.shp)	CityEngine looks for a `.prj` file with the same name in the same folder.
File Geodatabase (.gdb)	The coordinate system is read per layer of the gdb dataset.
KML/KMZ	KML/KMZ latitude/longitude data is always interpreted as WGS 1984. When KML/KMZ data is imported into a scene with no coordinate system set, UTM with matching zone is suggested as Scene Coordinate System.
OpenStreetMap (OSM)	OSM latitude/longitude data is always interpreted as WGS 1984. When OSM data is imported into a scene with no coordinate system set, UTM with matching zone is suggested as **Scene Coordinate System**.

Georeferenced image data

Image data is considered to be georeferenced if:

- 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 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 coordinate system for the data. If no `.prj` file is found, the Select Coordinate System menu opens and lets the user choose the coordinate system for the data.

Map extent

CityEngine looks for a world file (e.g. `.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, 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 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 (e.g. 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 non-georeferenced images in the Map Layer menu.

Scene Objects

Scene objects overview

A scene consists of different types of scene objects organised 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. In order 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 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 geopreferenced shapes is through ArcGIS Online and the Get map data feature.

The Non-geo-referenced 3D formats Wavefront OBJ (.obj), Autodesk FBX (.fbx), and Khronos Group glTF (.gltf, .glb) 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 edges that connect the nodes so that a mesh is formed. A graph layer holds edges and nodes as well as shapes that are created automatically.

Each edge 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 edge/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.

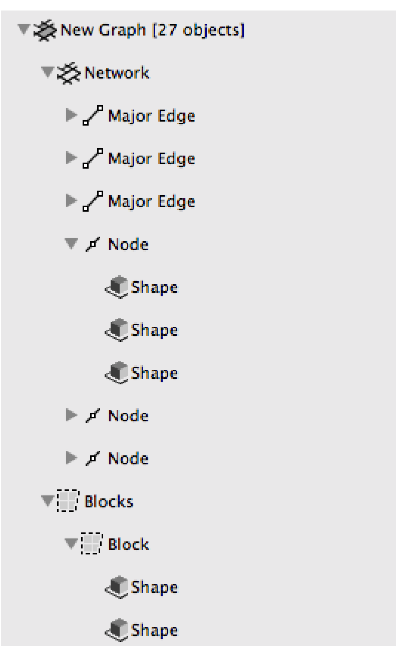
Blocks

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 edge, 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 with Edges, 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.

Static models

A static model is any geometrical model imported "as is", without modification by CGA rules, textures, or individual vertices within CityEngine. They 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 your scene.

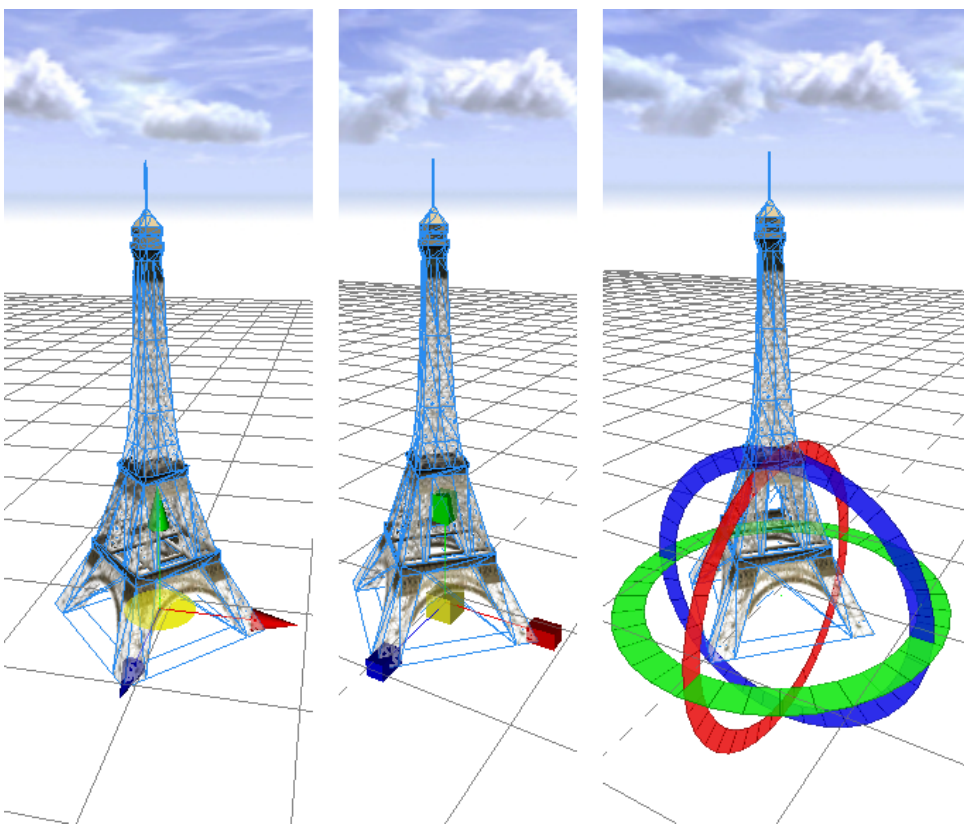
Note:

- 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 entry **Asset_file** in the **Inspector** to fix the reference.

Place a static model

A static model can be positioned and scaled as you wish within the scene by the means of 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 be saved in source data. The settings are saved in the scene and re applied to the source when reopening the scene.

Asset_File	Location of the asset in the workspace. Especially useful in case the reference to asset got broken.
Material_Colorize	• Colors the model to increase visibility of an object in the scene. • Setting it to white (#ffffff) turns tinting off.
Material_Transparency	Control the transparency of the model.
Mesh	Cleans up and reduces the geometry. It can help to improve the frame rate in the viewport.
Normals	Control 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).
Pos_Bottom_Align	Places the origin of the model to the bottom or to the center of the model.
Pos_Center	Places the origin of the model to the center or to location as it is defined in the source file. If `Pos_Bottom_Align` is set, the vertical center is overridden (default, except fot KML import).
Pos_zUp	Some models are created in a z-up coordinate system instead of the CityEngine y-up system. Use this switch in order 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 would like to align to a terrain
2. Click **Terrains** > **Align Static Models to Terrain...**

Parameters

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

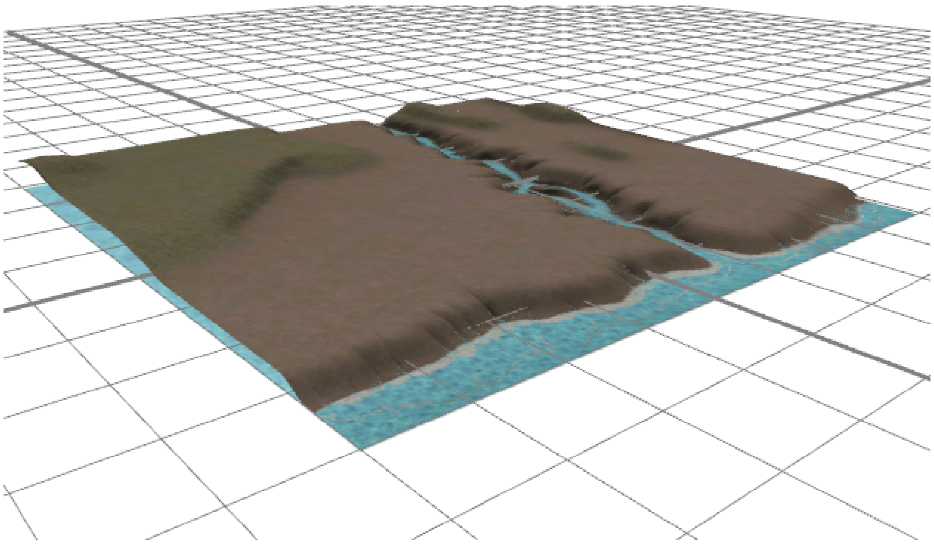
Note:
The model is not draped onto the terrain. The point of origin only is placed exactly onto the terrain (offset not included). Therefore other parts of the model might disappear in the terrain or hover above it.

Map Layer

Map layer overview

A map layer is used mainly to add terrain and satellite imagery to your 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

- 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 your scene. Useful for water bodies.

Auxiliary map layer types

They are used to influence the behavior of procedural functions.

- Obstacle layer — Defines the areas where the street grow algorithms are prohibited to create streets. It can also be used to select objects based on their location on the map layer. An **Obstacle Layer** offers a true/false information for each location on the map.
- Mapping layer — An arbitrary combination of image map channels and mathematical functions. Typically used to control the height or usage type of a building based on its location. A map layer can offer 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.

Creating a terrain layer

A terrain layer can be created in the following ways:

- An image is dragged from the Navigator into the scene.
- A terrain is imported with **File** > **Import** > **Terrain Import**.
- A new terrain layer is created with **Layer** > **New Map Layer** > **Terrain** .

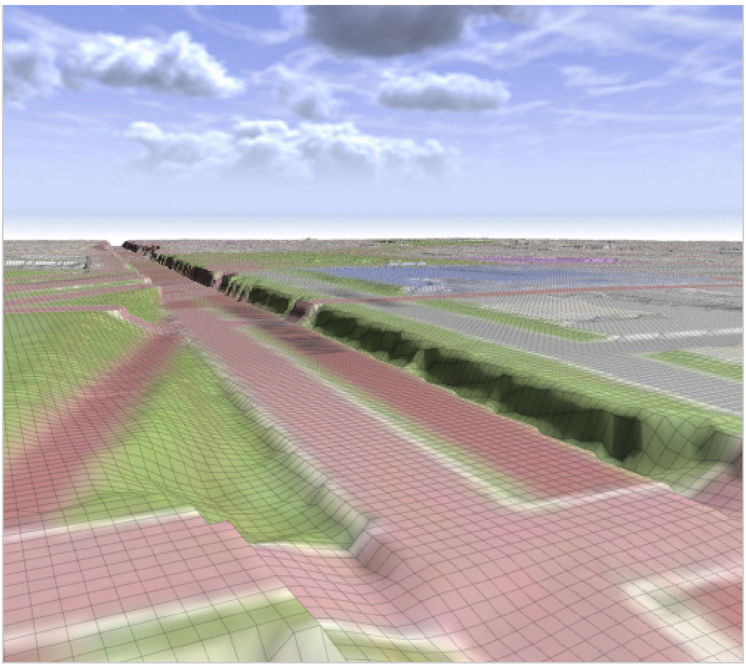
Terrain options

Heightmap File	Choose an image from your workspace which will be used as heightmap mesh / Digital Elevation Model. Normally this is a greyscale image. When a georeferenced image is selected, elevation and bounds are set automatically.
Texture File	Choose an image from your workspace which will be used to texture the heightmap mesh. **Note:** 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 filetypes like 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 like 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. **Note:** • The button right of **Location** can be used to change the reference point of the terrain's position. • It is recommended to use 32-bit float geoTiffs. • In addition to 8-bit images, also 16-bit and 32-bit images are supported for heightmap files. The 16/32-bit range is scaled to the elevation bounds similar as it is done with standard images.

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 on the terrain layer and choose **Frame Layer**.

Note:
Terrain layers (like all map layers) can not be selected in the Viewport directly, but only through the Scene Editor.



Terrain layer

Inspector options

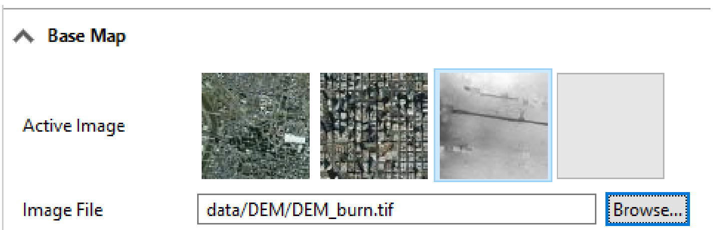
To change the options of a terrain, you can select the terrain layer in the Scene Editor and review its attributes in the **Inspector**. with the following options:

Name	Layer name
Visibility	Layer visibility
Locked	Lock terrain layer
Transparency	Layer transparency
Color	Color value multiplied onto layer
Wireframe	Wireframe enabled/disabled
Dimensions	Extent of layer
Location	Location of layer
Elevation Offset	Elevation offset applied to layer. This value is also available as a built-in function in the **Layer Attributes** code.
Heightmap File	Heightmap file location
Resolution	The number of terrain mesh vertices in u and V direction.

Apply Alignments	If enabled, the built-in function "elevationDelta" in the **Layer Attributes** code returns the elevation deltas resulting from the terrain alignment tool. If disabled, the function returns 0.
Heightmap Sampling	**Bilinear** and **Nearest Neighbor (hard edges)** determine the smoothness of the elevation.
Minimum / Maximum Height	• The elevation data from the image is mapped into the range [Minimum, Maximum]. • Both values are also available as built-in functions in the **Layer Attributes** code.
Layer Attributes	Contains the (editable) cga code for evaluation of the elevation.

Base map

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



Base map properties in Inspector

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(s) to export.

Export terrains to 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's 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 chose 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's with your organization or the public.

4. Right-click the TPK file you want to share and click **Share as....**.
   This opens the share **Tile Package** dialog.
5. Fill out 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 image

To export to image files, select **File** > **Export...** > **CityEngine** > **Export Selected Terrains as Image** and choose a format.

Export terrains to geometry

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 the scene.

Create a texture layer

A texture layer can be created in the following ways:

- A texture layer is imported **File** > **Import** > **Texture Import** in the main menu.
- Click **Layer** > **New Map Layer** > **Texture** in the main menu.

The **Texture** dialog opens.

Texture layer options

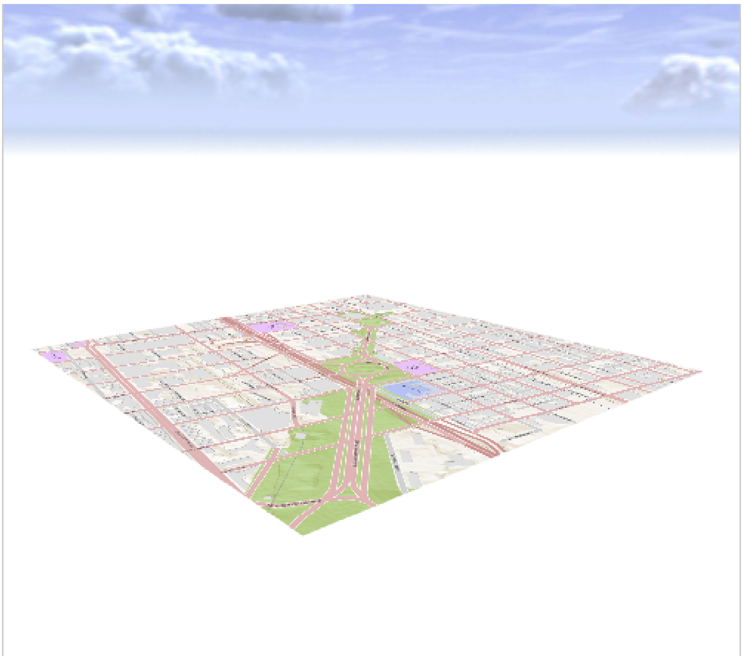
You can modify the following options when you open the **Texture** dialog:

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

Texture layer in 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 on 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.



Texture layer

Inspector options

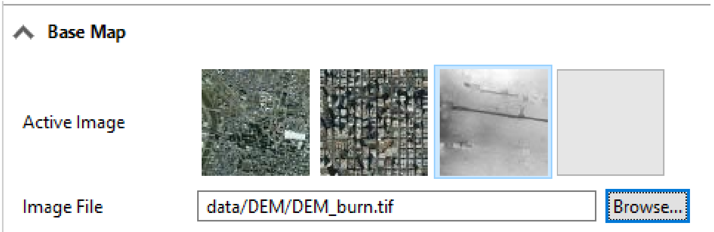
You can select a texture layer in the Scene Editor to change the following parameters in the Inspector:

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

Minimum / Maximum Height	• The elevation data from the image is mapped into the range [Minimum, Maximum]. • Both values are also available as built-in functions in the **Layer Attributes** code.
Layer Attributes	Has no effect on the rendered texture.

Base map

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



Basemap properties in Inspector

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 where the water is marked in a dark color (i.e. brightness below 0.5) and the land in bright color. The obstacle attribute layer can then be selected in the automatic street generation wizard and streets are generated accordingly (i.e. no streets in dark regions).

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

Obstacle Layer options

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

Obstacle file	Choose an image from your workspace which will be used to as the obstacle in your scene.
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.
Obstacle Threshold	The **Obstacle Threshold** defines the image brightness, which distinguishes between obstacle and non obstacle.For example, if you set the **Obstacle Threshold** to 0.5 in the `brightness` channel, any cell value below 0.5 is considered dark and above is bright and part of the obstacle.
Bounds Dimensions	The **Width** and **Height** of the resulting texture in meters. When a georeferenced image is selected, this value is set automatically.
Bounds Location	The location of the resulting texture in meters. When a georeferenced image is selected, this value is set automatically. **Note:** The button right of **Location** can be used to change the reference point of the obstacle's position.

Obstacle layer in a CityEngine scene

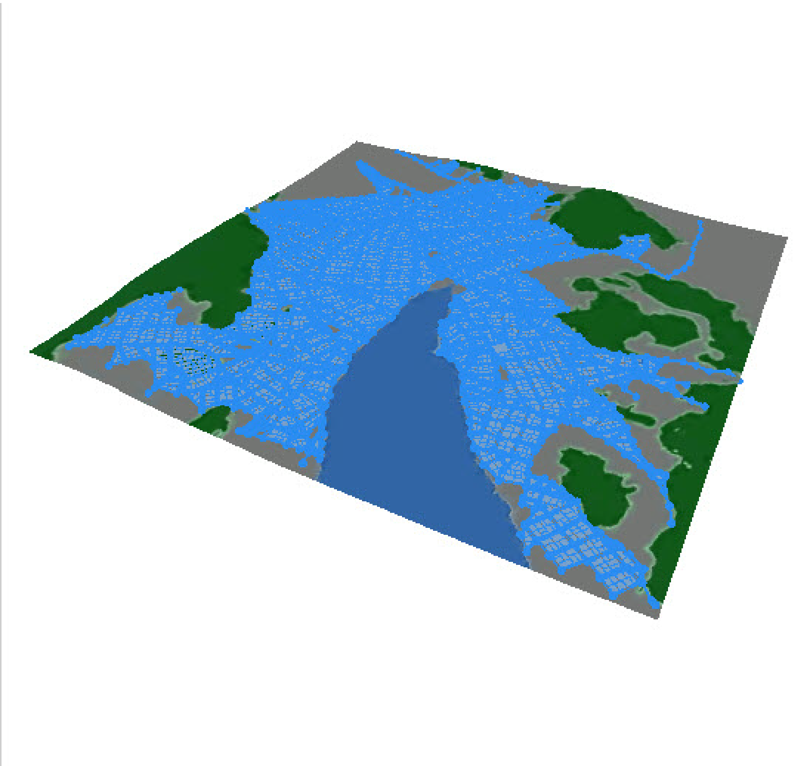
After the obstacle type has been selected in the wizard the obstacle attribute layer can be created as follows:

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 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**.

As a result 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.



Street network generation 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, where you can create mappings from an image file to actual 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 source for the rule attribute in the Inspector, see Mapping image data with rule attributes.

Create a mapping layer

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. Create a new attribute by clicking on the top-left "insert-row" icon (left to **Attribute**).

    Note:
    An arbitrary number of mappings can be defined per layer.

6. Right-click in the attribute area and select **Add Row** to create a new 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 lets you 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.

Create a function layer

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 opens, click **Function**.

After the function type has been selected in the wizard, the function attribute layer can be created in the function dialog by entering a function which 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 Editing map layer attributes.

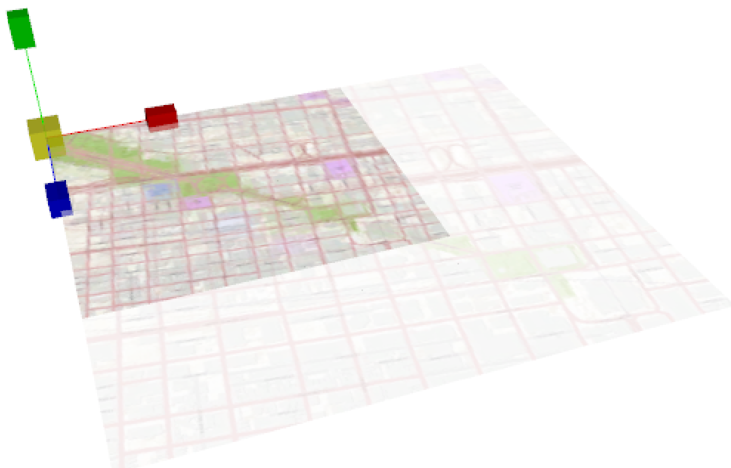
Edit map layer

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.

Furthermore, the mapping functions can also be edited in the Inspector, see Edit map layer functions.

Move and scale a map layer in the Viewport

If you select one or more attribute layers in the Viewport, you may use the transform or scale tool in order to move or scale the layer(s).



Scaling a map layer

Editing 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 very similar to CGA shape grammar editing, but only a subset of functions are available for attribute layers and no rules or shape operations. Use the command completion Ctrl`+ `Space` 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

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

Layer attribute code in detail

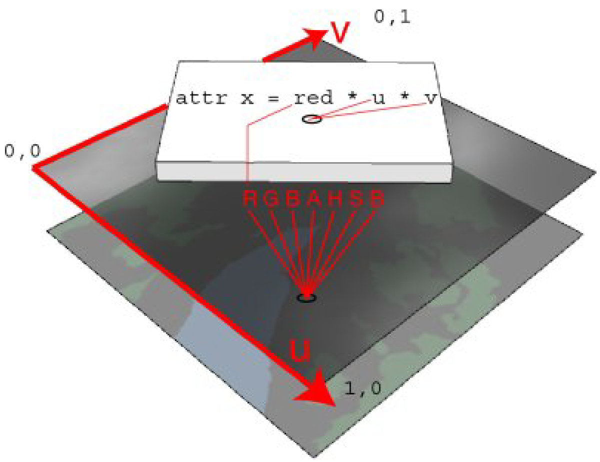
Map layers in general define one or more attributes as a function of the location and optionally a mapping channel. The dimension of the map are normalized to the interval [0..1]. Thus the lower left corner of your map has the coordinates (0, 0) and the upper right corner of the map has the coordinates (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 to that, inside an attribute function, "red", "green", "blue", "alpha", "hue", "saturation", "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 to that, the map is sampled at the position "u,v" and its red channel is used for the calculation of "x".

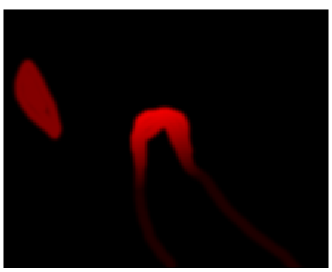


attr x at center of centroid

Mapping image data to rule attributes

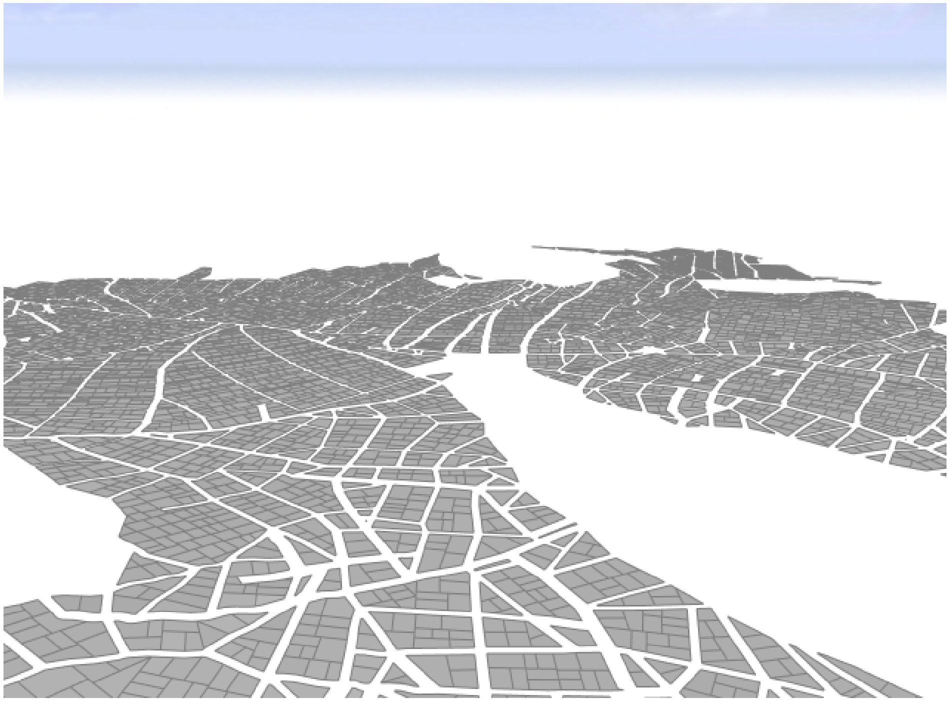
Map layers are a very powerful tool to control CGA shape grammar rules. Any attribute that you have defined in your CGA shape grammar rules can be mapped from an attribute layer. This allows you to guide your rules by maps. Typically, maps are used for controlling building attributes such as height or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

The image below is going to be used as 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 like in the image below:



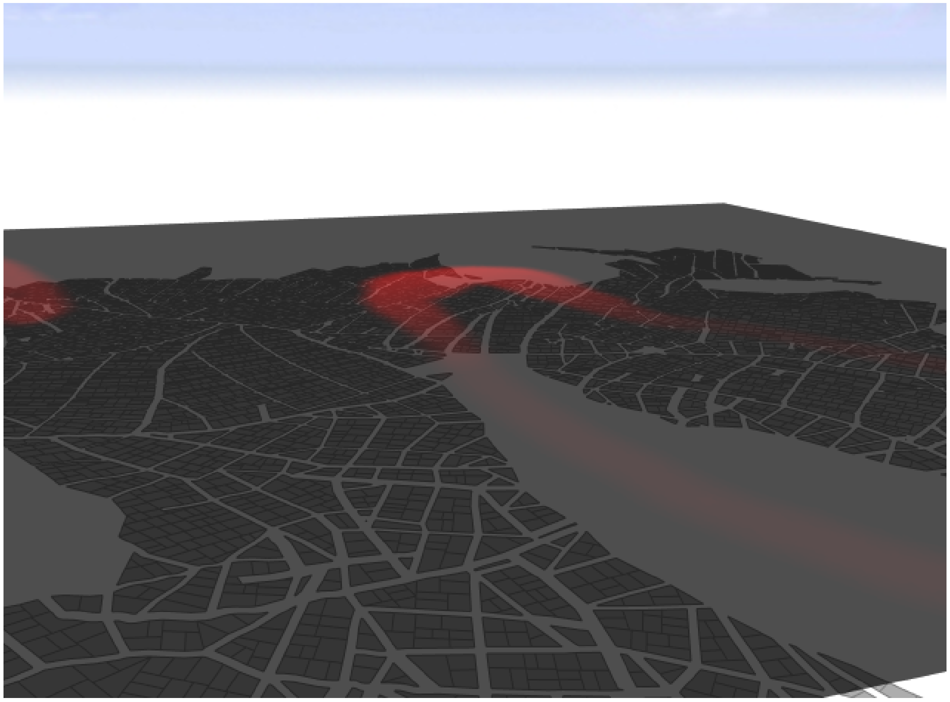
Scene with building footprints

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 new 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.

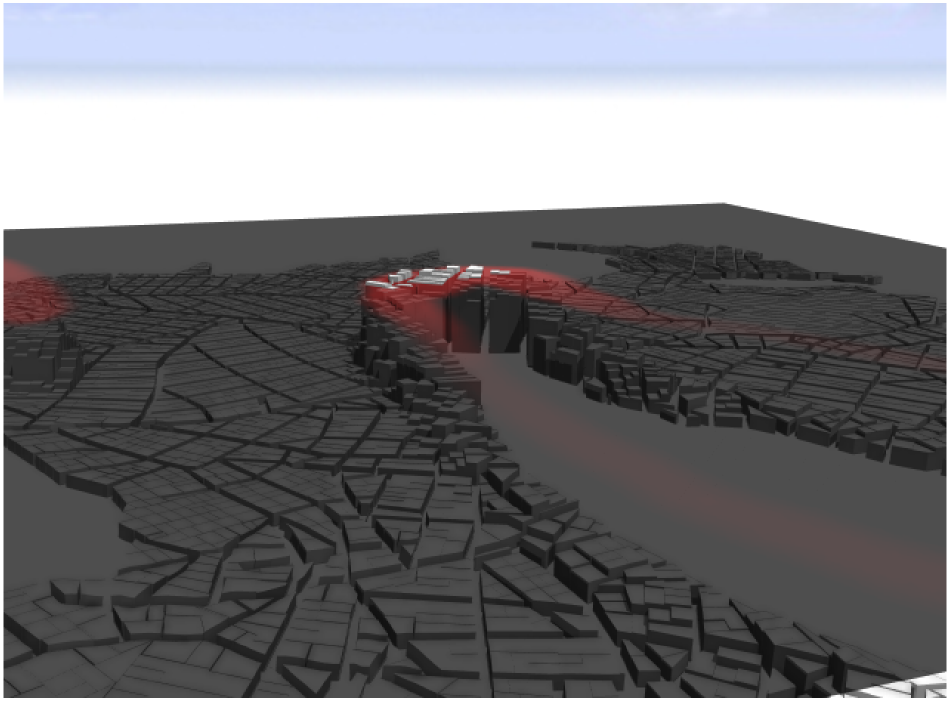


Building heights before being defined by mapping layer range

Connection Editor

To connect attributes, do the following:

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


Building heights defined by mapping layer

Selection with image maps

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

Using the terrain for selection

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

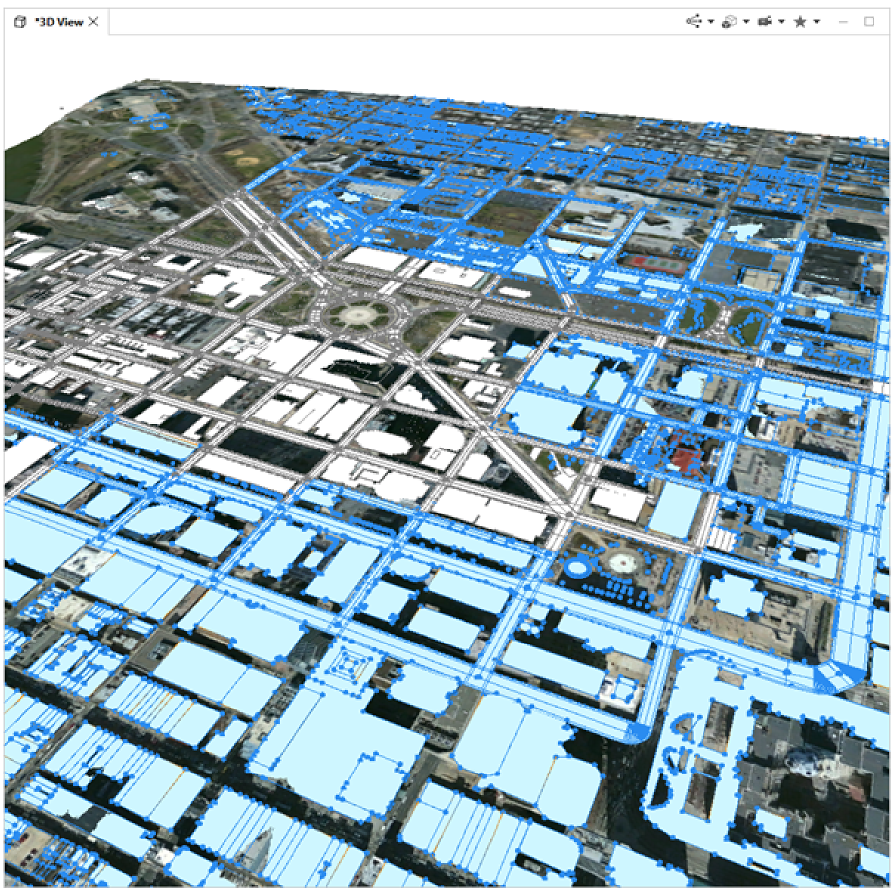
```
attr elevation = map_01(brightness, 100, -100)
```

You want to select all elements that have an elevation of 10 meters or higher. Simply add a new attribute that evaluates to true when elevation is bigger 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:

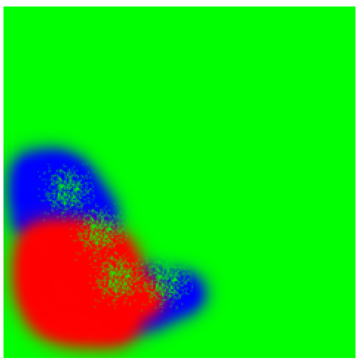


Selected objects in Map layer

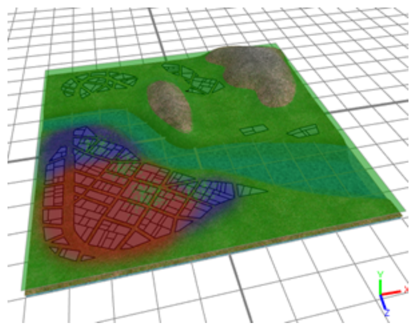
Note:
By selecting a layer before selecting objects by map layer you can select only objects from the selected layer. If you have a layer selected that does not contain any objects this leads to nothing being selected.

Use a landuse map for selection

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



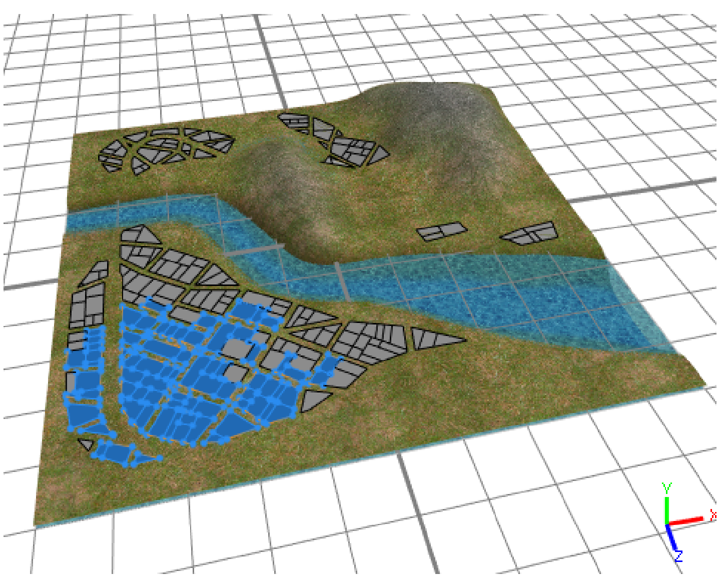
Landuse map in red, blue and green



Landuse map placed in the scene

After adding a new map layer with the landuse map, three new boolean attributes are defined in the Inspector view of the new layer. Depending on the color of the map, landuse types are evaluated.

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



Landuse types defined by Map layer

Selection using the u,v coordinates of a map layer

For the following landuse attribute definition,

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

just 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 overview

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

Shapes

Shapes 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 Push Pull tool to extrude shapes and Guides to perform 3D shape editing with precision. In this section you will also learn how to align shapes and terrains as well as manual shape texturing.

Graphs

A graph is a structure consisting of edges and nodes. In CityEngine graphs are mainly used to layout street networks. When drawing and editing edges and nodes, dynamic shapes for the drive and pedestrian lane are created automatically.

Draw shapes

Polygons, rectangles, circles

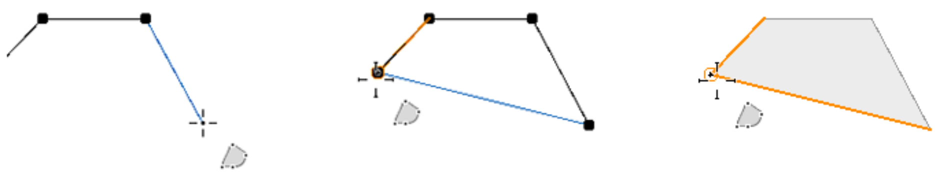
CityEngine offers tools to draw polygons, rectangles, and circles.

- 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`) or click **Shapes** > **Polygonal Shape Creation** in the main menu.

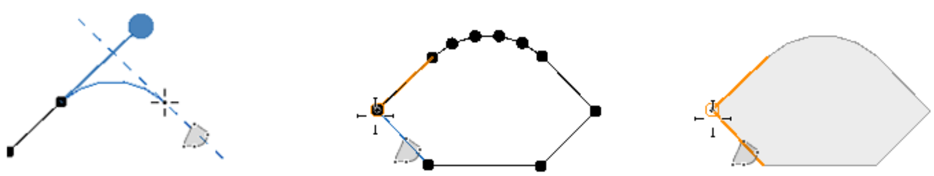
Click to create the first vertex and click again to place additional vertices. To finish the shape, click the first vertex. You can also double-click or press `Enter` to finish the shape. Double-click an edge to add a vertex at that place. See Polygonal Shape Creation Tool Options for shape creation options.



Polygonal Shape Creation tool

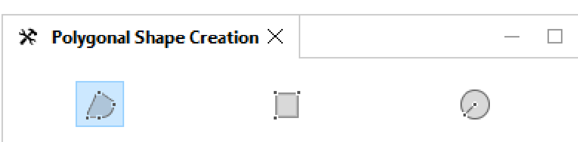
Draw an arc

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 includes the following options:

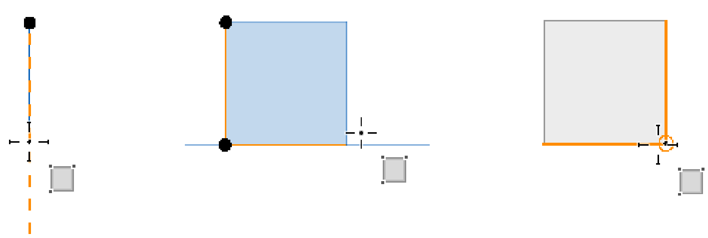
Polygonal Shape Creation	
Snapping	 Turn snapping options on and off in the scene. See Snapping.

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 `Enter` to apply the value.
Arc Mode	Change the edge into an arc. • Enter a value for the resolution of the arc. The smaller the value, the more sides the arc has. • Change the value by using the scroll wheel or the up and down arrow keys. • Press (`a`) to switch between an edge and arc.
Force New Shape	Force a new shape to be created, including when drawing intersecting shapes.
Force Planar	Force the polygon to be planar. Press (`t`) to turn force planar mode on and off. See Force Planar.
Automatic Closing	Create a shape that closes along the edges of connected shapes. See Automatic closing.

Draw a rectangle

To draw a rectangle, click the **Polygonal Shape Creation** tool (`S`) and then 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 Rectangular Shape Creation Tool Options for shape creation options.



Rectangular Shape Creation tool

Rectangular Shape Creation Tool Options

The **Rectangular Shape Creation** tool includes the following options:

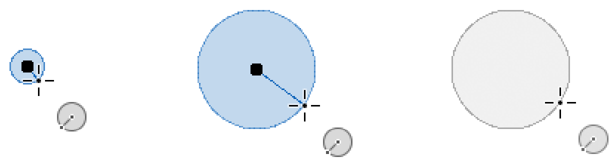
Rectangular Shape Creation	
Snapping	Turn snapping options on and off in the scene. See Snapping.

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 `Enter` to apply the value.
Force New Shape	Force a new shape to be created, including when drawing intersecting shapes.
Force Planar	Force the rectangle to be planar. Press (`t`) to turn force planar mode on and off. See Force Planar.

Draw a circle

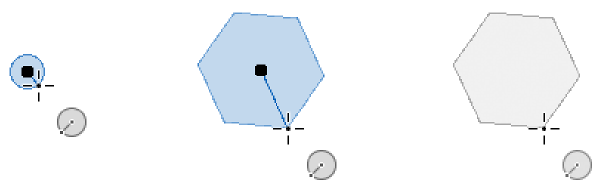
To draw a circle, click the **Polygonal Shape Creation** tool (`S`) and click the **Circular Shape Creation** tool (`Shift+C`) in the **Tool Options** window. You can click **Shapes** > **Circular Shape Creation** in the main menu.

Click once to set the center of the circle, move the pointer to define the radius, and click again to finish. See Circular Shape Creation Tool Options for shape creation options.



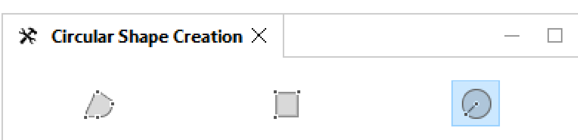
Circular Shape Creation tool

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 Snapping.
Radius (m)	The radius length in meters. You can enter a value before or while drawing to lock the radius length. Press `Enter` to apply the value.
Segments	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.

Force New Shape	Force a new shape to be created, including when drawing intersecting shapes.
Force Planar	Force the circle to be planar. Press (`t`) to turn force planar mode on and off. See Force Planar.

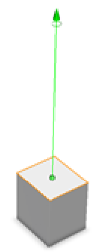
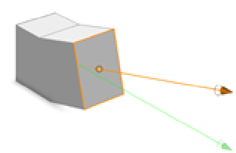
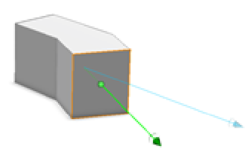
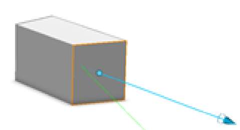
Push Pull Tool

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

Extrude face

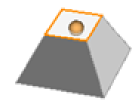
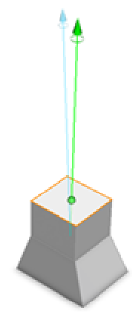
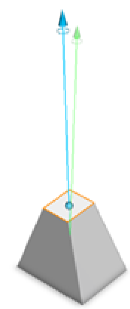
Depending on the shape, different arrows appear. This allows you to create shapes along different directions. Hover over a direction arrow for immediate feedback. There are four types of directions, each with a unique color and mouse icon.

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.

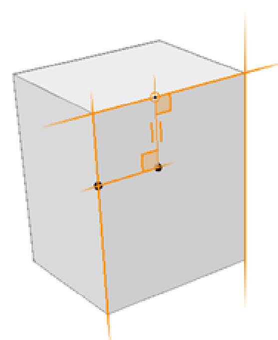
Special edge direction

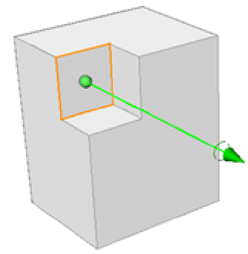
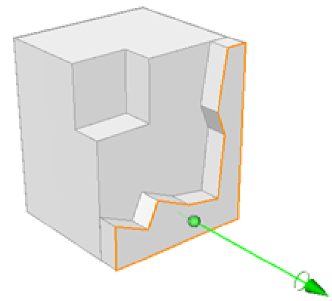
When dragging along the special edge direction arrow, all edges are extended along their adjacent faces during dragging. The arrow appears with a slight offset when it has the same direction as another arrow. The table below shows the difference between the up and special edge direction dragging.

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

3D shape editing

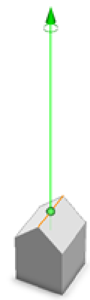
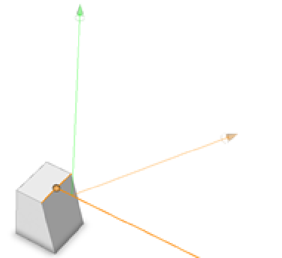
3D shapes allow polygonal editing, such as snapping, splitting, and automatic closing. Split parts can be moved to further refine the 3D model.

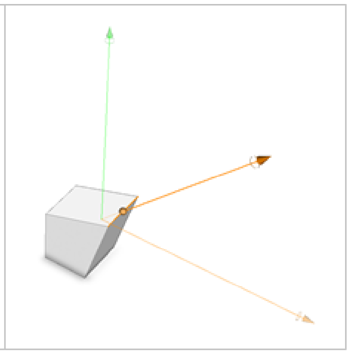
Preliminary split.	

3D move.	
Final split and 3D move.	

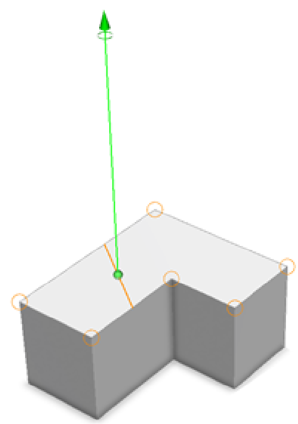
3D edge move

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

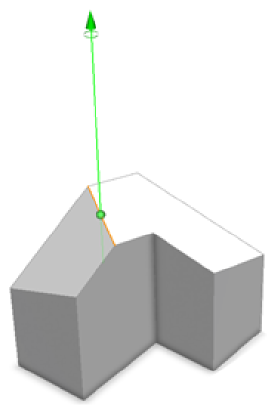
Edge move along global axis.	
Edge move along adjacent faces.	

Edge move along average face normal.	

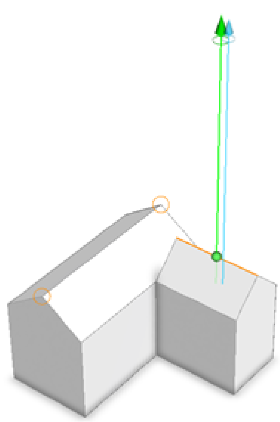
While moving the edges with the **Push Pull Tool**, connected faces update to maintain planarity. In addition, the moved edge is intersected with neighboring polygons. Both features are useful for creating roofs as shown below.



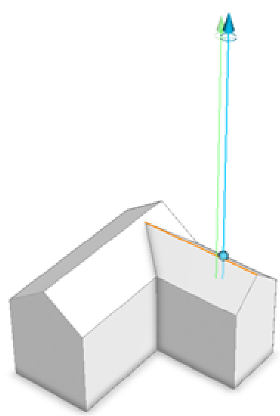
Edge before moving orthogonal faces



Orthogonal faces moving with edge



Inset edge along roof



Edge move along the blue axis insets the edge along the existing roof

Tool options

The **Push Pull Tool** tool includes the following options:

Push Pull Tool	
Distance (m)	Distance in meters of the edge or face extrusion. • You can enter a value to lock the extrusion distance along the available directions. • Click again to start a new extrusion.
Snapping	Turn snapping options on and off. • Extrude a face and snap to faces, edges, vertices, of the same shape. Also, snap to guides. • Extrude an edge and snap to vertices of the same shape. • Press `Shift` to temporarily enable or disable snapping.
Force New Edges	• Force the creation of a new edge when extruding. • Press `Ctrl` to temporarily turn on or off the creation of new edges.

Guides

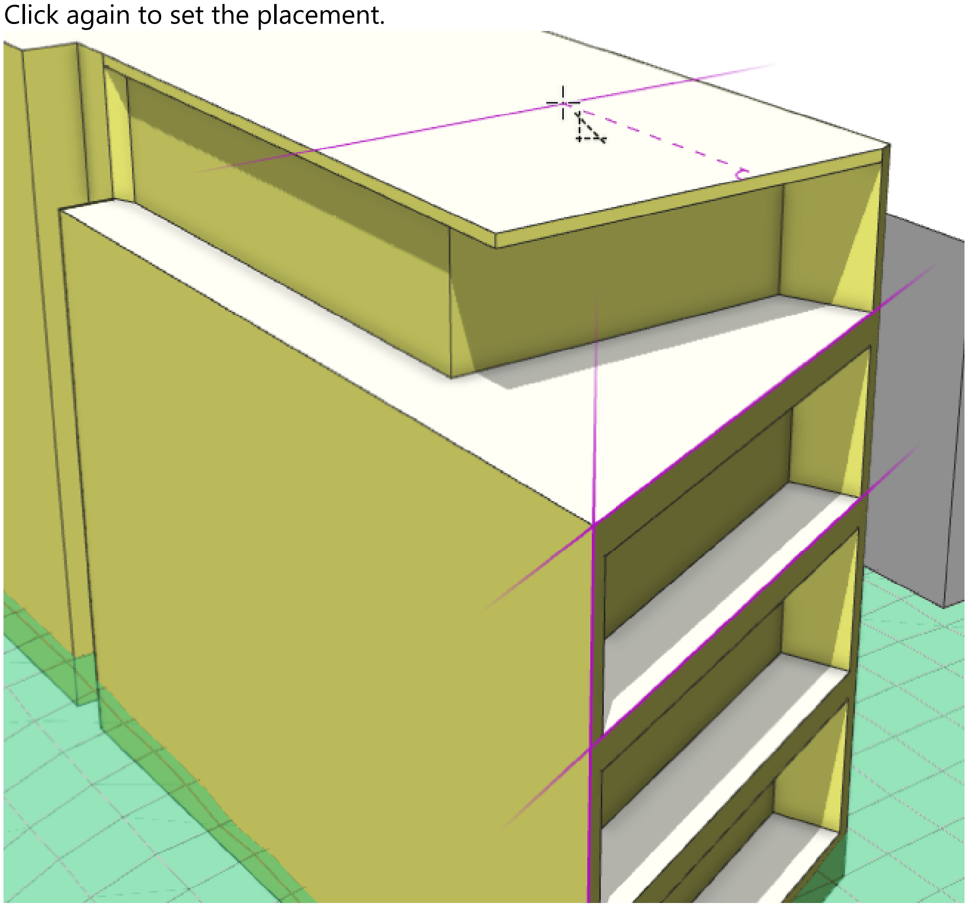
You can create lines as guides in 3D that help you position and align with other existing objects as you draw.

To use the guide tool, do one of the following:
- Click the **Guide Creation** tool.
- 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.



A guide is created with an offset on the roof of a building

The guide is an additional snap target that allows parallel as well as extension snapping. You can have up to 20 guides at a time. If you add more, the first guide is removed. Guides are saved in the scene.

You can hover over an existing guide to display the remove guide icon and click the guide to delete. Click the **Delete All Guides** button in the **Guide Creation Tool** tool options to remove all guides.

Tool options

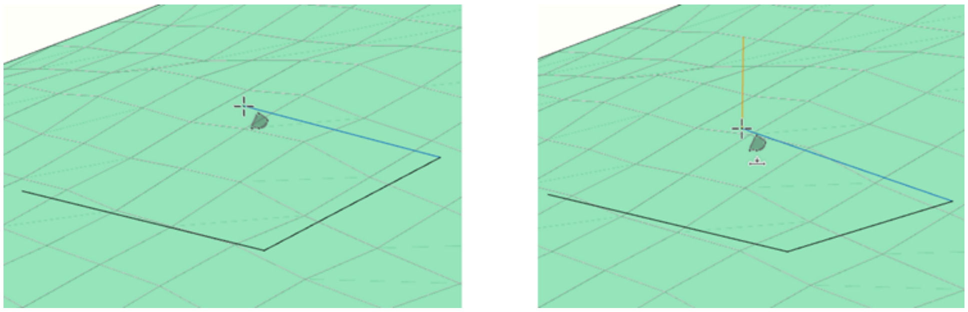
The **Tool Options** window includes the following settings:

Guide Creation Tool	
Offset (m)	The offset distance in meters from the edge that is clicked. • You can enter a value to lock the offset. • Preview the offset guide placement until the next click sets the guide.
Force Planar	Force the guide to be planar. Press (`t`) to turn force planar mode on and off. See Force Planar.
Delete All Guides	Click to delete all guides.

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 Polygons, rectangles, 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 cursor 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 cursor changes and orange highlights on edges and vertices are displayed.

Snapping options

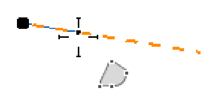
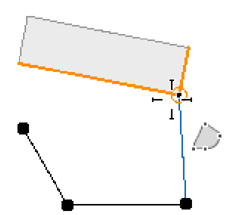
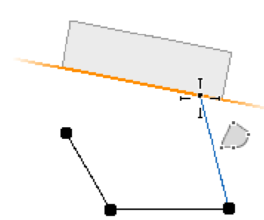
You can turn snapping on and off in the drawing tools by clicking the following types of snapping elements:

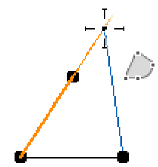
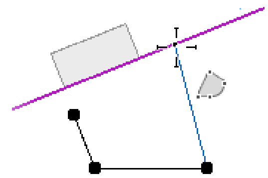
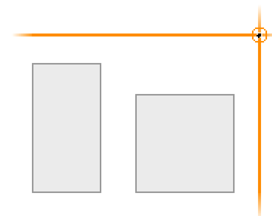
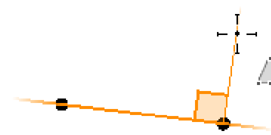
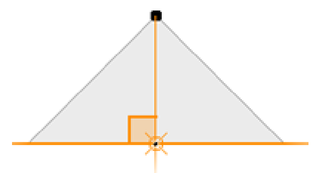
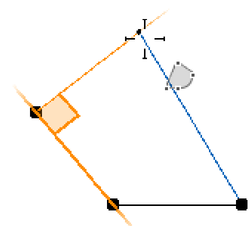
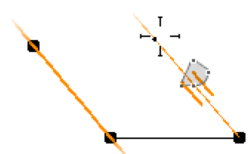
Locked objects	Snap to locked objects in the Scene Editor.
Shapes	Snap to shapes such as footprints, 3D models, and streets.
Global Axes	Snap to global axes.
Guides	Snap to guides.
Terrain	Snap to the terrain. See also Force Planar.

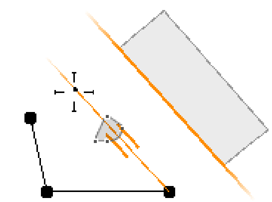
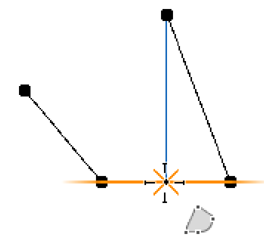
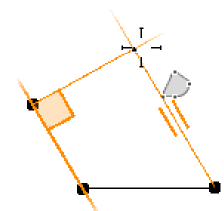
Note:
Press `Shift` while drawing to temporarily disable the selected snapping elements.

Snapping features

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

Global axes Snap to the global axes.	
Vertex Snap to shape or street vertices.	
Edge Snap to shape or street edges.	

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

Parallel guides Current segment is parallel to guides.	
Midpoint Snap to the midpoint of an edge.	
Combination Snap to the intersection of combined snapping features.	

Tip:

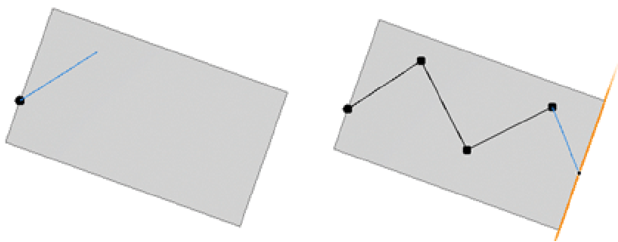
- To finish the shape when drawing, double-click or press `Enter`.
- Snapping is also available with the Transform Move tool.

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

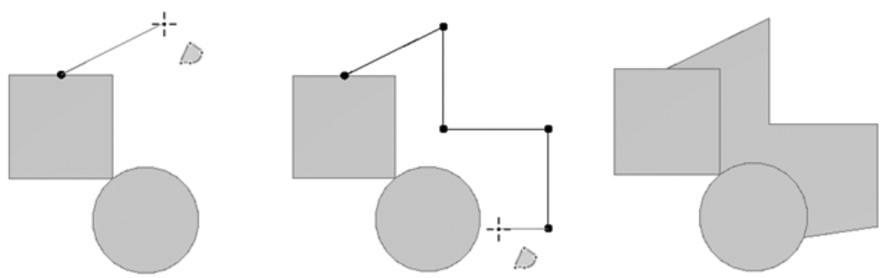
You can split shapes using the **Polygonal Shape Creation** tool (S). When the first and last vertex are snapped to an edge or vertex of the same shape, and the whole line is contained in this face, it is automatically split into two parts.*



Splitting polygonal shape

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 **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.*

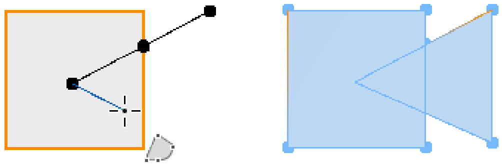


The polygon is automatically closed along the indirect connection.

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

Combine shapes

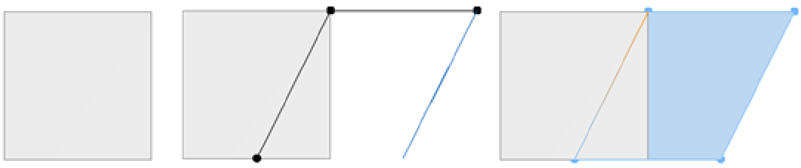
To combine shapes into one shape, click any of the shape creation tools and draw a shape that snaps to an existing shape. Ensure that **Shapes** option is enabled under **Snapping** in the shape creation tool options.*



Combining shapes

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

Note:
*This option is unavailable when **Force New Shape** is enabled.

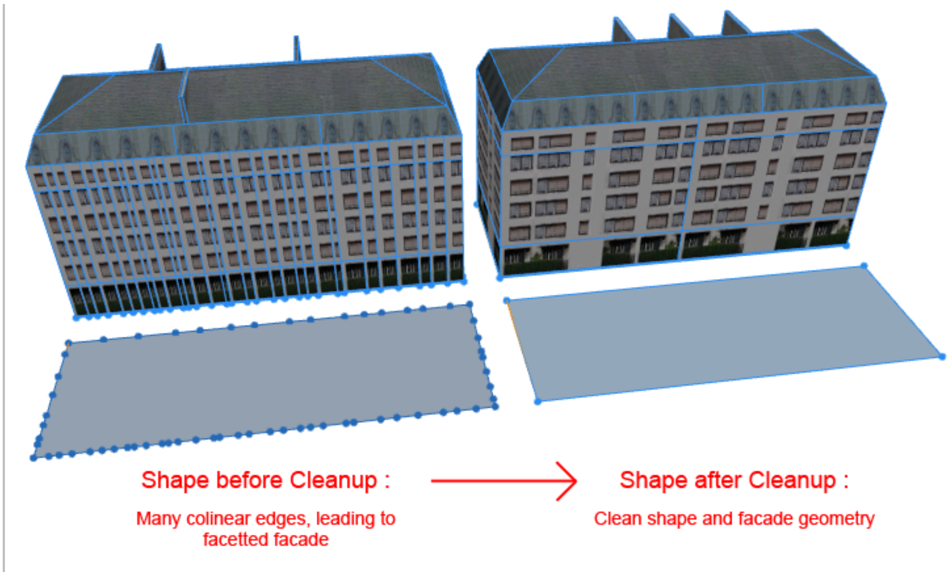
Cleanup shapes

Use the **Cleanup shapes** tool simplifies and cleans the geometry for polygons (faces), vertices and edges. It is very 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 parameters

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

Merge Vertices	If the distance between two vertices is lower than the threshold, they are combined into one.
Remove Coplanar Edges	Merge connected coplanar polygons.
No Cleanup on Discontinuous Textures	When the texture coordinates are not continuous for a vertex, all operations are skipped for this vertex.
Remove Collinear Vertices	Multiple vertices on one straight line are removed.
Remove Double Faces	Faces that have identical vertices (up to shift and inversion) are removed but one is kept.
Remove Zero Faces	Faces with zero size are removed.
Intersect Edges	All edges are intersected, and new vertices are inserted for every intersection point.
Split Coplanar Polygons	Overlapping polygons on the same plane are split into non-overlapping polygons. This requires the "Intersect Edges" operation.
Conform normals	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 in order to be successful.
Distance Tolerance, Angle Tolerance	Thresholds for the above operations.

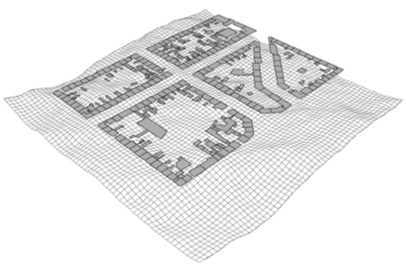


Example of the Cleanup Tool in action, removing collinear vertices

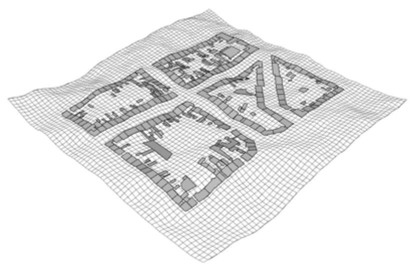
Align shapes to terrain

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



Shapes not aligned to terrain



Shapes aligned to a terrain

Tool settings

The following parameters control the alignment:

Align function	The alignment function to apply to the vertices of the shape polygons. • **Project All**: Projects all shape vertices onto the terrain. • **Project Below**: Projects the vertices located below the terrain only. • **Project to Object Average**: Projects the shape vertices to the average of the vertices. • **Translate to Average**: Translates the shape to the average elevation of the projected vertices. • **Translate to Maximum**: Translates the shape to the maximum elevation of the projected vertices. • **Translate to Minimum**: Translates the shape to the minimal elevation of the projected shape vertices on the terrain.
Terrain	The terrain to align the shapes. All attribute layers with an "elevation" attribute plus the y=0 level are listed here.
Offset	The offset to add after alignment to the y-coordinate of the shape points.

Texture shapes

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

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

Note:
The tool dialogs are usually modal, i.e. 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 a simple and interactive means for texture assignment. The image the user wants to use as a texture may be chosen as follows:

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

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

Orientation Alignment	Select an orientation for the following parameters, either **global** or **local**.
Rotation	Correct the image orientation by the following three options in 'Image Transformation' group: • **Rotation**: None | 90° counter clock wise | 180° | 90° clock wise | Arbitrary • **Flip horizontally** • **Flip vertically**
Arbitrary Rotation	Specify an arbitrary rotation angle in degrees, if **Rotation** above is set to "Arbitrary".

Texture Coordinates Mapping Mode	(see below for examples)
	There are four modes with according attributes for the texture coordinate mapping, which are specified in the 'Texture Coordinates Mapping' group:
	• **Keep current mapping**
	Leaves the currently set mapping (no attributes).
	• **Stretch to polygon**
	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.
	  ◦ **Align to**: Bottom left corner | Bottom right corner | Top left corner | Top right corner (has only an effect if one of the repetition numbers is not a whole number).
	  ◦ **Horizontal repetitions**: number of times the texture is repeated horizontally (decimal number allowed).
	  ◦ **Vertical repetitions**: number of times the texture is repeated horizontally (decimal number allowed).
	• **Dimension**
	This mode allows the user to set the actual dimension of the used texture.
	  ◦ **Align to**: Bottom left corner | Bottom right corner | Top left corner | Top right corner
	  ◦ **Absolute texture width**: given in meters.
	  ◦ **Snap horizontally to bounds**: stretches the given texture's width such that it horizontally fits the selected face with a whole number of repetitions.
	  ◦ **Absolute texture height**: given in meters.
	  ◦ **Snap vertically to bounds**: stretches the given texture's height such that it vertically fits the selected face with a whole number of repetitions.

Scene selection issues

The **Texture shapes** tool acts a little 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: using 1.8 horizontal repetitions, 1.3 vertical repetitions with Align to Top left corner

Dimension: 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 offers 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 Street Parameters for more information.

Draw a polygonal street

To draw a polygonal street, click the **Polygonal Street Creation** tool (`G`) or 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 window includes the following options:

Polygonal Street Creation	
Segment Length (m)	The length of the current street segment in meters. • Enter a value before or while drawing to lock the length. Press `Enter` to apply the value. • Lock and unlock the length by clicking either the lock or the unlock buttons.
Re-use settings from neighbors	If enabled, settings are copied from neighbor streets, if available. If an existing street is extended, settings are typically copied from that segment.
Street width	The street width.
Left sidewalk width	The left sidewalk width.
Right sidewalk width	The right sidewalk width.
Rule file	The rule file is assigned to all new street shapes.
Apply rule-based model generation	If enabled, model generation is automatically triggered.
Align terrain	Aligns the street with the terrain.

Draw a street freehand

To draw a street freehand, click the **Polygonal Street Creation** tool (`S`) and then 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 window includes the following options:

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

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.

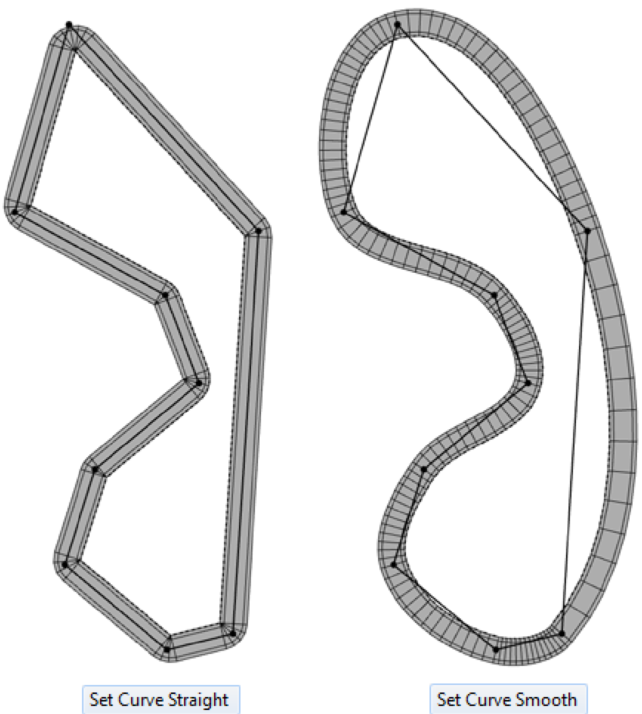
Edit street and curves

The **Edit street/curves** tool offers intuitive 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 vs. smooth

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



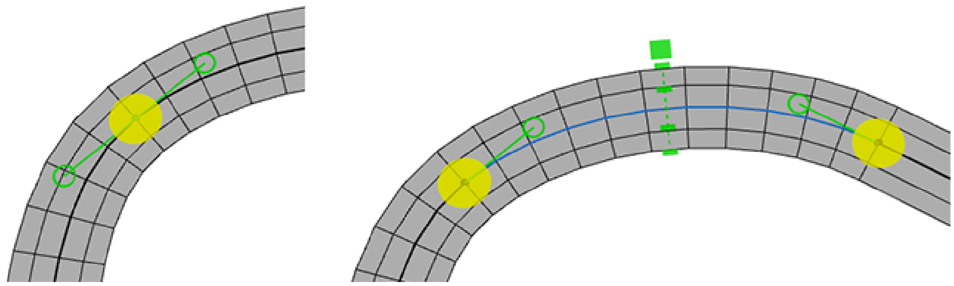
Curve Straight and Curve Smooth tools

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, in order to prevent oscillations.

Street editing

Once you activate the **Edit street/curves** tool 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.



Left: Curve handle. Right: Curve and street width handle

Tip:

In order to lock the direction, press the `Shift` key before dragging a curve handle.

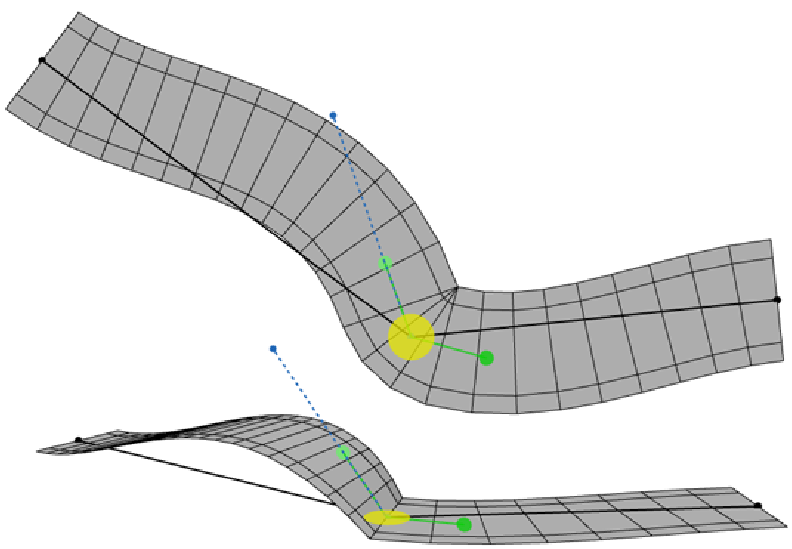
Curve handles

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 the mouse hovers over one of the curve handles) indicates the local weight of the curve.

Note:

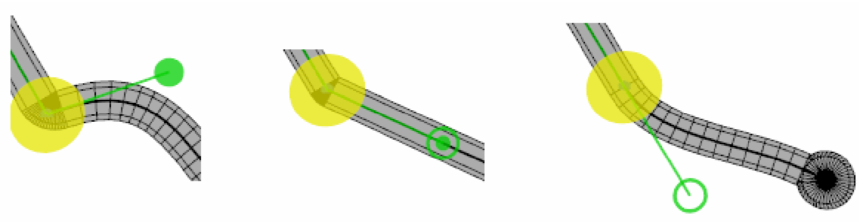
The spline is mathematically defined by the two end points of the dashed blue lines, plus the segment's start and end points.

To edit either the horizontal or vertical components of a curve handle, the user just needs to change the viewing angle relative to the segment node. A steep angle lets the user edit the horizontal component, a glancing viewing angle the vertical component.



Curve handles

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

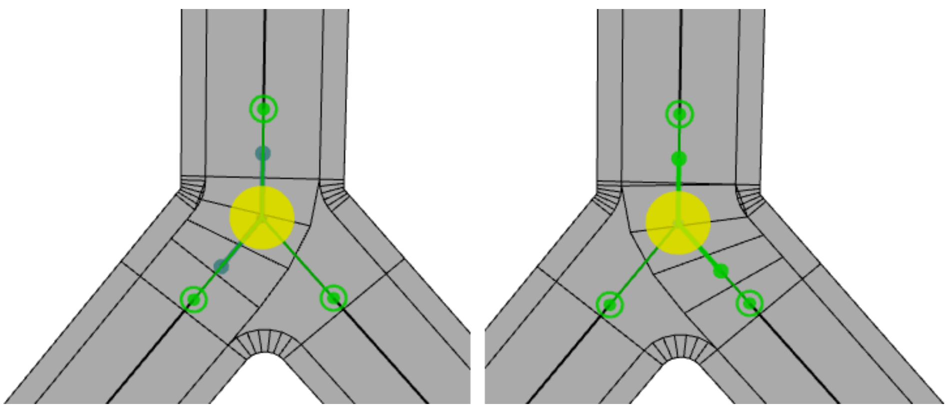


Indication of node type in green circle: (left) smooth with manual directions (middle) straight (right) smooth with automatic directions

Note:
The segmentation of the street shapes defines the segmentation of the neighboring lot shapes. Thus, the user should keep an eye on the overall number of segments per curve to avoid issues such as e.g. 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.



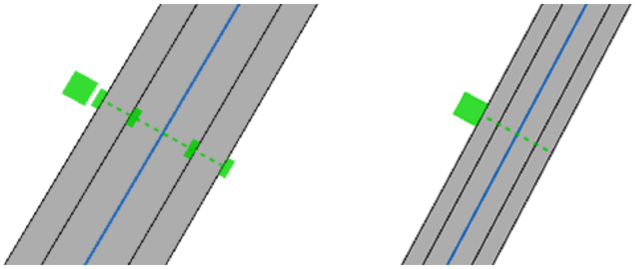
Dark green principle street handles (above, left) indicate that the default principle street will be chosen. Once a principle street is specified manually, the handles are coloured light green (right).

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 with clear the principle street at the intersection, resulting in the default behavior.

Street width handles

Editing the different dimensions of a street can be done with the `StreetWidth` handle. The width and offset of the street shape plus the widths of the right and left sidewalks can be driven individually by dragging the small green rectangles. Symmetrical adjustments are performed by pressing the `Shift` key while dragging. The big green rectangle is the combined width handle. Dragging it changes street width and sidewalk width simultaneously.



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

Note:
Street width parameters can be set numerically as well in the street parameters.

Cleanup streets

Imported, merged or self-drawn graph networks may contain the following:

- duplicate or close-by nodes
- duplicate or close-by segments
- intersecting segments that do not have nodes where segments intersect

Such unclean graph networks induce a number of problems when creating street shapes or extracting lots.

Note:
The **Cleanup streets** tool allows for fast cleanup of such graph networks by merging nodes, merging segments and creating nodes at intersecting segments.

The tool can be executed in the in the following ways:

- Click the **Cleanup streets** tool .
- Click **Graph** > **Cleanup Graph...** in 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 settings

The checked operations (intersect, snap, merge and/or resolve shape conflicts) are executed one after another. The following parameters can be set:

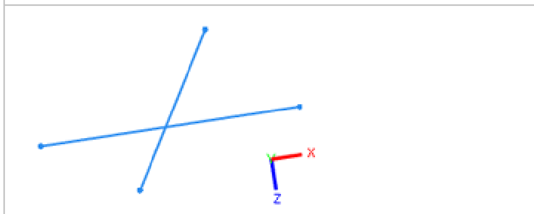
Intersect Segments	If checked, missing nodes of intersecting segments are created.
Snap Nodes to Segments	When checked, nodes snap to segments. **Note:** Nodes with smaller street widths always snap into segments with larger street widths. Node street width is defined as the maximal street width of the adjacent segments.
Horizontal Snapping Distance	The maximal horizontal distance between a node and a target segment. Only meaningful if the option above is checked.
Vertical Snapping Distance	The maximal vertical distance between a node and a target segment. Only meaningful if the option above is checked.
Merge Nodes	When checked, nodes that are close to each other are merged.
Horizontal Merge Distance	Nodes that are closer than this distance in a horizontal direction are merged into one. Only meaningful if the option above is checked.
Vertical Merge Distance	Nodes that are closer than this distance in a vertical direction are merged into one. Only meaningful if the option above is checked.

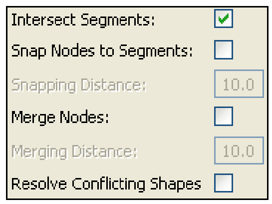
Resolve Conflicting Shapes	When checked, the tool collapses all street segments which cause street shape conflicts. This is executed iteratively until no more conflicts exist. **Note:** Segments with the smallest minimal adjacent node valence are collapsed first, i.e. this 'segment valence' determines the order of the segment collapse iteration.

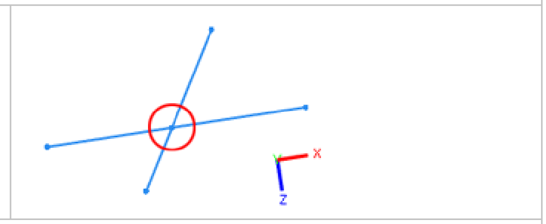
Note:
This tool operates planar in the x-z plane. The y-coordinate is neglected. Therefore, running this tool on graph networks containing segments on different y-levels is not recommended.

Examples

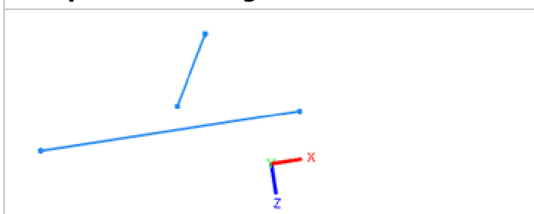
Simple Intersections

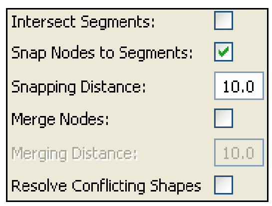


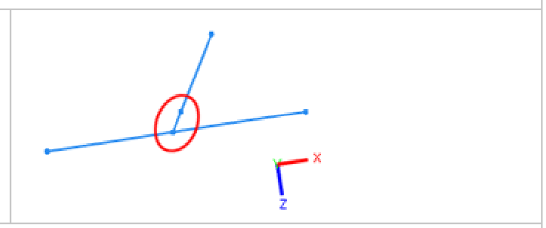




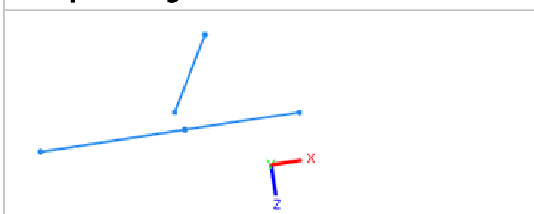
Snap Nodes to Segments

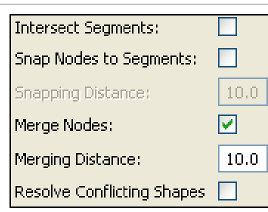


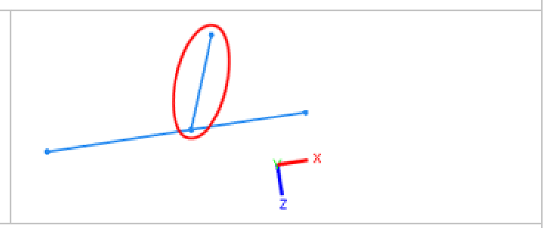




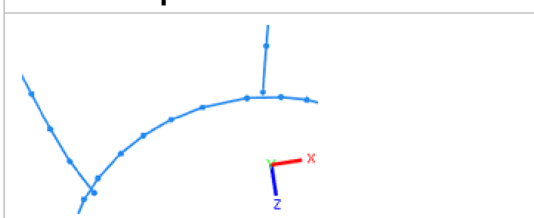
Simple Merge

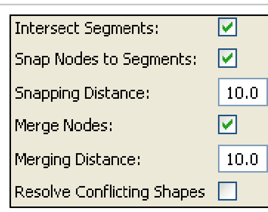


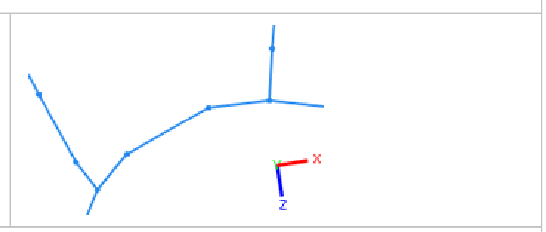




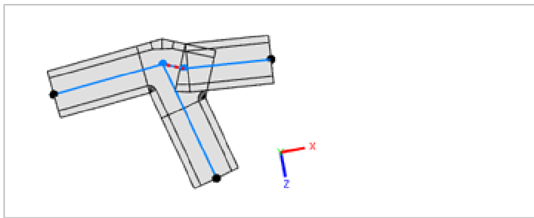
Full Cleanup

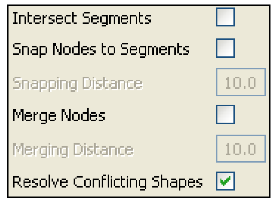


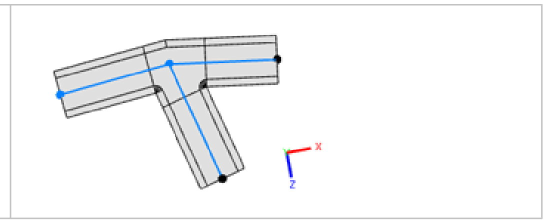




Resolve Conflicting Shapes



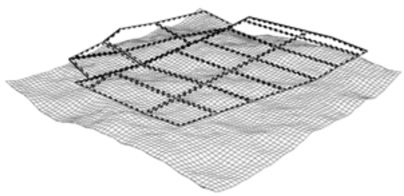

Intersect Segments
Snap Nodes to Segments
Snapping Distance 10.0
Merge Nodes
Merging Distance 10.0
Resolve Conflicting Shapes



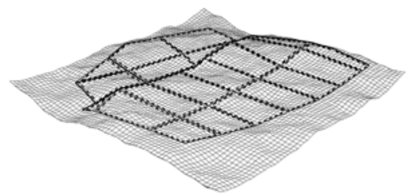
Align streets to terrain

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

- Click the **Align streets to terrain** tool in 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:

Align function	The alignment function to apply to the nodes of the graph. • **Project All**: Projects all nodes onto the terrain • **Project Below**: Projects the nodes located below the terrain only.
Terrain	The terrain to align the graph. All map layers with an "elevation" attribute plus the y=0 level are listed here.
Offset	The offset to add after alignment to the y-coordinate of the nodes.

Generation of street networks

The **Grow streets** tool can be used to generate typical street networks. Three different street patterns (organic, raster and radial) can be arbitrarily combined. The dialog with a number of settings allows the user to generate street networks according to its needs.

The tool can be used in the following different ways:

- Create a new 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** the main menu.

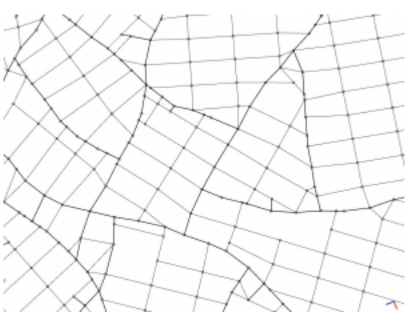
Note:
The algorithm distinguishes between major and minor streets. Basically, major streets are created until they enclose an area, called a quarter. Then the quarter is subdivided by minor streets. Afterwards the algorithm continues creating major streets and so on.

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).

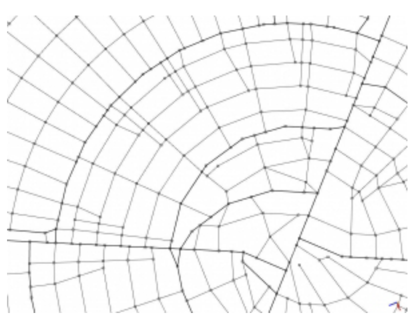
- **Basic Settings** consist of the number of streets to generate and the street patterns.
- **Pattern Specific Settings** define the street patterns more precisely.
- **Advanced Settings** specify the algorithm behavior and the algorithm constraints.
- **Environment Settings** include obstaclemaps to restrict the growth area and terrains to adapt the created streets to the elevation.
- **Street Width Settings** define the street widths 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

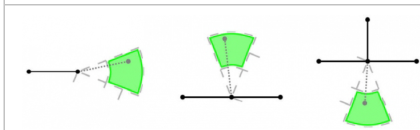
Street patterns need two street lengths: The long and the short length. The organic pattern needs just one length (the short length is used).

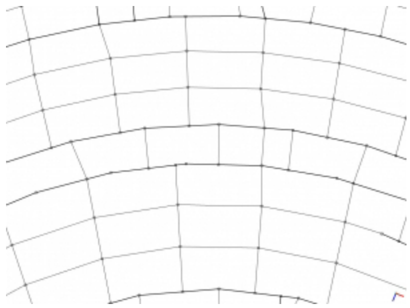
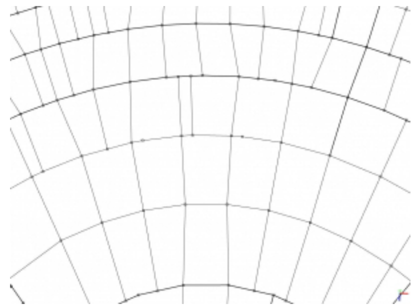
Basic setting has the following parameters:

Number of Streets	The number of streets to generate in total.
Pattern of Major Streets	The street pattern used for major streets: Organic, raster or radial.
Pattern of Minor Streets	The street pattern used for minor streets: Organic, raster or radial.
Long Length	The average length of the long streets (used for the raster and radial pattern).
Long Length Deviation	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.
Short Length	The average length of the short streets (used for all patterns). See Short Length Deviation.
Short Length Deviation	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.

Pattern specific settings

The pattern specific settings specify the street patterns in more detail.

Max. Bend Angle (Organic)	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).
	
City Center (Radial)	The city center used for the radial pattern. Streets go radial or centripetal around or outside of the center.
Max. Bend Angle (Radial)	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.
Street Alignment (Radial)	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.

Advanced settings

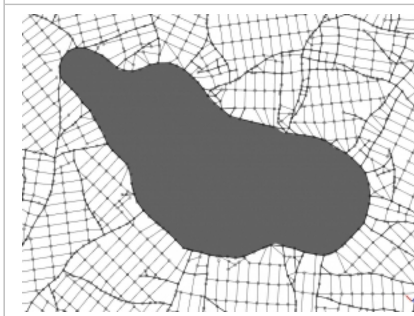
The advanced settings specify the algorithm behavior and the algorithm constraints.

Snapping Distance	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, one can control the minimal distance between any two nodes of the street network. Note that only the half of this distance is applied if a minor street intersects with a major street (in order to model more realistic quarter subdivision).
Minimal Angle	The minimal angle between any two neighbor streets of the street network. It is guaranteed that no smaller angle originates.
Street to Crossing Ratio	Using the street to crossing ratio, one 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. Street to Crossing Ratio = #Major street nodes / #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.
Development Center Preference	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.
Angle Offset of Major Streets	Before major street creation, this offset angle is added to the proposed street angle.
Angle Offset of Minor Streets	Before minor street creation, this offset angle is added to the proposed street angle.

Environment settings

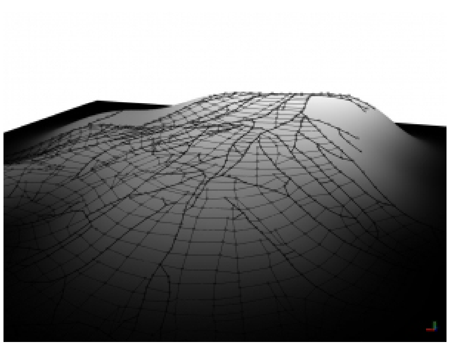
Using environment maps one can define boundary conditions like terrains or obstaclemaps.

Adapt to Elevation	Enables or disables adaption to elevation.
Critical Slope	Only proposed streets with a slope greater than the critical slope are adapted.

Maximal Slope	The maximal allowed street slope.
Adaption Angle	The maximal angle a proposed street is adapted (rotation around the y-axis).
Heightmap	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 which defines the float attribute `elevation`.
Obstaclemap	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 which defines the boolean attribute `obstacle`. 

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, i.e. 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 downwards.

Street width settings

Street and sidewalk widths are assigned to the new streets. If an existing street is extended, their street and sidewalk widths are copied to the new street. Otherwise, street and sidewalk widths are randomly set according to the following parameters.

General parameters

Calculate width using street integration	Uses the graph topology to calculate street and sidewalk widths using graph connectivity (slower), otherwise randomly distributed street widths are used (faster).

Street integration parameters

Minimum number of street lanes	The minimum number of lanes that each street may have.
Maximum number of street lanes	The maximum number of lanes that each street may have.
Minimum sidewalk width	The minimum width of each street's sidewalks.
Maximum sidewalk width	The maximum width of each street's sidewalks.

Randomly distributed street widths parameters

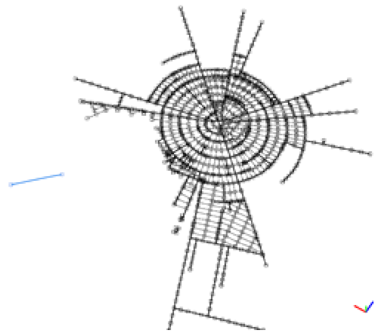
Width of Major Streets	The average street width of a major street. See Width Deviation of Major Streets.
Width Deviation of Major Streets	The street width deviation for major streets. The street width is randomly set within the interval [Width of Major Streets - Width Deviation of Major Streets, Width of Major Streets + Width Deviation of Major Streets].
Sidewalk Width of Major Streets	The average sidewalk width of a major street. See Sidewalk Width Deviation of Major Streets.
Sidewalk Width Deviation of Major Streets	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].
Width of Minor Streets	The average street width of a minor street. See Width Deviation of Minor Streets.
Width Deviation of Minor Streets	The street width deviation for minor streets. The street width is randomly set within the interval [Width of Minor Streets - Width Deviation of Minor Streets, Width of Minor Streets + Width Deviation of Minor Streets].
Sidewalk Width of Minor Streets	The average sidewalk width of a minor street. See Sidewalk Width Deviation of Minor Streets.

Sidewalk Width Deviation of Minor Streets	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].

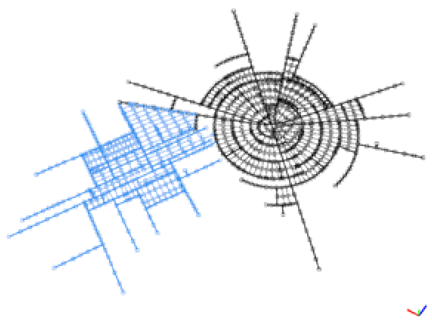
Street pattern examples

Workflow examples as impressions

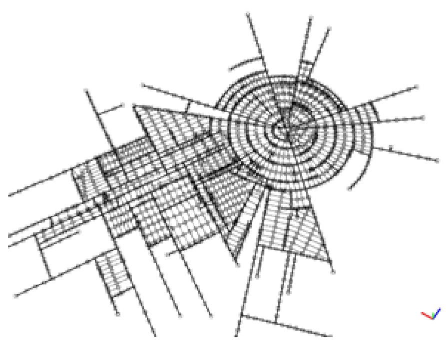
- Create a separated single street.



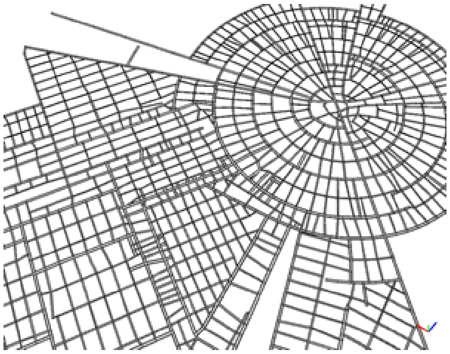
- First growth phase with raster pattern.



- Second growth phase with raster pattern.

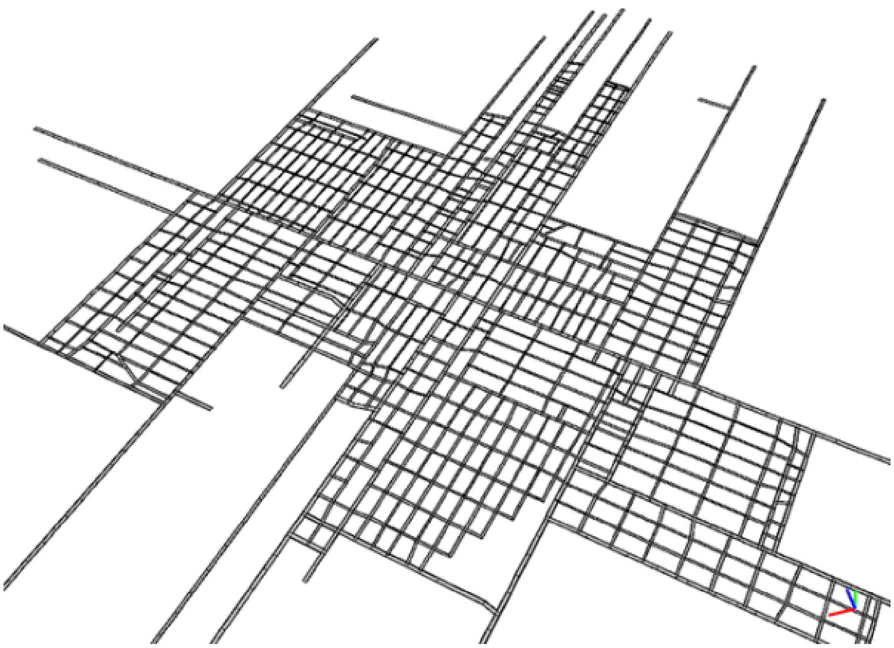


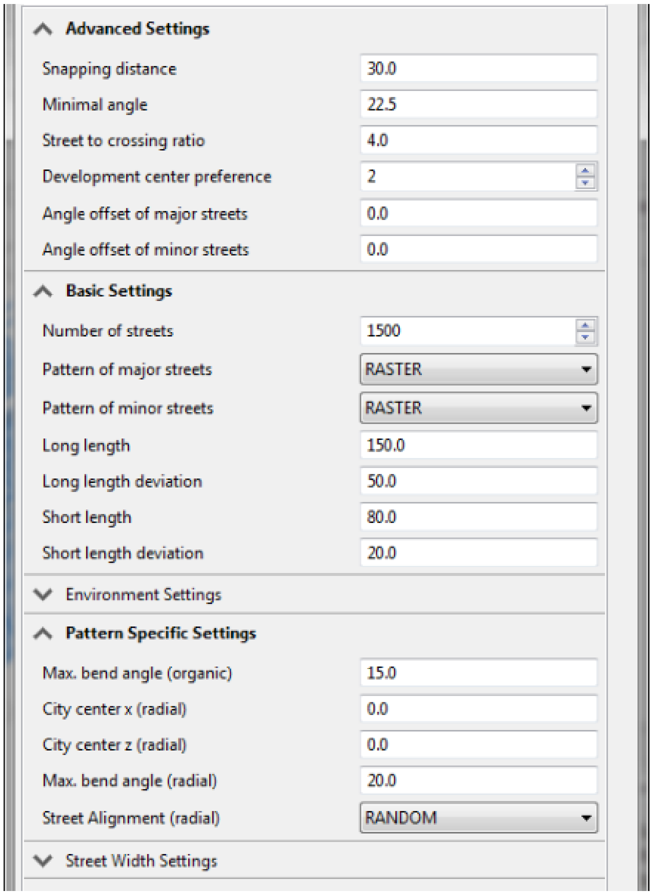
- The two networks are connected.



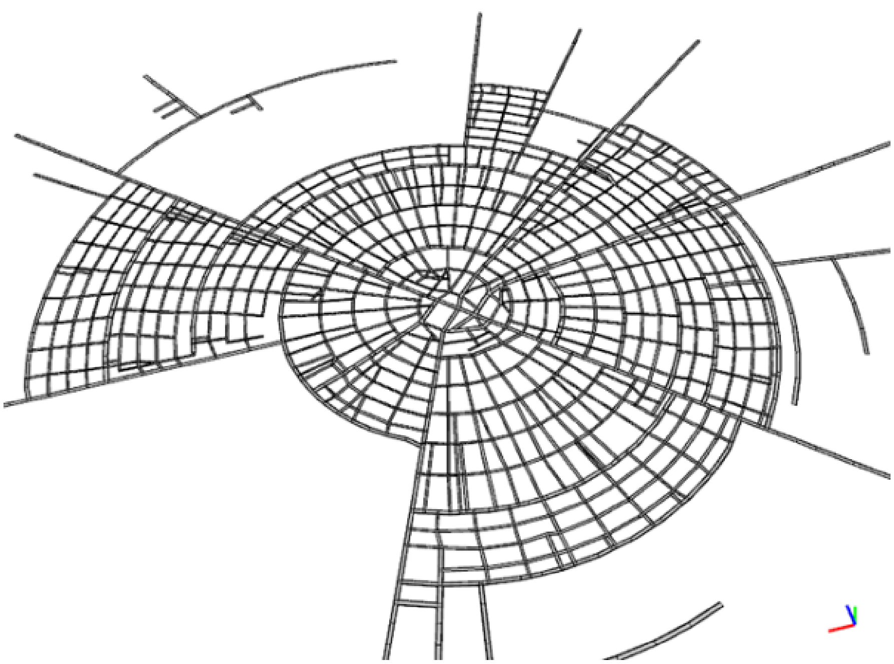
Parameter sets

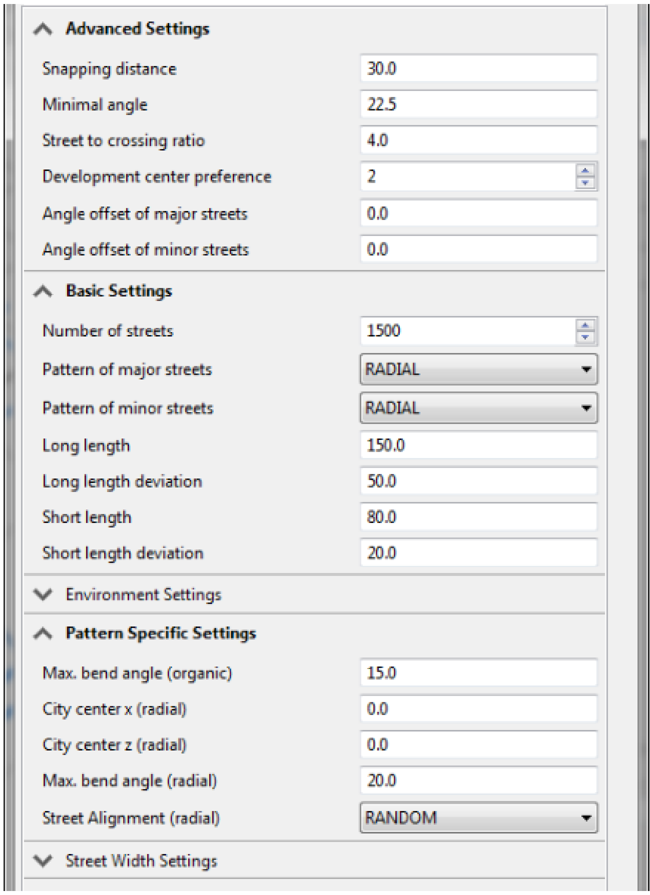
- Raster pattern.



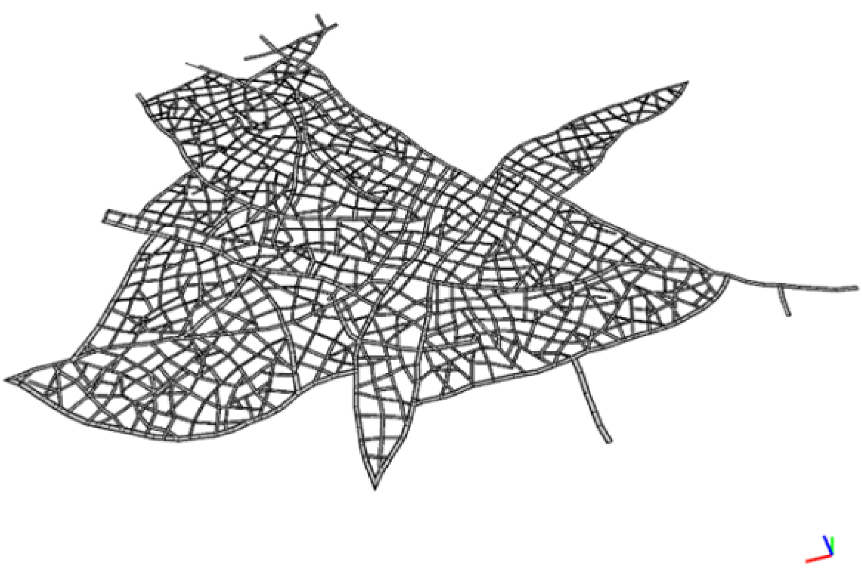


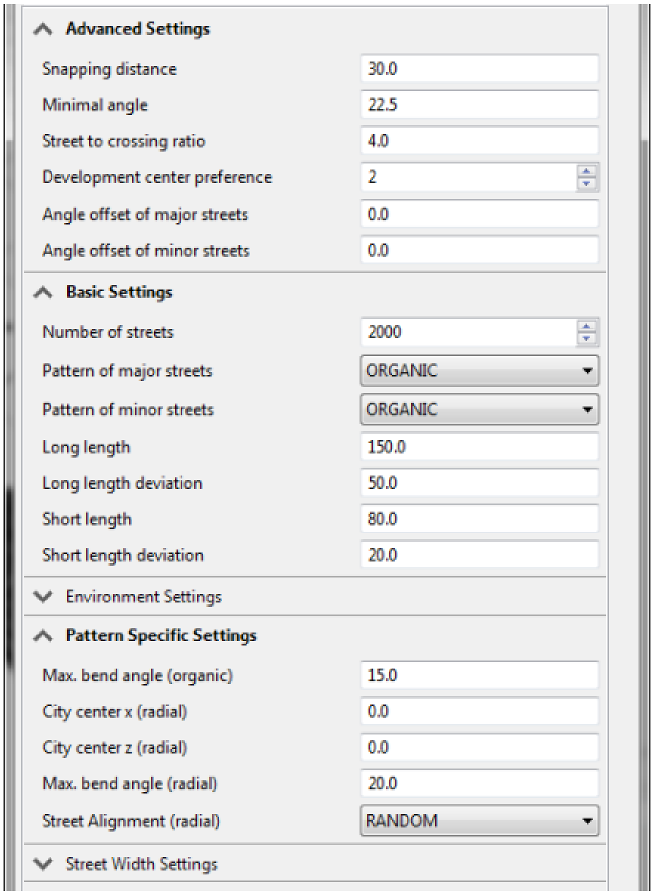
- Radial pattern.



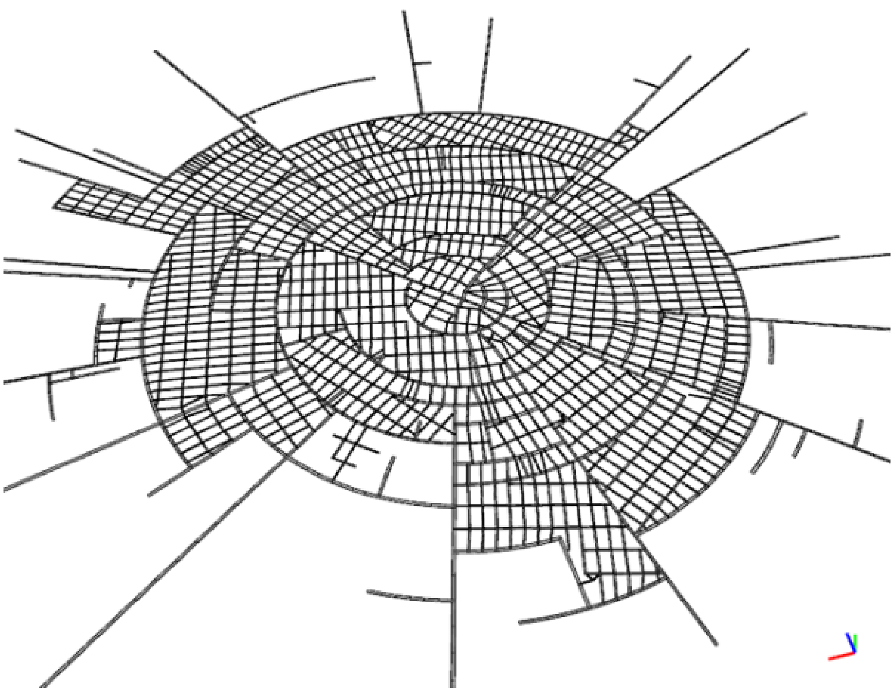


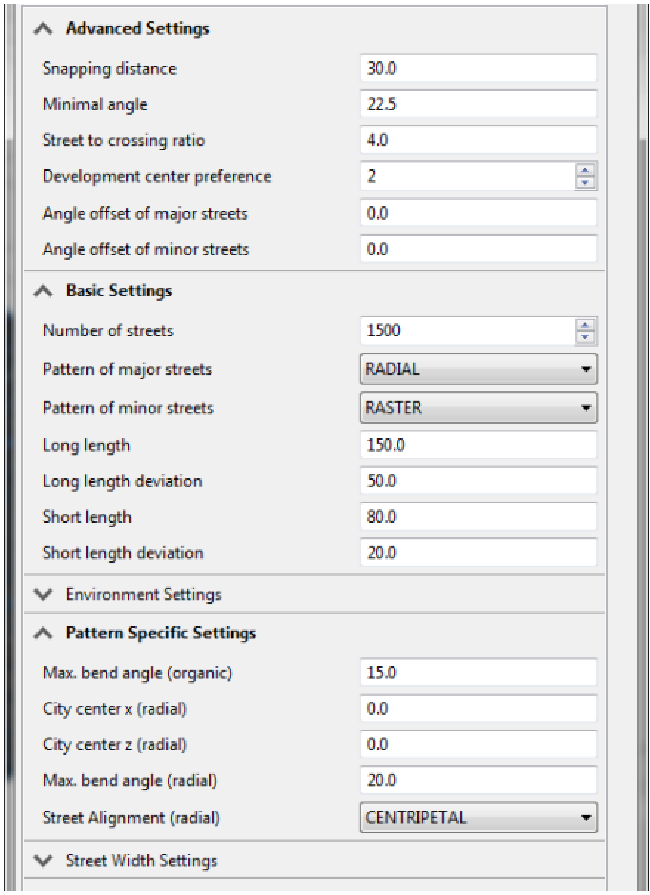
- Organic pattern.



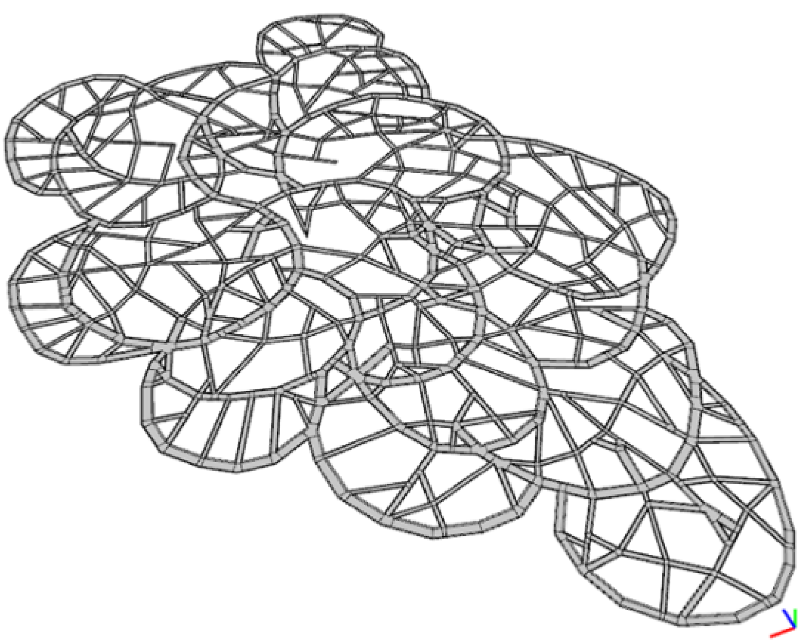


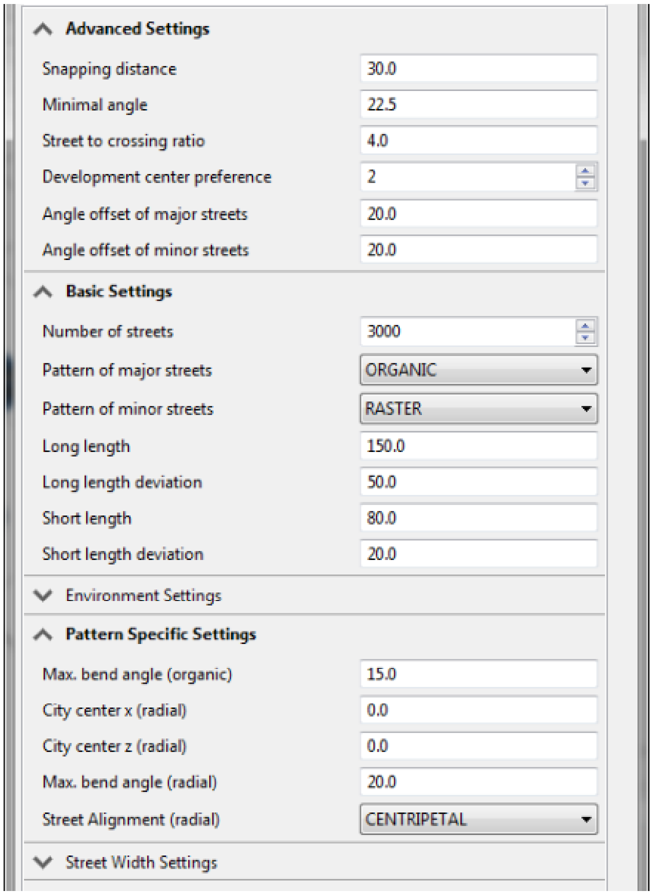
- Radial major streets with raster pattern on minors.



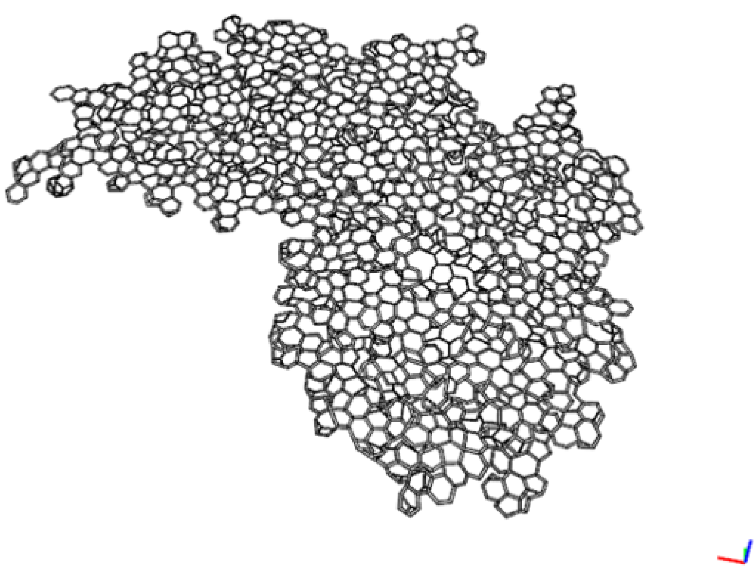


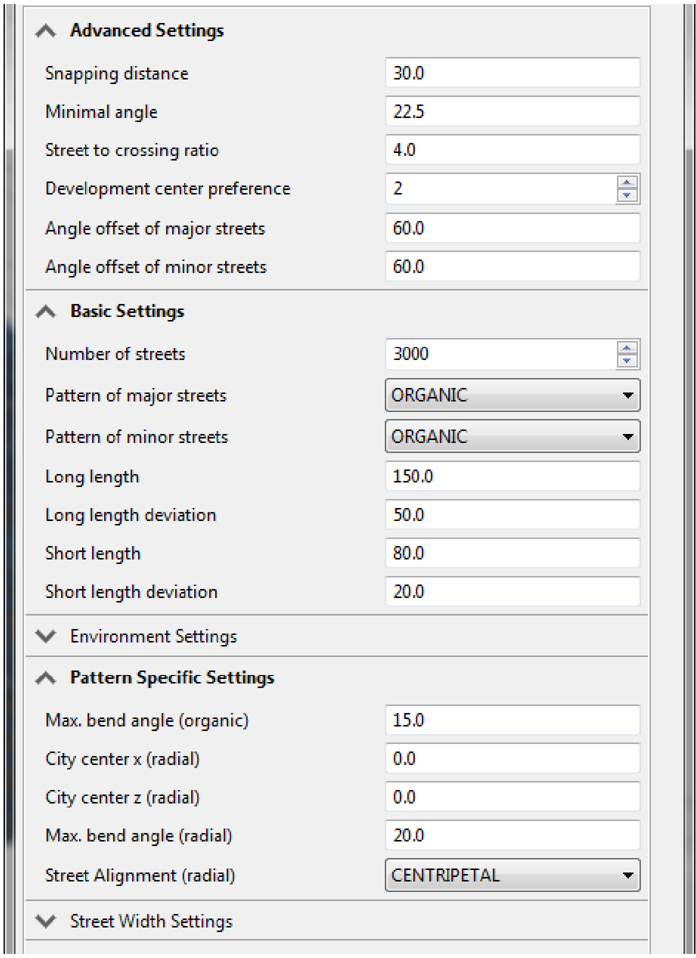
- Organic circle pattern.



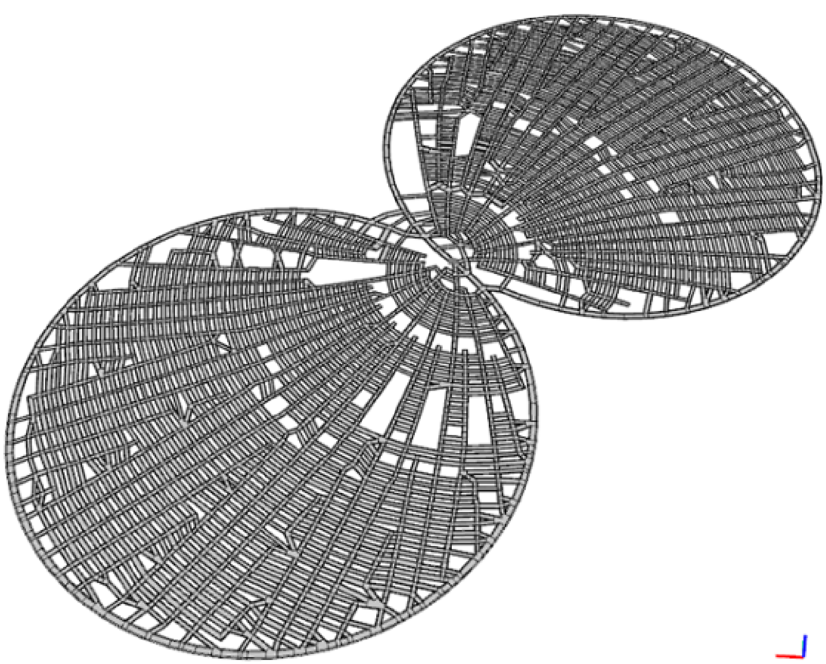


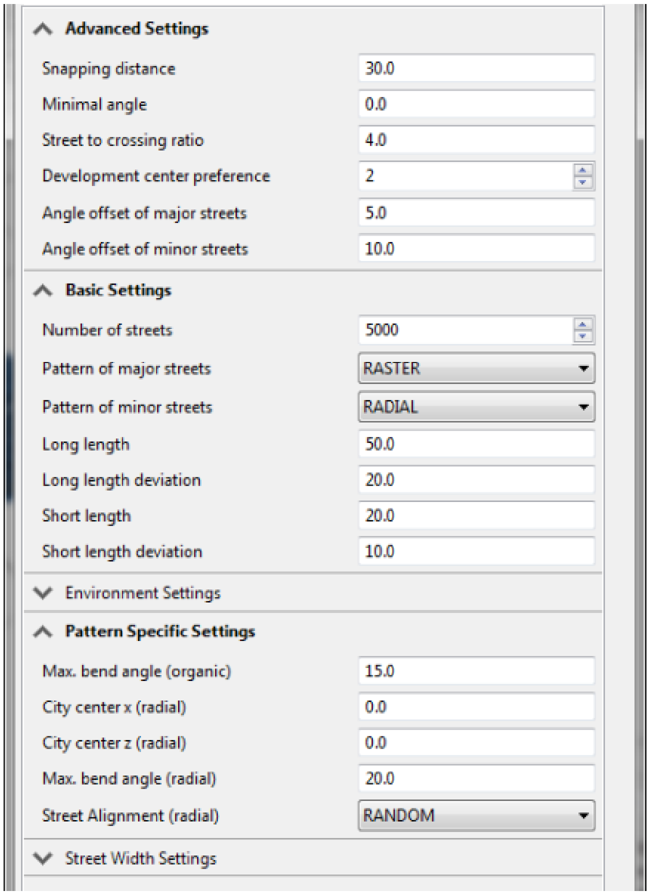
- Honeycomb style.



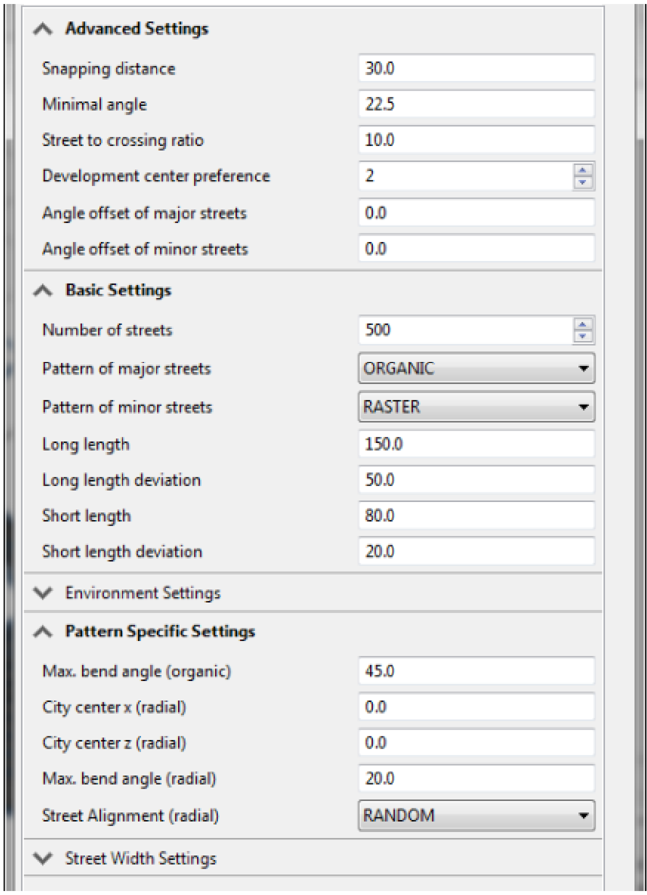


- The Glasses style.





- Organic distribution of rasters.

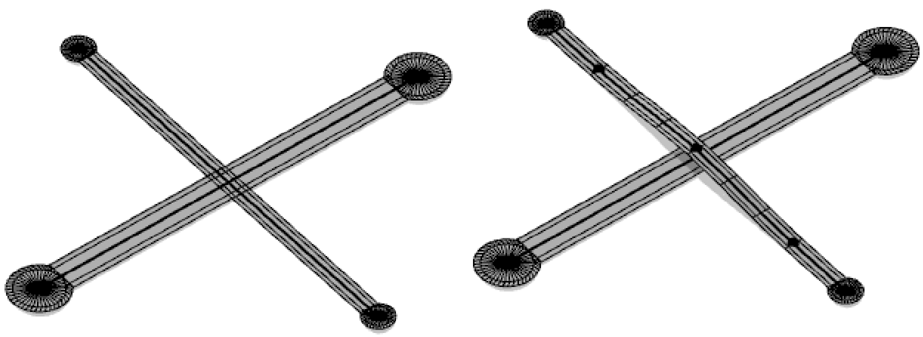

Advanced Settings
Snapping distance
30.0
Minimal angle
22.5
Street to crossing ratio
10.0
Development center preference
2
Angle offset of major streets
0.0
Angle offset of minor streets
0.0
Basic Settings
Number of streets
500
Pattern of major streets
ORGANIC
Pattern of minor streets
RASTER
Long length
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

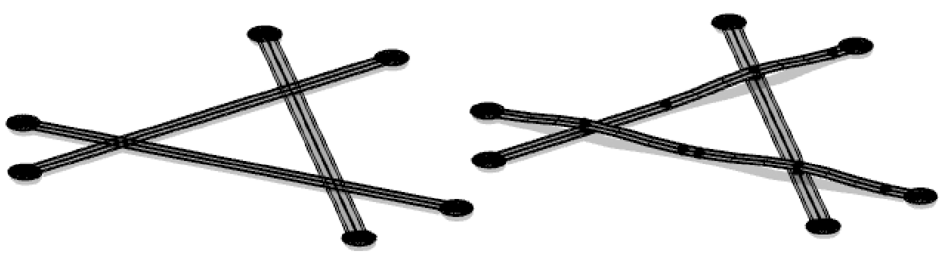
Imported or manually created street networks often lack elevation data, which is necessary for the 3D display of crossing streets. The **Generate Bridges** tool can automatically create such data. It can by executed by clicking **Graph** > **Generate Bridges....** in the main menu. It operates either 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, please 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

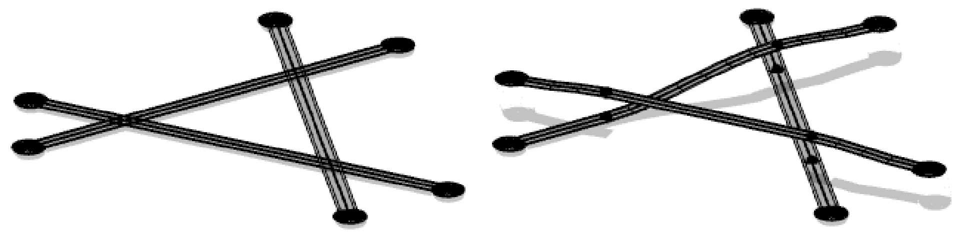


(left) Multiple Streets without elevation (right) Multiple streets with elevation data added after tool

Generate bridges settings

Level height

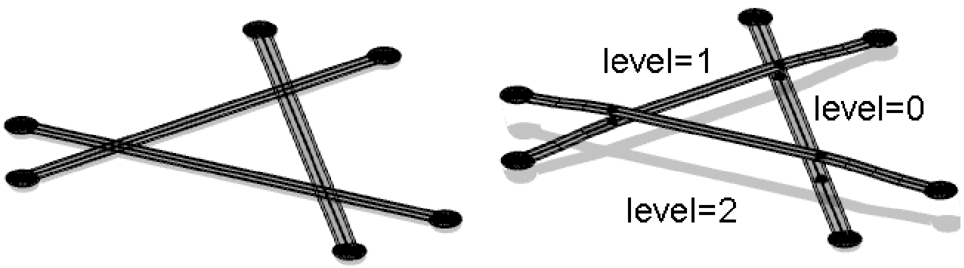
Vertical distance to be set between two crossing streets. Note that the 'Ramp maximum slope' influences the resulting node elevations.



(left) Original (right) Level height = 20

Object attribute for level (optional)

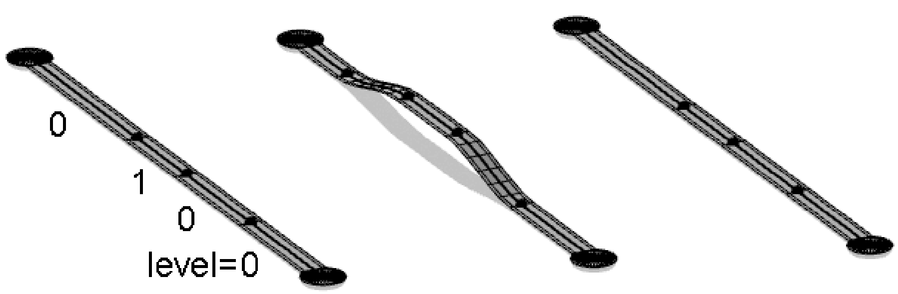
The height coordinate can be calculated from specific object attributes of streets. The height coordinate is set to "Level height" multiplied with the indicated attribute name. Attributes can either originate from imported data or be manually assigned, allowing full control of the vertical street layering.



(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.



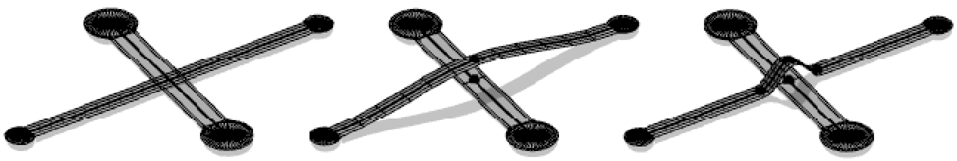
(left) Original with faulty 'level' attribute value (middle) Option unchecked (right) Option checked

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

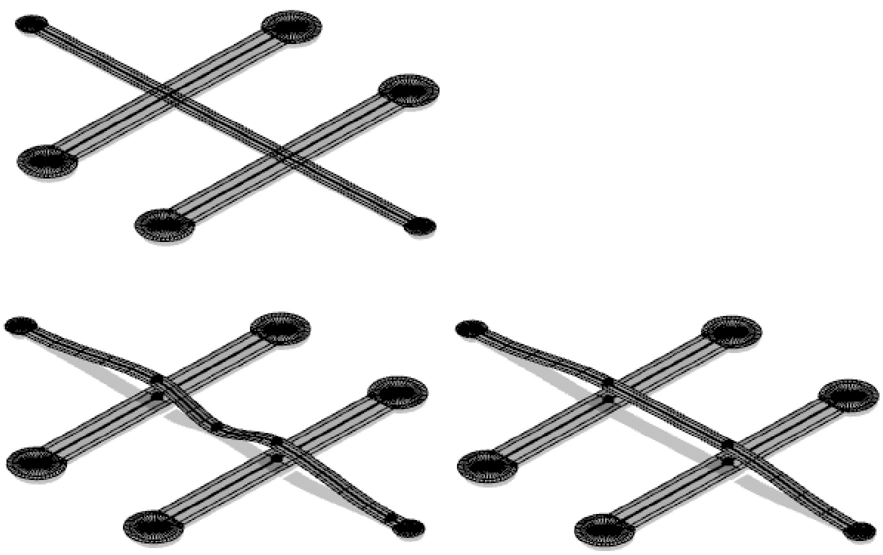
Maximum slope of ramps (vertical climb per horizontal unit).



(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.



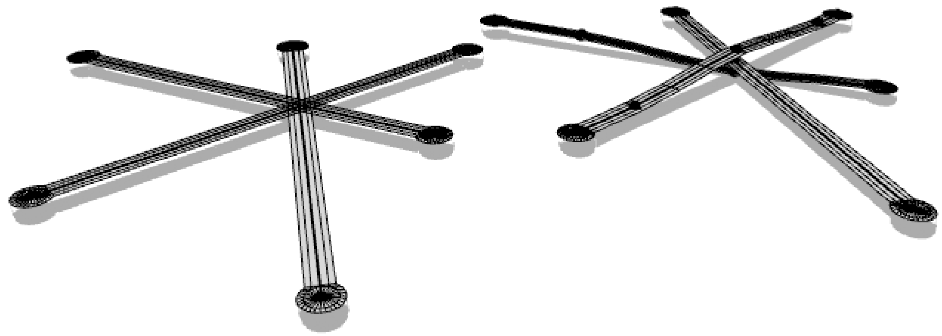
(top) Original (bottom left) Bridge join = 0.1 (bottom right) Bridge join = 1.0

Lock non-zero heights

Do not change height of nodes with non-zero position.

Allow tunnels

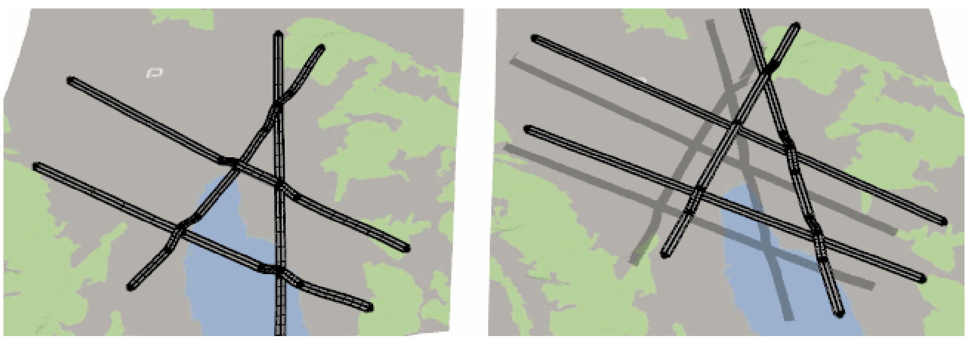
Allow tunnels (streets below zero height). Note that this is quite a rare state to construct. Note also that the Shadow Plane is rendered at smallest (thus negative) elevation value.



(left) Original (right) Solved with tunnels

Use visible terrain

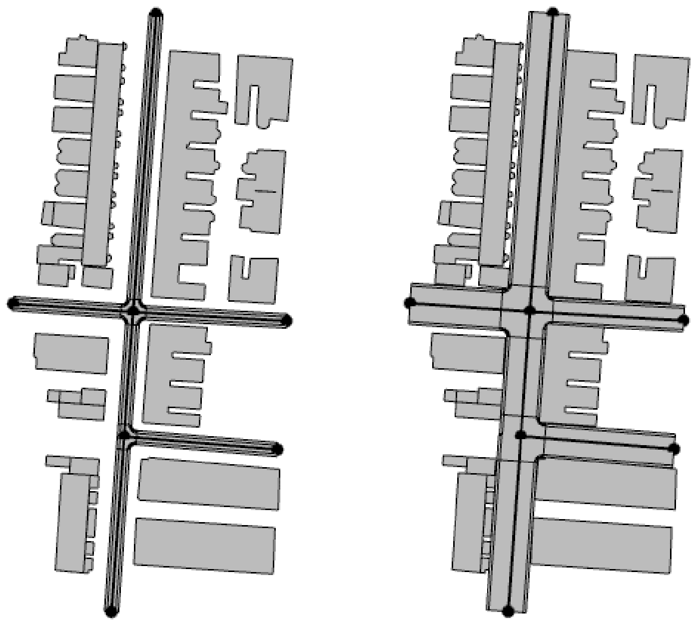
Treat all heights as relative above terrain (if any). This causes bridges to follow the terrain.



(left) Use visible terrain (right) Ignore visible terrain

Fit widths to shapes

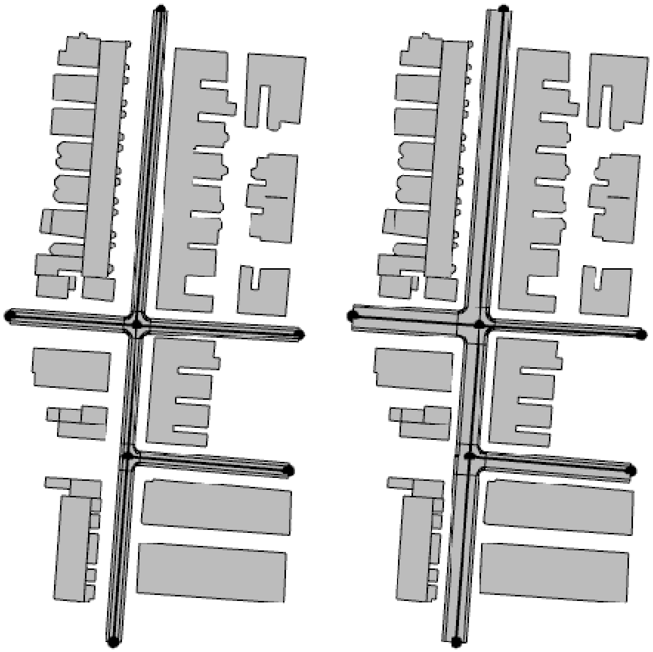
Often, datasets with street center lines do not have width attributes. Using the **Fit Widths to Shapes** tool lets the user easily adjust them accordingly. It can be executed by clicking **Graph** > **Fit Widths to Shapes...** in the main menu.



(left) Original streets do not line up with the footprint shapes (right) Widths and offsets are adjusted to touch the footprints

Parameters

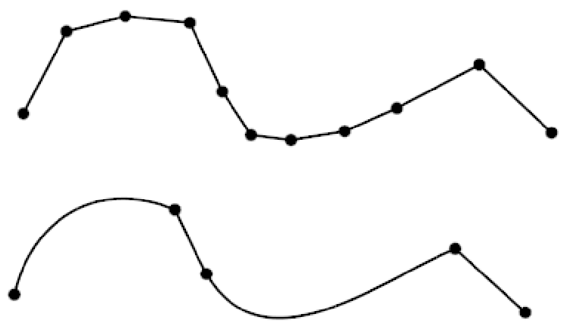
Max Street Width	Maximum width the streets will be fitted to. When the calculated width is larger than this parameter, the street is left unchanged.
Min Street Width	Minimum width the streets will be fitted to. When the calculated width is smaller than this parameter, the street is left unchanged.
Sidewalk Scale	Determines if sidewalks widths are retained ("Do Not Change") or scaled proportionally to the street ("Scale Proportionally").
Adjust Street Offsets	Determines if the street offsets should be adjusted to better fit the static shapes.
Additional Margin	This allows to increase the margin between streets and static shapes.



(left) Original (right) Additional Margin = 3

Simplify graph

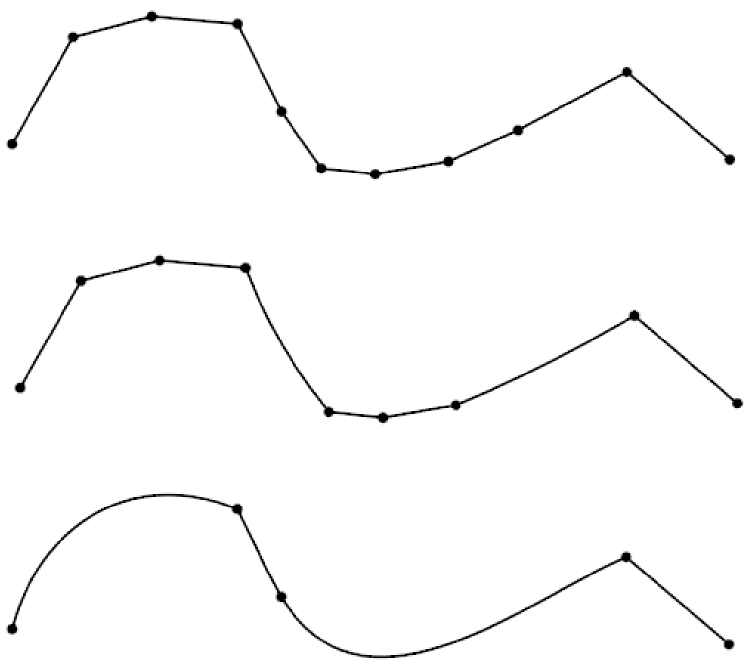
Street graphs often contain a lot of contiguous short straight streets, which can be simplified to longer curved streets with this tool. It can by executed by clicking the main **Graph** > **Simplify Graph...** in the main menu. It operates either on the current street selection or on all streets when nothing is selected.



(top) Street with unnecessary nodes (bottom) simplified version

Parameters

Threshold Angle: Angle in degrees. Streets with a higher angle than this form boundaries between fitted curves.



(top) Input street (middle) Threshold = 10 (bottom) Threshold = 50

Analyze graph

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.

In order to open the dialog, select a set of graph segments or a graph layer and click **Graph** > **Analyze Graph...** from the main menu.

Settings

Mode	Three modes are available: • **Calculate analysis only (as object attribute)**: For each street the three analysis values global integration, local integration and inbetween centrality are computed. These values are stored as object attributes. • **Visualize analysis (assign rule)**: 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 **Inspector**. • **Set street width (based on integration)**: 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/change the mapping code in the layer attributes in the inspector.
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, only used in Set street width mode.
Street Width Max	Street width upper bound, 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 which pass this segment is computed. Then, the values are normalized so that each value is in the range zero to one.

Example

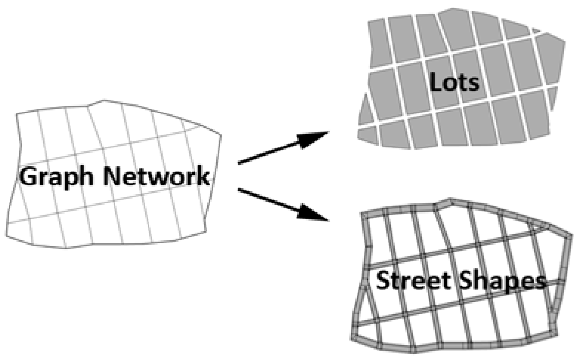
The global integration of a typical street network

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 (Block, Lot, Node, Segment, Sidewalk) 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.

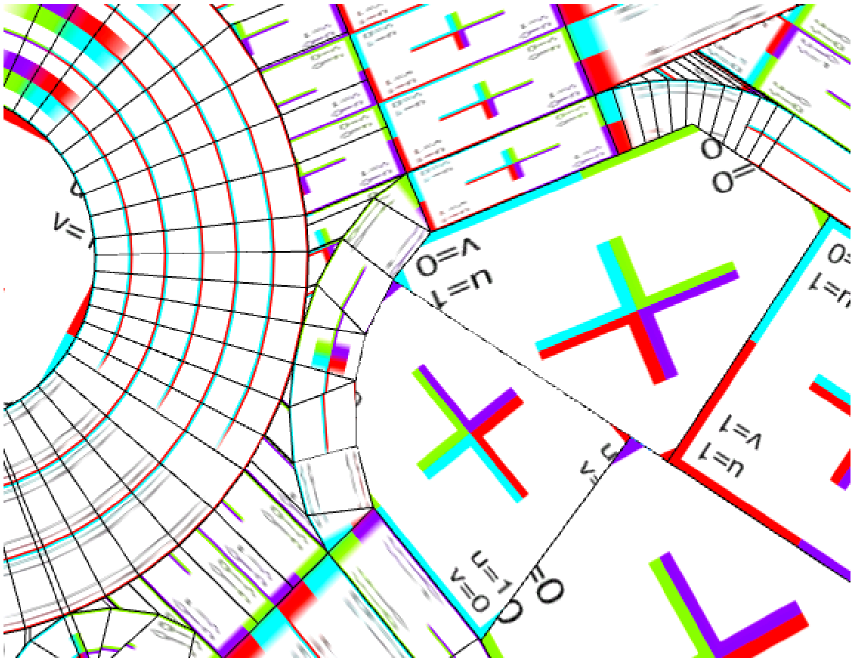
Note:
For each loop in the graph network, a block is automatically created, see Graphs for details.

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

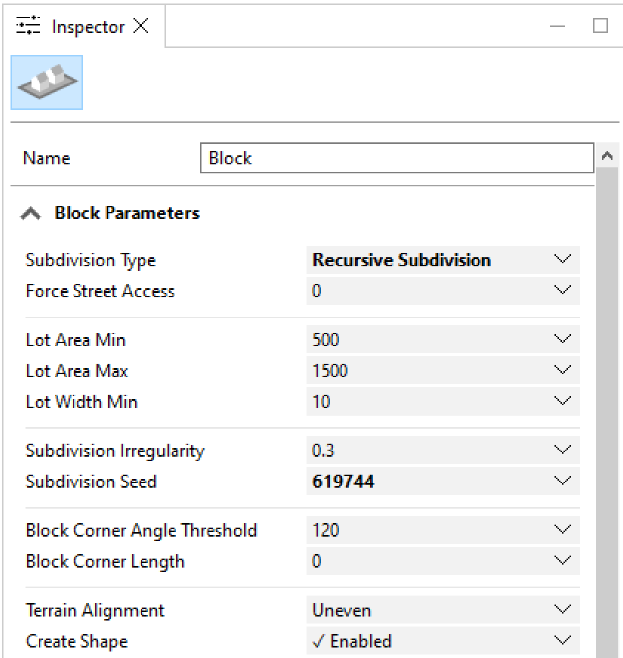
UV coordinates are generated for each shape. They can be used for UV Splits and texturing. For details of the UV coordinates, see Street and Intersection Shape UVs.



UV Coordinates

Block parameters

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



Block Parameters

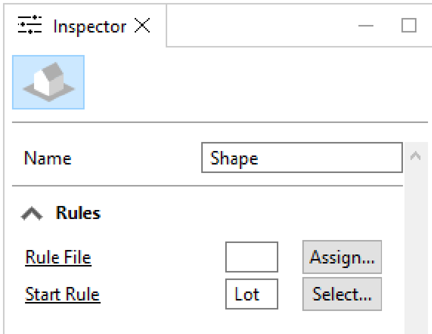
Note:
Parameters (attributes) can be mapped to Default, User, Object or to a map layer. See Mapping Attributes for details.

Lot default rules

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

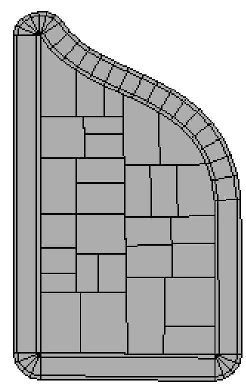
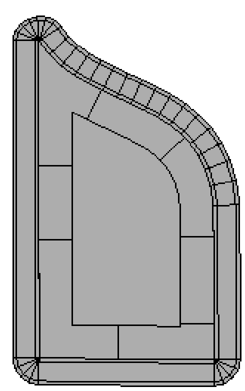
Lot	Polygonal shape touching a street.
LotInner	Lots that reside within a block but do not touch a street.
LotCorner	Created when the corners of a lot are given a width when you set the `Block Corner Length` parameter on a block with recursive or offset subdivision.

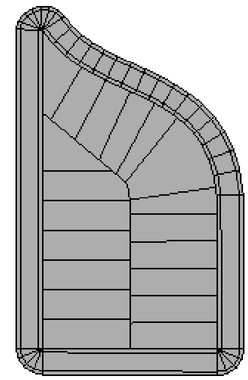
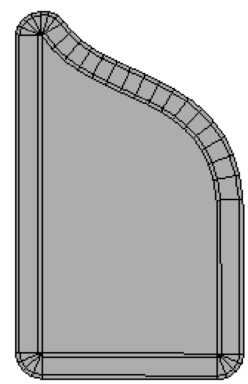
When you select a lot shape within a block, it displays the one of the start rules:



Subdivsion type parameters

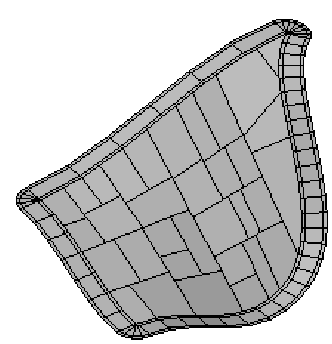
Several parameters are available that allow you to control the street shapes, see Subdivision types .

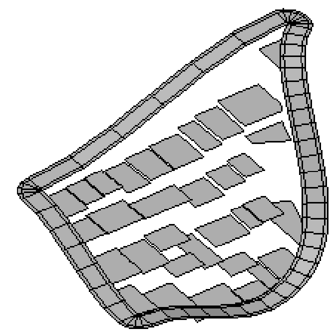
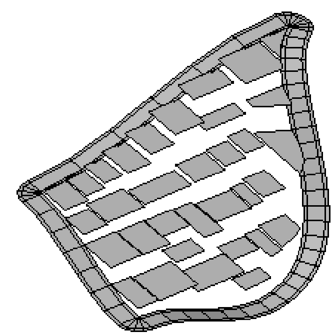
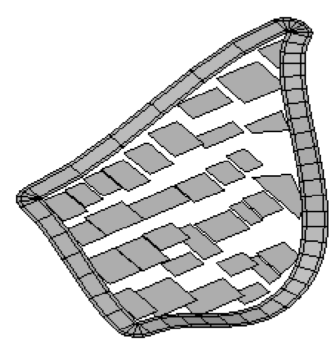
Subdivsion Type (attr type)	
Specifies the subdivision algorithm to use: `Recursive Subdivision`, `Offset Subdivision`, `Skeleton Subdivision`, or `No Subdivision`	
`Recursive Subdivision` creates rectangular lots by repeatedly splitting the block.	
`Offset Subdivision` creates lots only within a given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

`Skeleton Subdivision` creates street-aligned lots that always have access.	
`No Subdivision` doesn't perform subdivision on this Shape.	

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]	
0. `Uneven`. The lots follow the terrain, given uneven heights.	

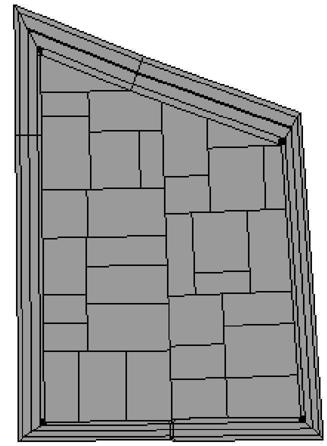
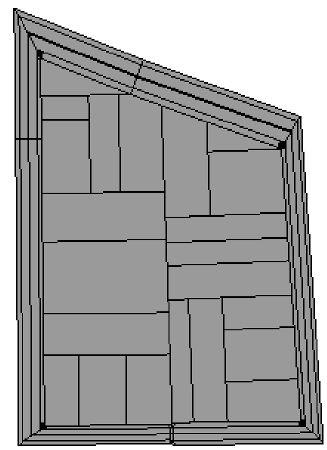
1. `Minimum`. The lots lie at their lowest point of the terrain that they cover.	
2. `Maximum`. The highest point of the lot is used.	
3. `Average`. The average height of the lot is used.	

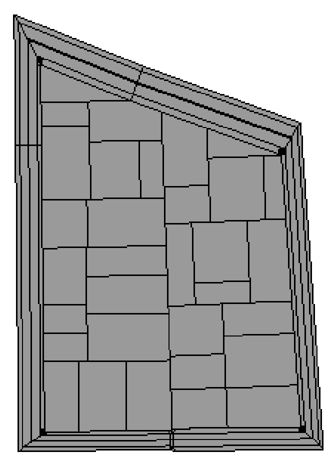
Create Shape (attr shapeCreation)
Enable/disable the shape geometry creation from the segment.

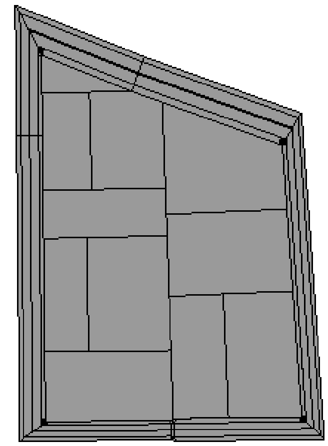
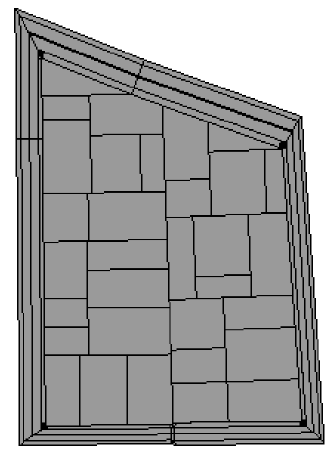
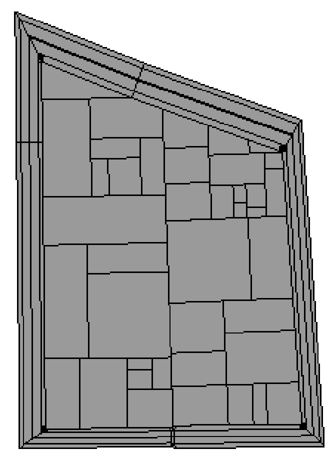
Recursive subdivision

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

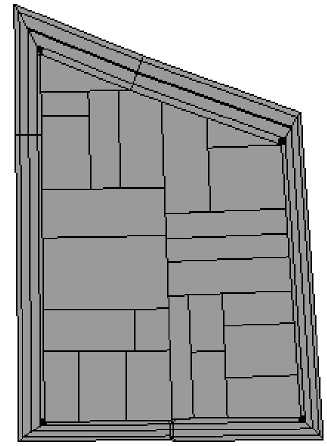
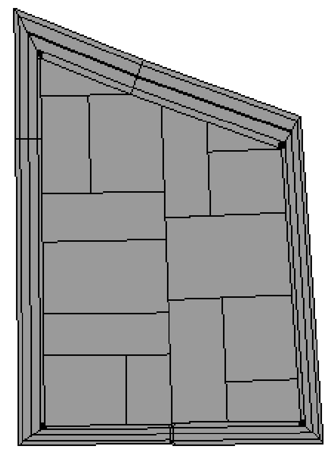
Force Street Access (attr forceStreetAccess)
The factor indicating the preference for lots with street access. A higher value results in more lots having street access.

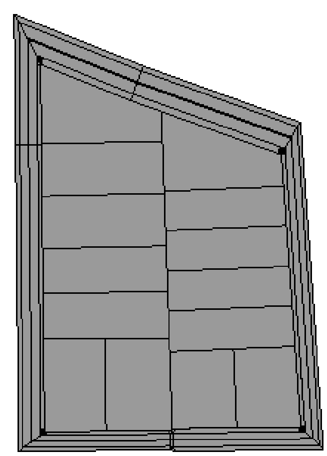
Given in range [0.0,1.0]	
Subdivision obtained for a `Force Street Access` value close to 0.	
Subdivision obtained for a `Force Street Access` value close to 1.0.	

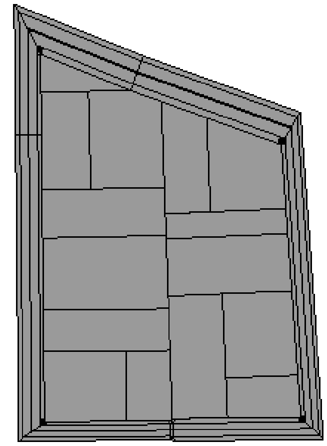
Lot Area Min (attr lotAreaMin) and **Lot Area Max (attr lotAreaMax)**	
The approximate lower and upper bounds of the area of lots obtained after subdivision.	
Given in absolute area units	
Subdivision obtained for smaller `Lot Area Min` and `Lot Area Max` values.	

Subdivision obtained for larger `Lot Area Min` and `Lot Area Max` values.	
Subdivision obtained when the difference between `Lot Area Max` and `Lot Area Min` is small.	
Subdivision obtained when the difference between `Lot Area Max` and `Lot Area Min` is large.	

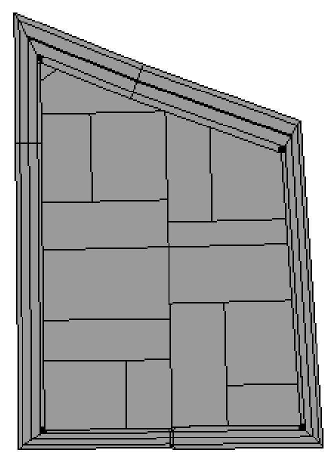
Lot Width Min (attr lotWidthMin)
The minimum width of the side of a lot. Subdivision stops if the length of any of the sides of any of the resulting lots is less than this value. If this value is high, the area of resulting lots might be larger than the area specified by `Lot Area Max`.
Given in absolute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

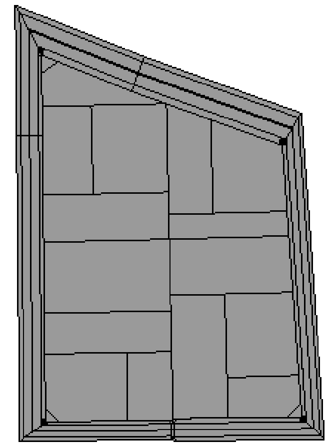
Subdivision obtained for a smaller `Lot Width Min` value.	
Subdivision obtained for a larger `Lot Width Min` value.	

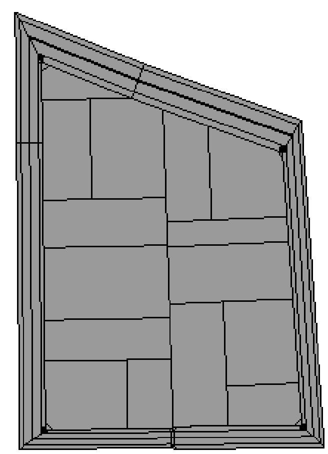
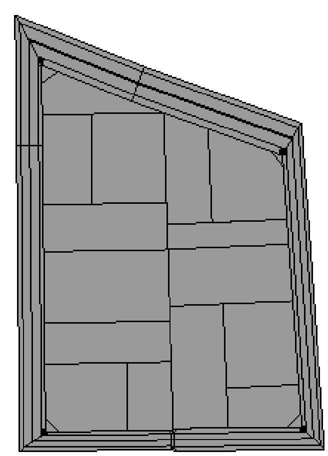
Subdivision Irregularity (attr irregularity)	
The relative deviation of the split line from the middle point of the center of the 'oriented bounding box' (OBB). If this value is 0.0, the split line will be pivoted at the middle point of the OBB of the parent lot. A higher value results in the split line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

Subdivision obtained for an `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.

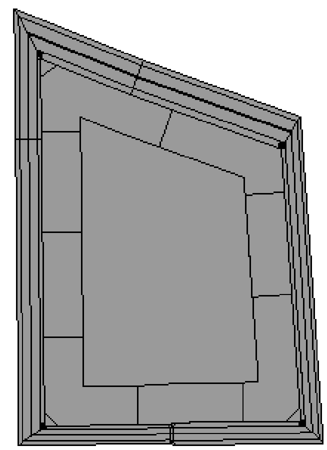
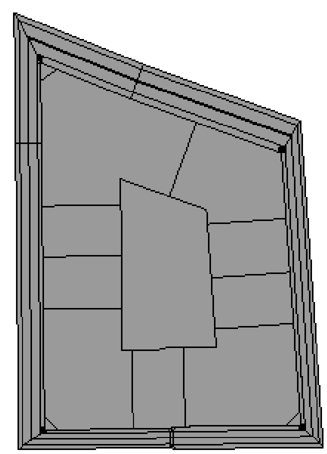
Block Corner Angle Threshold (attr cornerAngleMax)	
Corner angle threshold. If the angle at the vertex of a block contour is less than this value, a corner lot is inserted. A larger value results in a more relaxed criterion for inserting corners, and thus in more corners being created. If this value is 0.0 no corners are created.	
Given in degrees	
Subdivision obtained for a smaller `Block Corner Angle Threshold` value.	

<table>
<tr><td>Subdivision obtained for a larger <code>Block Corner Angle Threshold</code> value.</td><td></td></tr>
</table>

<table>
<tr><td colspan="2">Block Corner Length (attr cornerWidth)</td></tr>
<tr><td colspan="2">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.</td></tr>
<tr><td colspan="2">Given in absolute length units</td></tr>
<tr><td>Subdivision obtained for a smaller <code>Block Corner Length</code> value.</td><td></td></tr>
<tr><td>Subdivision obtained for a larger <code>Block Corner Length</code> value.</td><td></td></tr>
</table>

Offset subdivision

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

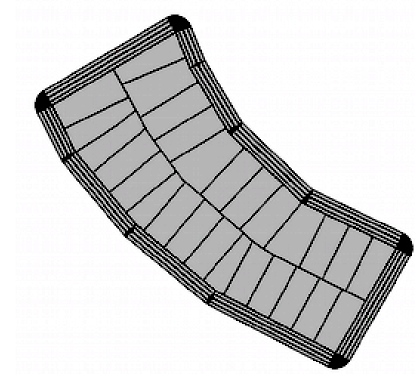
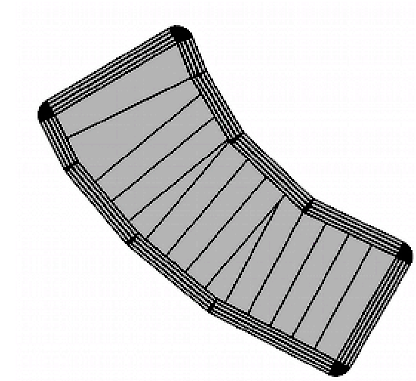
Offset Width (attr offsetWidth)	
The perpendicular distance from the block contour to the inwards offset polygon. Intuitively, this value corresponds to the depth of the lots that are created when offset subdivision is used. • If this value is close to 0.0, OBB subdivision is used. • If this value is high enough so that the offset polygon is collapsed, OBB subdivision is used.	
Given in absolute length units	
Subdivision obtained for a smaller `Offset Width` value.	
Subdivision obtained for a larger `Offset Width` value.	

Recursive Subdivision
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.

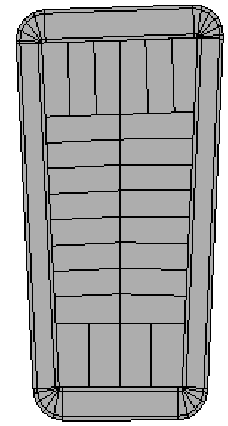
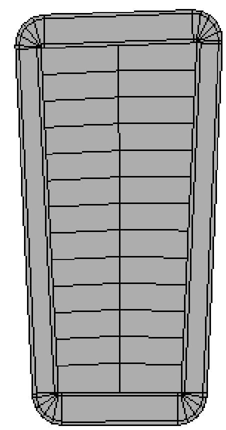
See Recursive subdivision for the rest of the specific parameters.

Skeleton subdivision

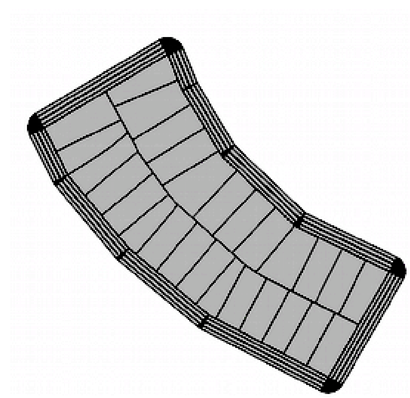
Skeleton subdivision attempts to subdivide a block such that every lot has access to the street. The sides of the lots are perpendicular to the roads they are adjacent to.

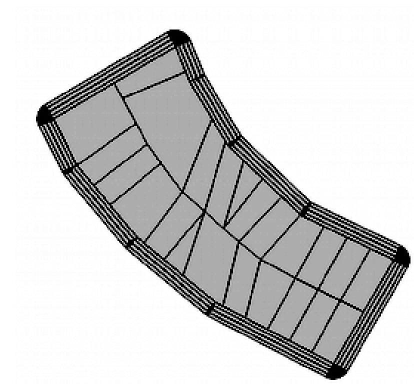
Shallow Lot Fraction (attr shallowLotFrac)	
Limit for merging triangular lots.	
Subdivision obtained for a smaller `Shallow Lot Fraction` value.	
Subdivision obtained for a larger `Shallow Lot Fraction` value.	

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`.

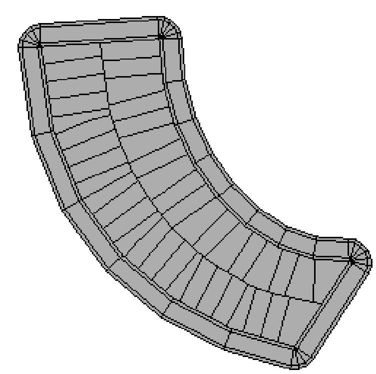
`Street width`: the widest street takes priority. If the streets have similar average widths, the street length is used instead.	
`Street length`: the longest street takes priority.	

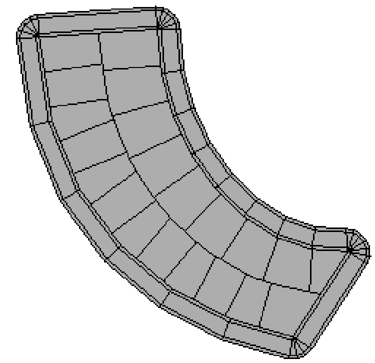
Simplify (attr simplify)
Amount of simplification that occurs. A high value creates irregular lots with fewer vertices.
Given in range [0.0,1.0]

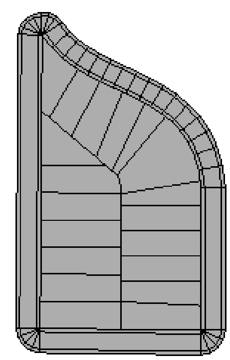
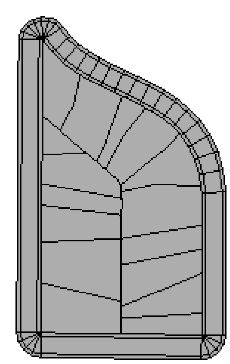
Subdivision obtained for a smaller `Simplify` value.	

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 possesses. 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.	

Subdivision obtained for a `Lot Width Min` value of 30.	

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

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.

No subdivision

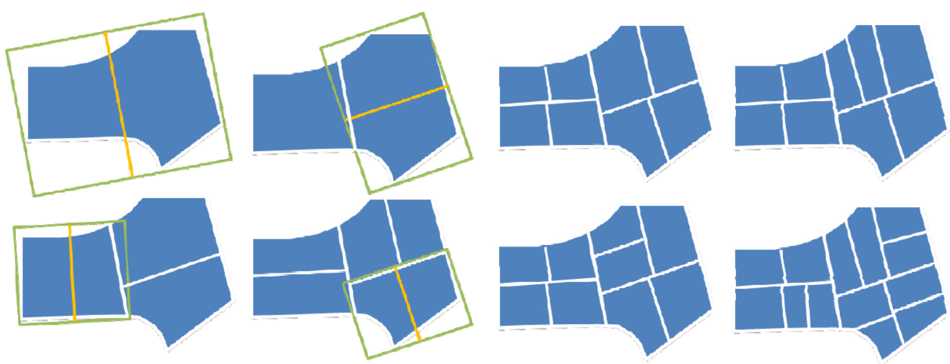
No Subdivision
This simple subdivision technique subdivides the block into a single lot of the same shape. There is an option to remove the lot's corners.

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 three 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.



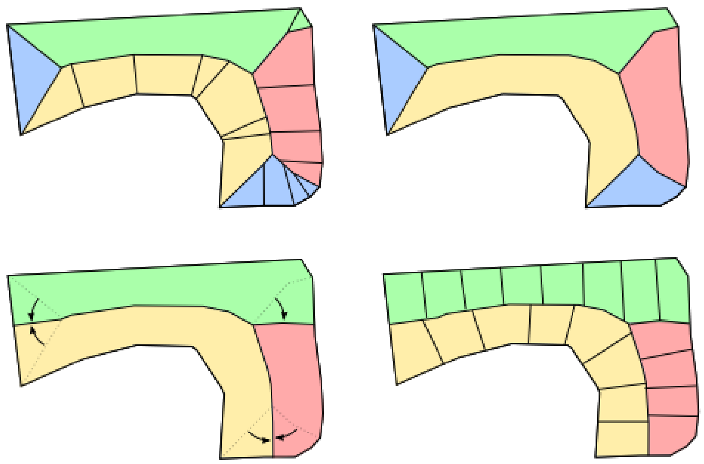
Successive steps of the recursive OBB algorithm

Offset algorithm

The offset subdivision algorithm computes the inwards offset of the block contour and subdivides into lots the stripe between the block contour and its offset. The inwards 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, we 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 we 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.



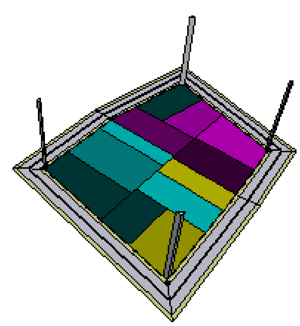
Successive steps of the skeleton subdivision algorithm

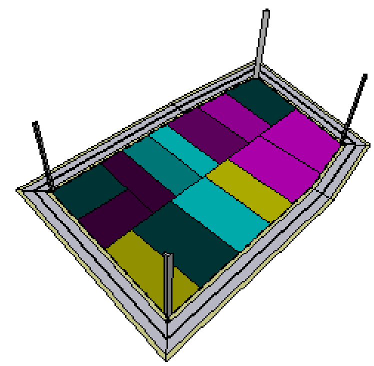
Consistent Indexing

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 might 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.

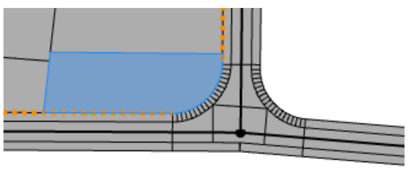
The same approach is also used to improve the consistency of the seed of each lot. As a result, two lots that are relatively in the same position of the block at two different subdivision configurations, have higher chances of sharing the same seed and attributes. The figure below shows a subdivision 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	

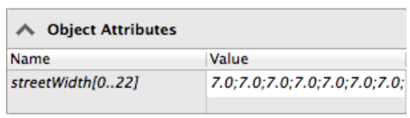
Subdivision for block after interactive editing	

Auto-generated street width attributes

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



A typical lot selected in the Viewport

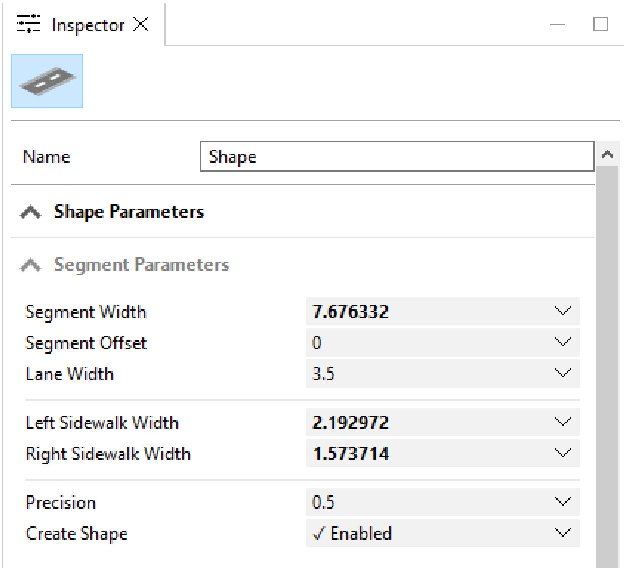


Street width attributes in the Inspector

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:

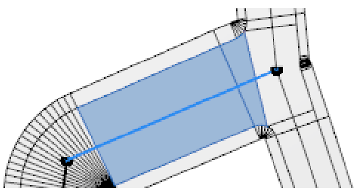
Segment Width (attr streetWidth)
Defines the width of the main street shape.
Segment Offset (attr streetOffset)
Defines the distance the geometry will be offset from the center line.
Lane Width (attr laneWidth)
Determines the width of lanes used for UV texture mapping of streets.
Left Sidewalk Width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

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

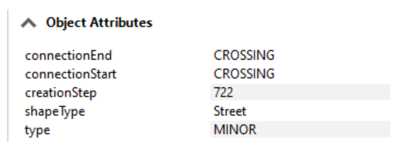
Note:
Street widths are given in absolute length units, precision is given in a normalized [0, 1] range.

Auto-generated 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



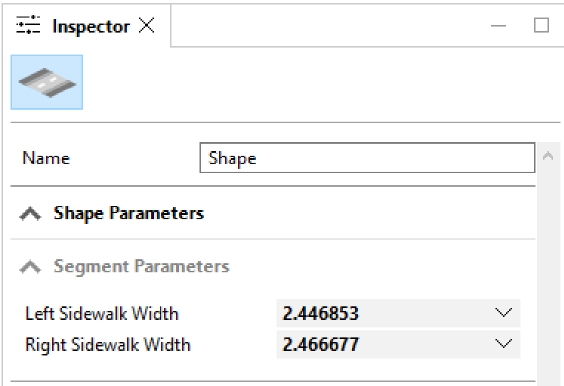
Street shape attributes in the Inspector

connectionEnd	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`
connectionStart	
shapeType	The `shapeType` is set to one of the following: • `Street`- Street shape • `Sidewalk`- Sidewalk shape

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

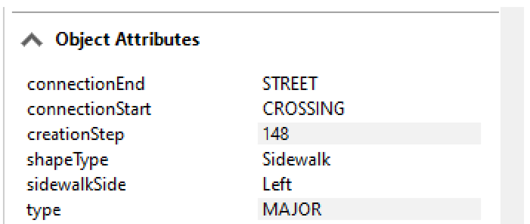
Sidewalk parameters

When sidewalk shapes are selected, you can manage parameters under **Sidewalk Parameters** in the **Inspector**. See Segment parameters for descriptions.



Sidewalk parameters

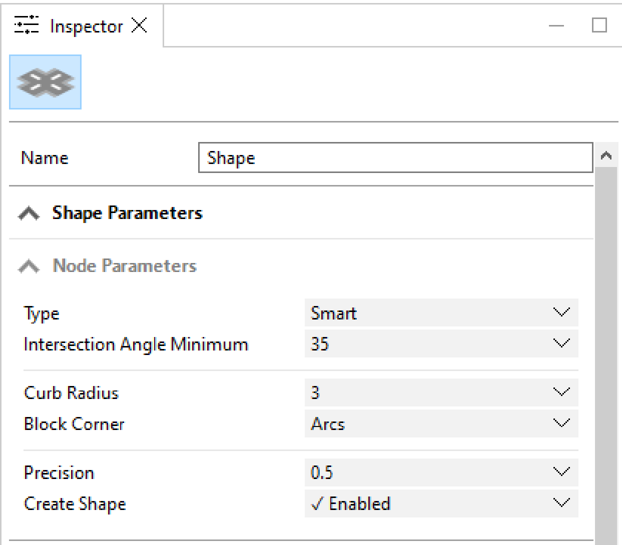
Under **Object Attributes**, the `shapeType` attribute is set to `Sidewalk` and the new `sidewalkSide` object attribute is added:



sidewalkSide	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 Shape are defined under **Node Parameters** in the **Inspector**.

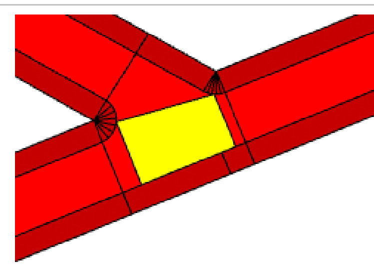


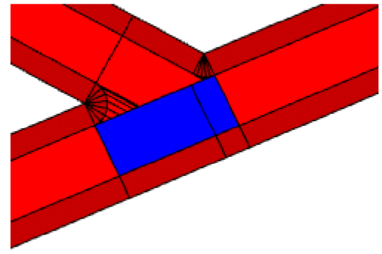
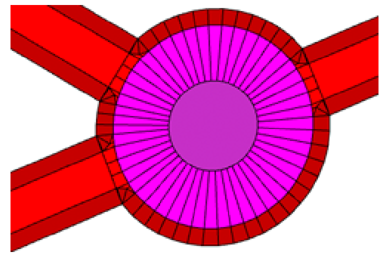
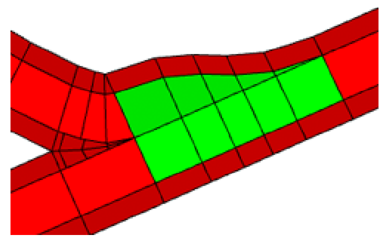
Node Parameters

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

Node type parameter

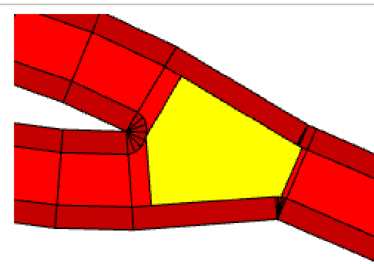
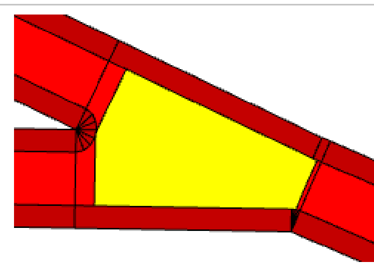
Several parameters are available for the user to control the resulting node shapes. Node parameters define the type of the node and specifies geometry details like the radius of the arcs for instance.

<table>
<tr><th colspan="2">Type (attr type)</th></tr>
<tr><td colspan="2">Specifies the type of the node. <code>Crossing</code>, <code>Junction</code>, <code>Roundabout</code> or <code>Freeway</code>. <code>Smart</code> automatically chooses between these types.</td></tr>
<tr><td>Crossing</td><td></td></tr>
</table>

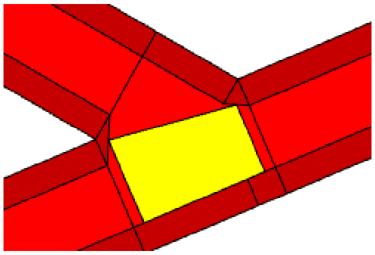
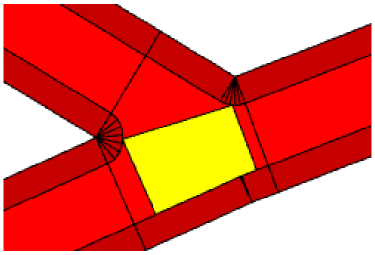
Junction	
Roundabout	
Freeway	

General node parameters

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

Intersection Angle Minimum (attr angleThreshold)	
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.	

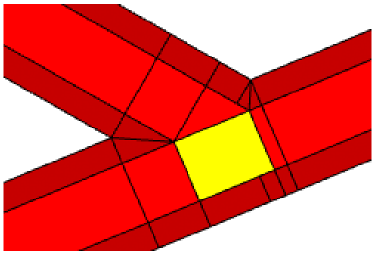
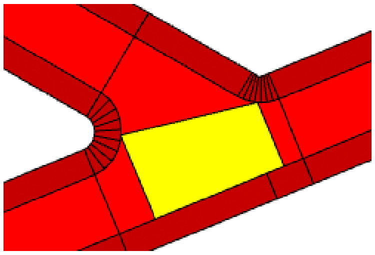
Precision (attr precision)

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`.	

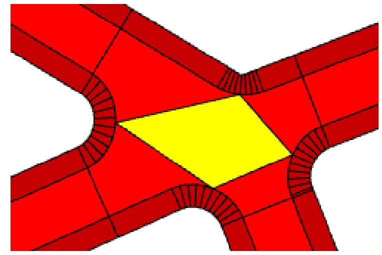
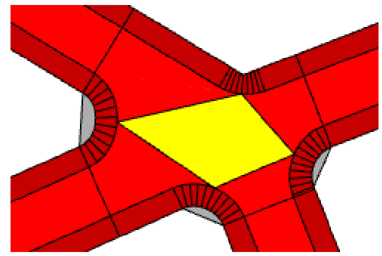
Create Shape (attr shapeCreation)
Enable/disable the shape geometry creation from the node.

Crossing and junction parameters

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

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`.	

Block Corner (atrr cornerStyle)
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`.	

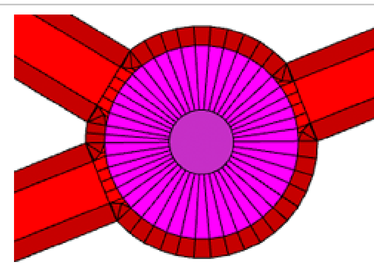
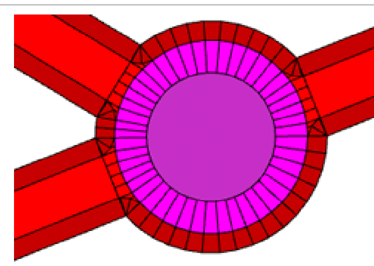
Note:
The `Block Corner` parameter is not available when `Type` is set to `Freeway`

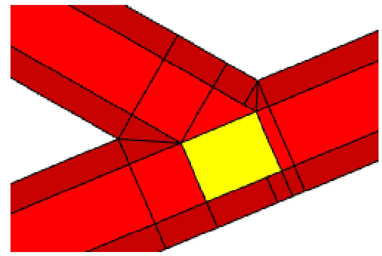
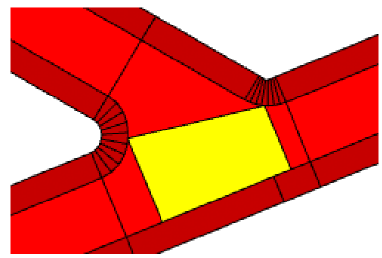
.

Roundabout parameters

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

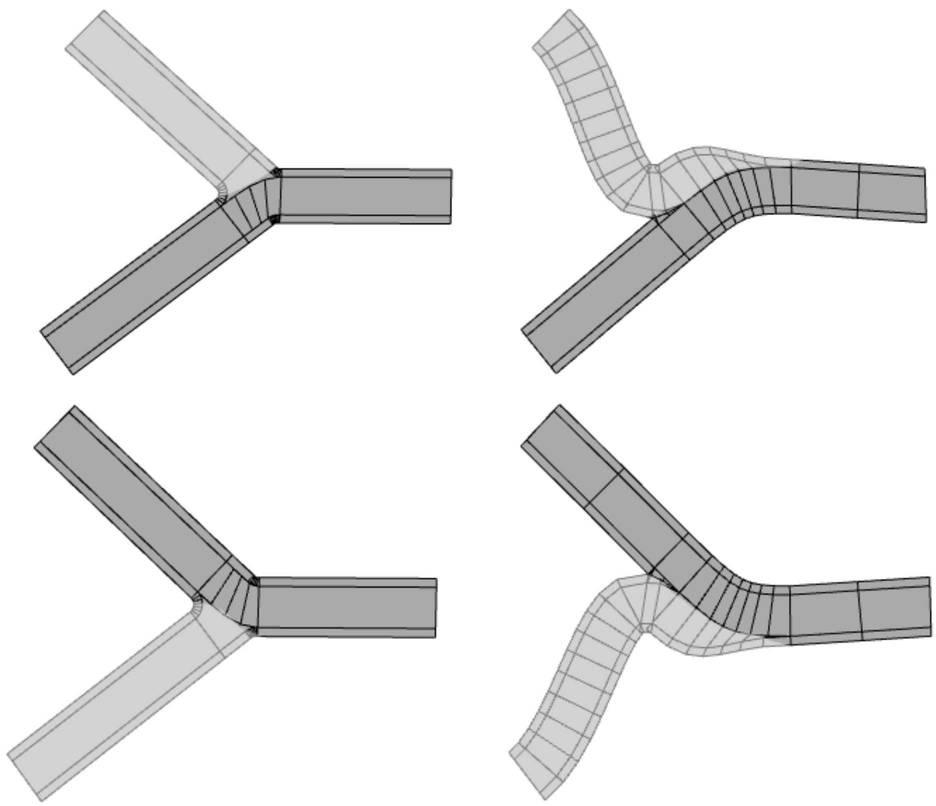
Roundabout Island Radius (attr innerRadius)
Defines the radius of the inner circle (the "island shape").

Roundabout Street Width (attr streetWidth)	
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`.	

Principle street selection

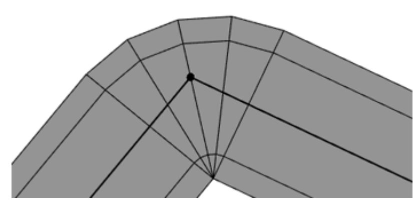
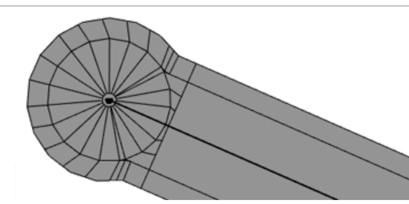
The `Junction` and `Freeway` node types make use of the principle street to determine the node geometry. The principle street does not alter the Smart junction behavior.

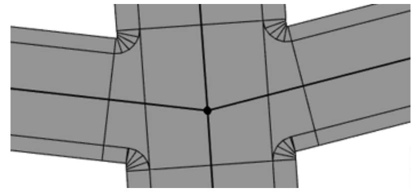


Left: Two junctions with different principle streets. Right: Two freeway intersections with different principle streets. In each case the principle streets are drawn in a darker shade of grey.

Note:
The principle 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 2 graph segments) usually lead to curves or links between the segments. **Note:** 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 `Crossing` or `Junction`.	
Dead end street	
By setting the type of a valence-one node to roundabout, one can model a cul-de-sac.	

<table>
<tr><th colspan="2">Junction</th></tr>
<tr><td>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 major street.</td><td></td></tr>
</table>

Auto-generated connection attributes

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

shapeType	Specifies the type of the node. `Crossing`, `Junction`, `Roundabout` or `Freeway`. `Smart` automatically chooses between these types, see Types.
valency	The number of street segments adjacent to a street node. Valency is added to all node shapes.

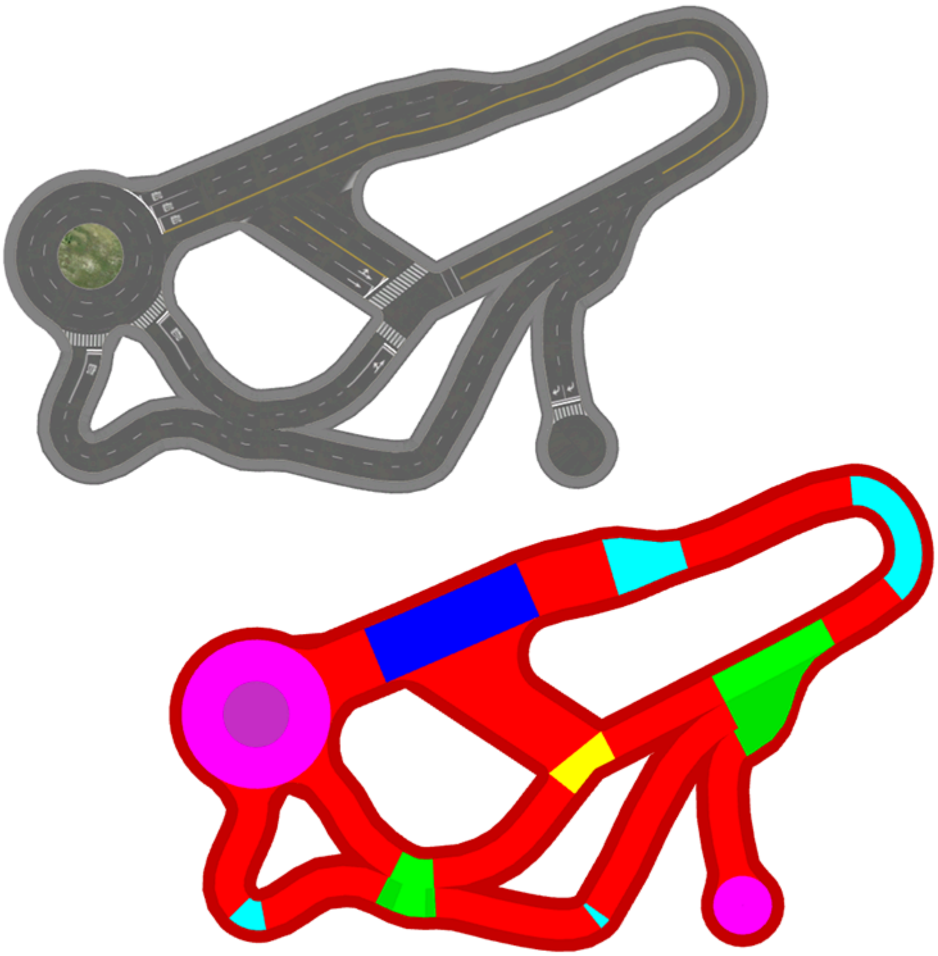
Note:
Connection attributes are object attributes of the graph node and are inherited to the shape, indicated by the italic font in the shape's **Inspector**.

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 2 adjacent streets
	Crossing	Crossing or Smart node type
	Junction	Junction or Smart node type
	Freeway	Freeway or Smart node type

Color	Shape start rule	Created by
	Freeway Entry	Freeway or Smart node type
	Roundabout	Roundabout or Smart node type
	Roundabout Island	Roundabout or Smart 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 UVs.

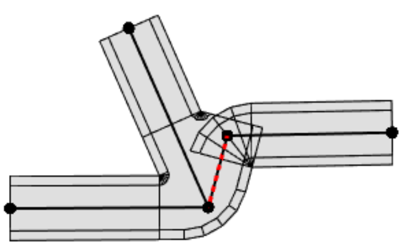
Reset shape attributes

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 execute the tool, select a set of shapes and then choose **Graph** > **Reset Shape Attributes** from the main menu.

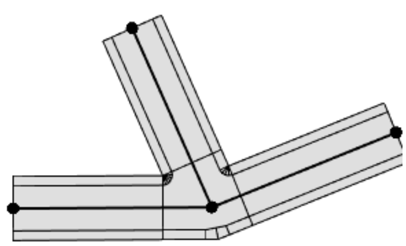
Conflicts

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.*

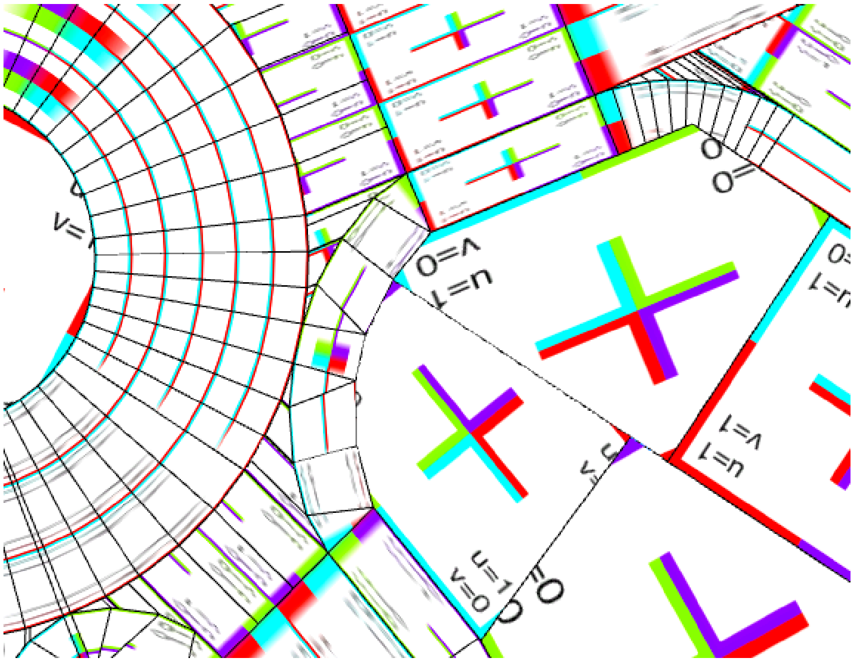
To resolve conflicts, you can choose from the following actions:

- Cleanup the graph, see Graph Cleanup Tool
- Edit the graph network, see Street creation tools
- Change shape creation parameters (i.e. 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.



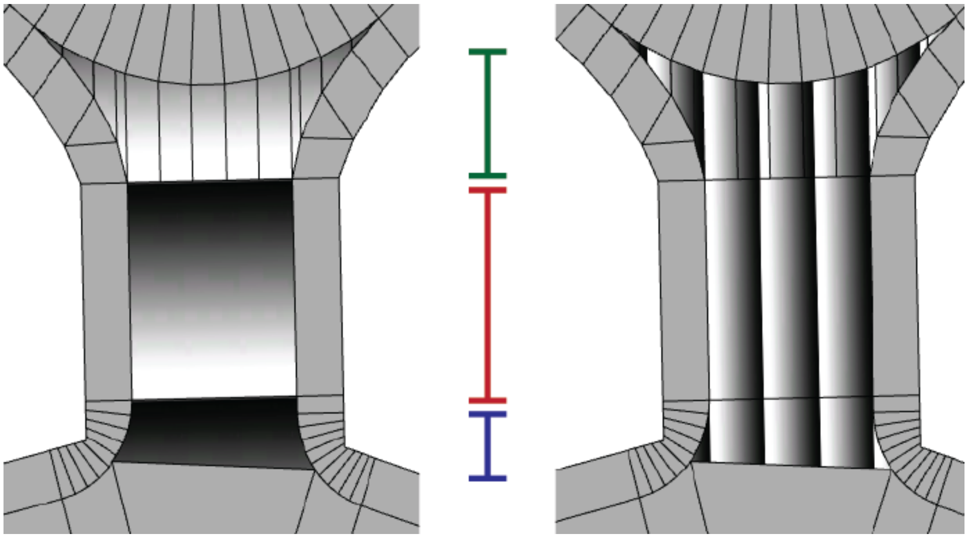
An example of generated UV coordinates for street, intersection and lot shapes

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 below images we see a street shape between a crossing (bottom) and roundabout (top). The left image illustrates the U values using shades of grey, while the right image illustrates the V values.

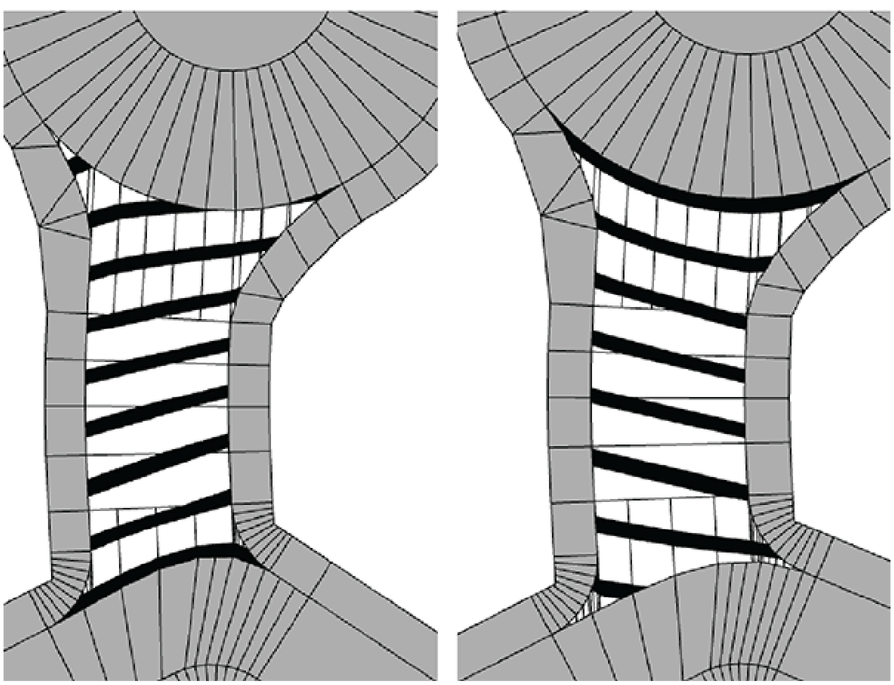


Street UV set 0

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

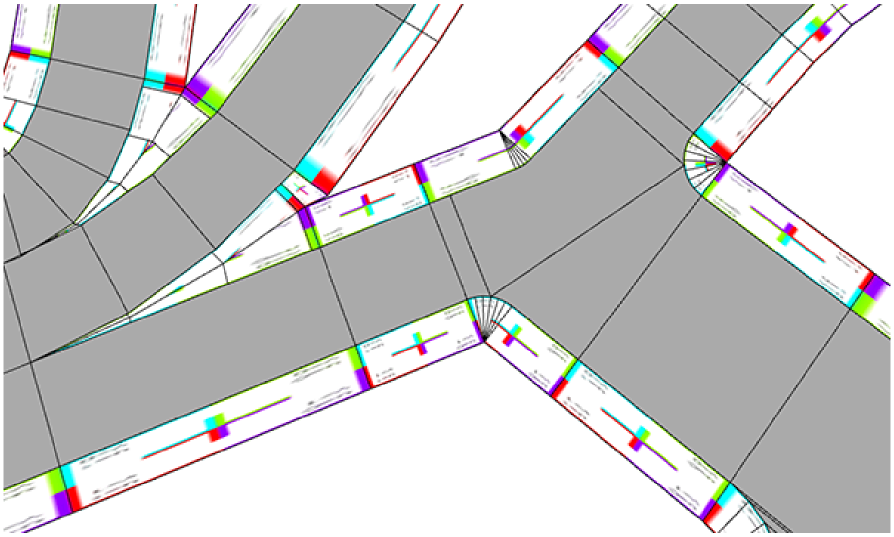
UV Sets 1, and 2 provide distance fields from the start of the entry, and end of the exit respectively. This information is provided in the U channel. The V channel is undefined. In the below image the street direction is from the bottom to the top, and shows the orientation of UV Set 1 (left) and UV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

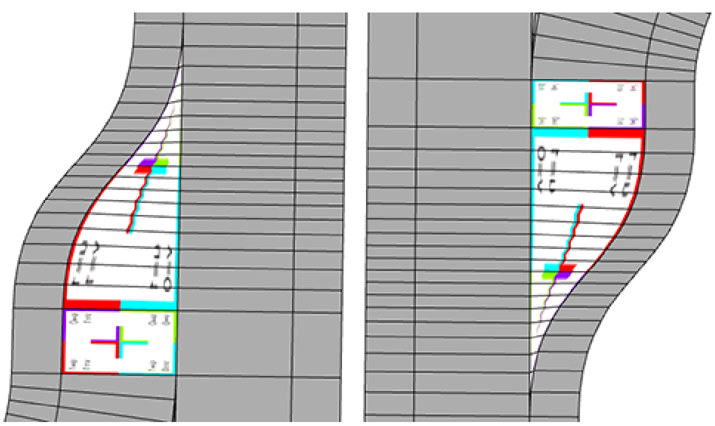
Intersection UVs

Intersection shapes that are street-like use the same UV set 0 parameterization as streets, without entries or exits. These shapes are Joint, Junction, Roudabout, 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

Editing

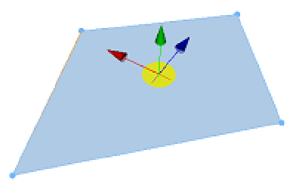
Use transform tools

The transform tools allow you to manipulate and manually adjust your objects by moving, scaling, and rotating.

Move

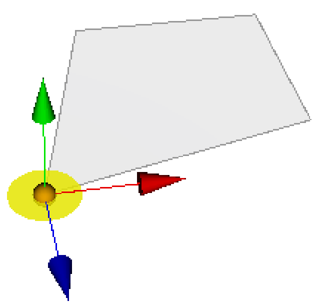
To move objects, click the **Transform Move** tool (`W`). You can also click **Edit** > **Move Tool** in the main menu.

Drag the handles to move along individual axes. Use the yellow handle to move the selection on the xz or object planes. 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.

When you select a vertex, as a single selection or as a lead vertex in a multiselection, an orange sphere displays that allows you to drag the vertex on any plane and snap to faces, vertices, edges, or guides.



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

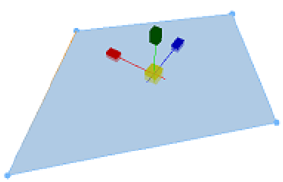
The **Transform Move Tool** includes the following options:

Transform Move Tool	
Reference System	See Reference system.
Adjust the Position and Orientation	See Adjust the Position and Orientation.
Translation (m)	Distance in meters to move objects along the axes of the reference system. • Press `Enter` to apply a value. • Press `Tab` to move to the next input field. • Apply only one value at a time.

Snapping	Turn snapping options on and off. Select objects or components and do one of the following: • Drag the orange sphere to snap to faces, vertices, edges, or guides on any plane. • Drag the yellow disc to snap to vertices, edges, or guides on the xz plane. • Drag the axes handles to snap to vertices, edges, or guides along the selected axis. • Press `Shift` to temporarily turn on or off snapping.

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.

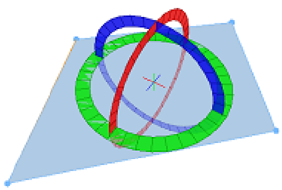
Transform Scale Tool Options

The **Transform Scale Tool** includes the following options:

Transform Scale Tool	
Reference System	See Reference system.
Adjust the Position and Orientation	See Adjust the Position and Orientation.
Transform Individually	When multiple objects are selected, scale individual objects around their centroid with axes defined by the reference system.*
Scale (%)	Scale objects along the axes of the reference system. • Press `Enter` to apply a value. • Press `Tab` to move to the next input field. • Apply only one value at a time. • —Enables uniform scaling along the axes.
Snapping	Turn snapping options on and off. Press `Shift` 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** includes the following options:

Transform Rotate Tool	
Reference System	See Reference system.
Adjust the Position and Orientation	See Adjust the Position and Orientation.
Transform Individually	When multiple objects are selected, rotate individual objects around their centroid in which axes are defined by the reference system.*
Rotation (°)	Rotate objects around the axes of the reference system. • Press `Enter` to apply a value. • Press `Tab` to move to the next input field. • Apply only one value at a time.
Snapping	Turn snapping options on and off. Select objects or components, drag a rotate axis handle, and do one of the following: • Rotate to snap to global axes. • Rotate to snap to planes. • Rotate to snap to parallels of guides. • Press `Shift` to temporarily turn on or off snapping.
Snap to discrete angles	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:

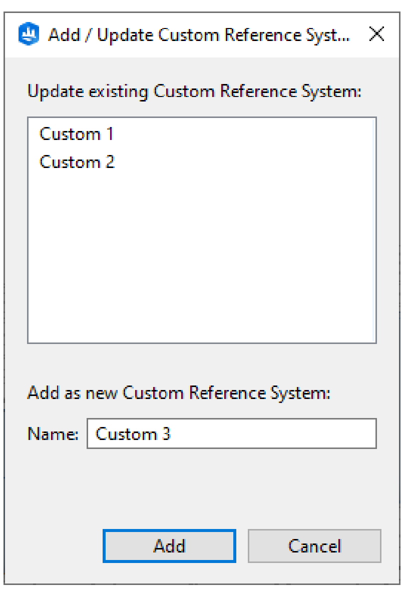
- **World**—Transform tools are aligned to the axes defined by the scene coordinate system.
- **Object**—Transform tools are aligned to the objects' features (edges and normals).
- **{Custom reference system}**—Transform tools are aligned to the axes of a custom reference system that you create and save. See Custom Reference System.

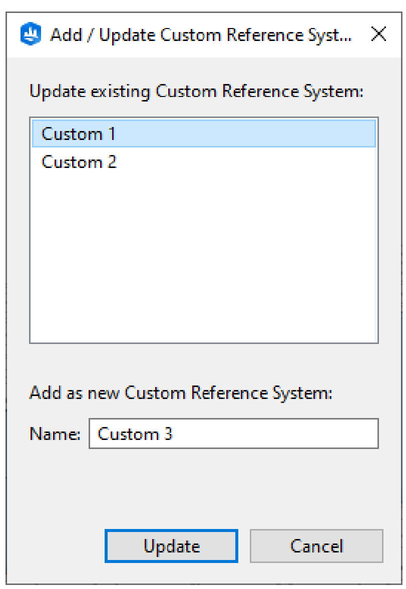
Create a custom reference system

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.

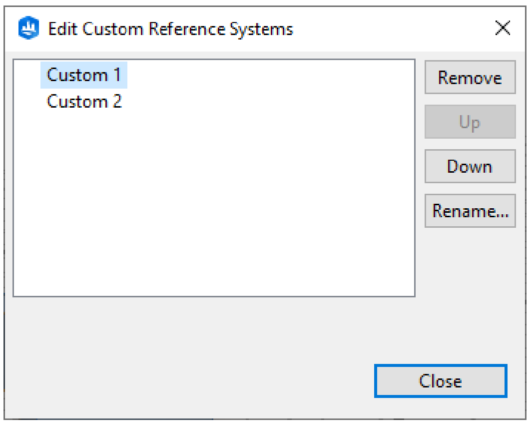
5. Click **Cancel** to close the window without adding or updating CRS.

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.

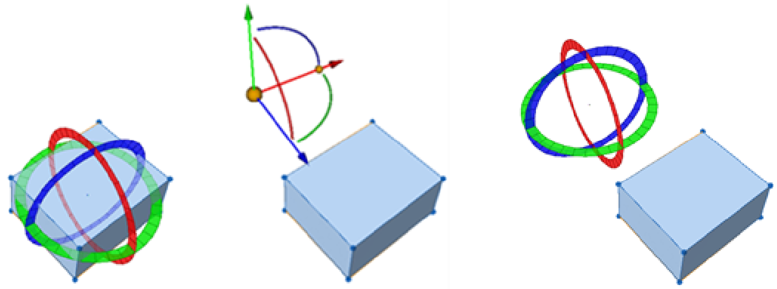


Adjust Position and Orientation

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.



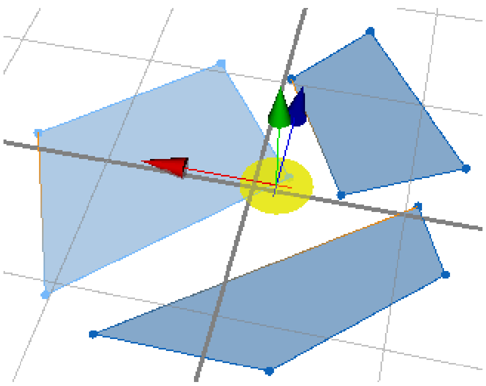
The original rotation tool, moved outside of the shape, and the new tool position are shown.

The tool automatically snaps to vertices, edges, faces, and guides as you reposition the tool.

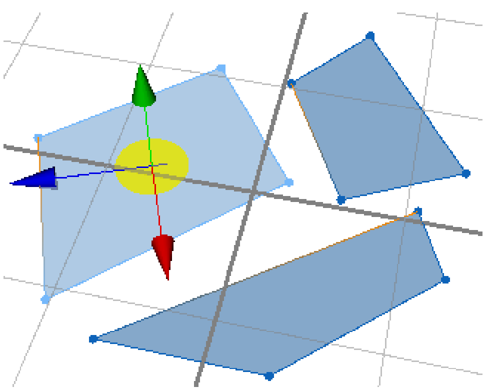
When you deselect, select another shape, or change reference systems in the drop-down menu, the reference system position is reset.

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

Shape edge tools

You can find additional shape tools under **Shapes** in the main menu.

Compute First/Street Edges

This operation automatically calculates the first edges and street width attributes of the selected shapes. The tool finds the nearest street (within 100m) 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).

Compute Edge Attributes

The Compute Edge Attributes tool computes orientation and street adjacency attributes for each edge of a shape.

Set First Edge

This operation sets the first edge of a face to the currently selected edge. This step is often needed to orient a face's "zero" edge towards a street (e.g. for placing the buildings front correctly). If a face is selected, the highlighted gradient line indicates the first edge (with gradient from vertex 0 to vertex 1).

Set Street Edges

This operation marks 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 CGA reference).

Shape geometry tools

You can find additional shape tools under **Shapes** in the main menu.

Subdivide

The **Subdivide** tool computes smaller shapes from selected shape(s). 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

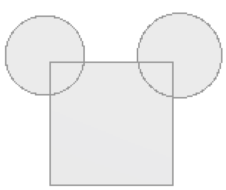
This operation reverses the normals, in other words, orientation, of all selected faces. This step is often necessary after importing shapes with reversed orientation.

Separate Faces

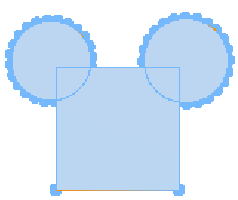
You can select a shape or multiple shapes and separate individual shapes for every face. Click **Shapes** > **Separate Faces** in the main menu. All new shapes are created into the existing shape layer.

Combine Shapes

You can combine all the faces of the selected shapes into a single shape. Click **Shapes** > **Combine Shapes** in the main menu.



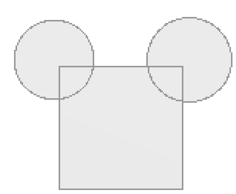
Shapes to combine

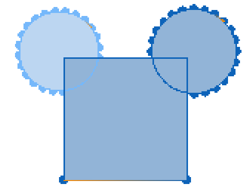
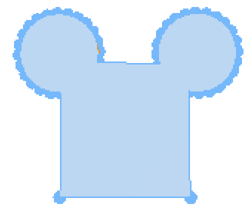


Multiple shapes combined

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.	

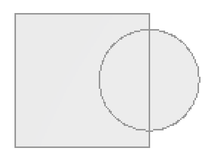
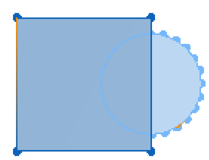
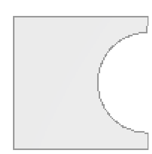
Selected shapes.	
Multiple shapes united into one shape.	

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

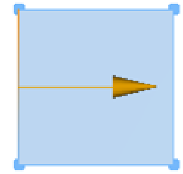
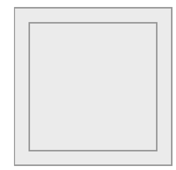
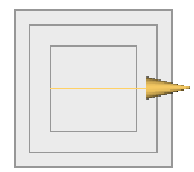
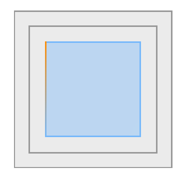
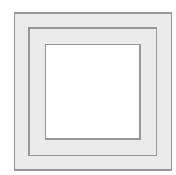
The **Subtract Shapes** tool performs a Boolean subtraction of the lead-selection shape from all other selected shapes. Click **Shapes** > **Subtract Shapes** in the main menu to subtract the lead-selection from the other shape. Press `Shift` while selecting a different shape to change the lead-selection.

Overlapping shapes.	
Selected shapes with the circle as lead-selection.	
Square shape after deleting the circle.	

Note:
This tool works only with planar shapes.

Offset Shapes

Offset Shapes tool creates a new 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.

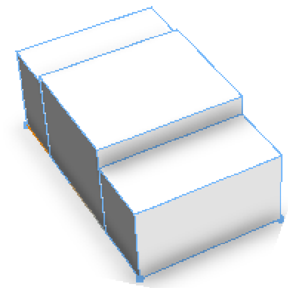
Offset Shapes	
Distance (m)	The offset distance in meters. You can enter a value and press `Enter` to apply.

Remove Holes

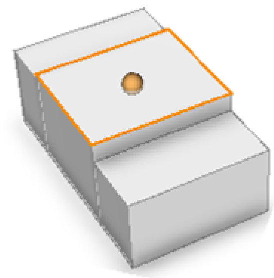
Removes all holes from the selected shapes.

Convert Models to Shapes

Use this tool to convert CGA models to shapes for manually editing.



CGA model



CGA model converted to shape

Note:
After this conversion, changes in attributes and CGA rules do not affect the shapes anymore.

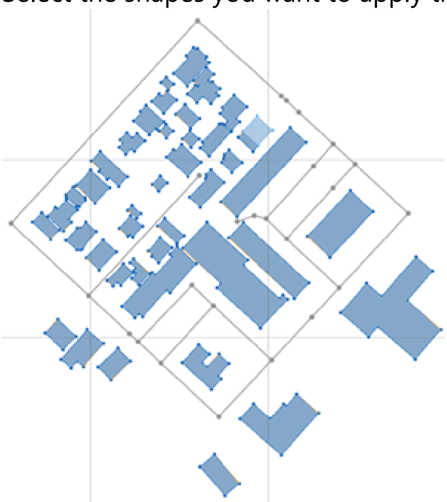
Compute edge attributes tool

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.

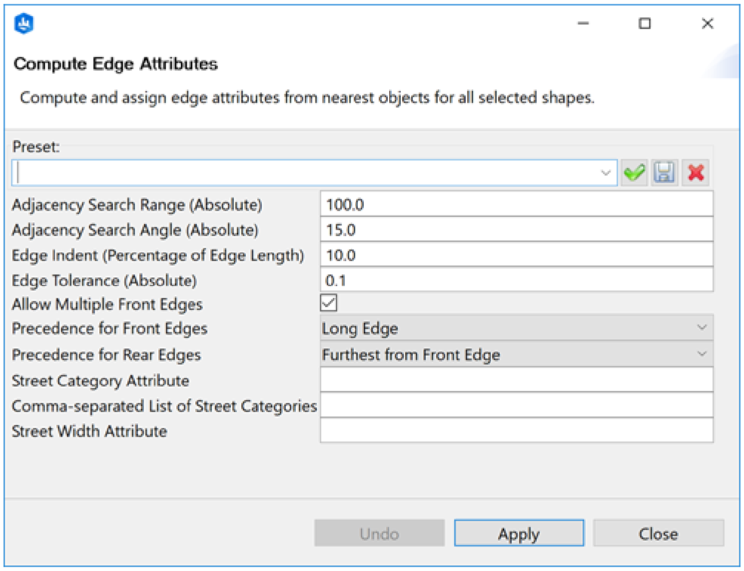
   

   Selected shapes

   The tool will compute the edge attributes for every selected shape with respect to the visible street segments, i.e. it is not necessary to select the street segments.

   Note:
   The algorithm currently only works for static shapes (see shapes). If you want 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**.
   This opens the tool dialog which provides numerous parameters to configure the algorithm. The algorithm and the parameters are explained in detail below. For the general use case, it is okay to use the default parameters and just press **Apply** to execute the algorithm.



Compute Edge Attributes dialog box

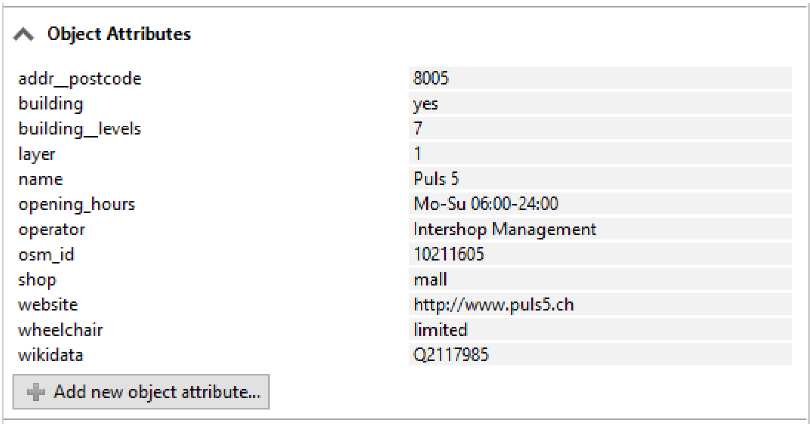
Inspect the computed edge attributes

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 selected shape to compute the attributes for:

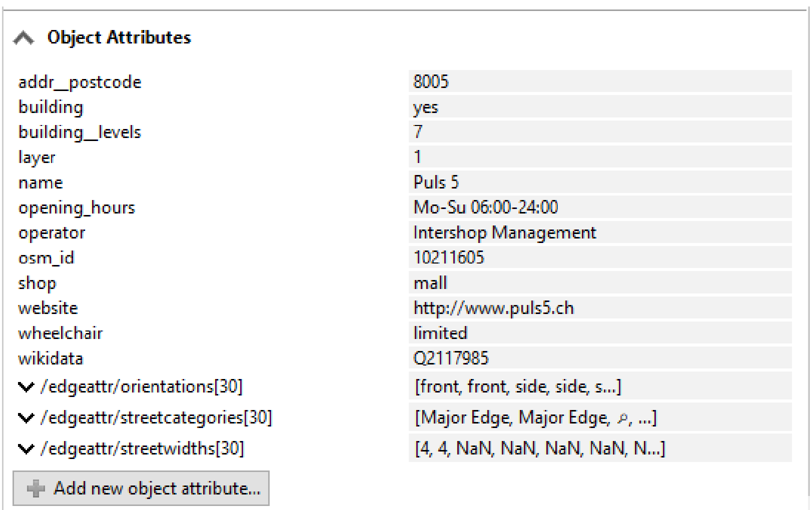
Selected shape for computing attributes

Shape attributes before running the tool:



Attributes before computing edge attributes

Shape attributes after running the tool:



Attributes after computing edge attributes

As shown, the tool adds three shape attributes which contain a list of values for each edge, starting with the edge at index 0:

- `/edgeattr/orientations`—enumeration[]
  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[]
  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[]
  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

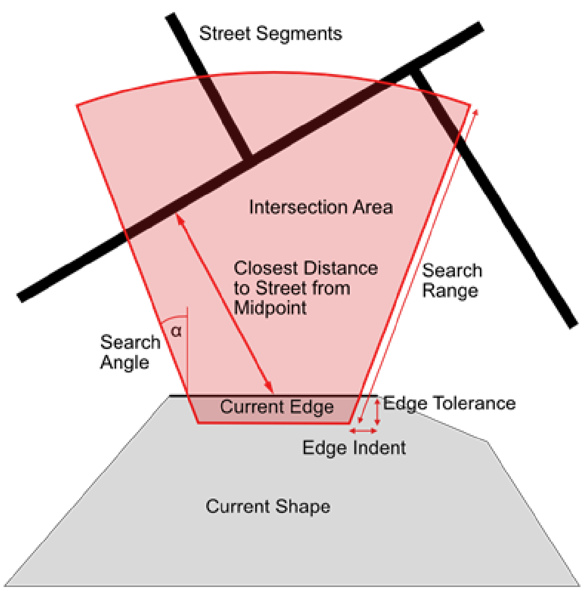
The computed edge attributes can be accessed from CGA using the edge attribute functions.

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

- `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 towards inside of a shape for increased tolerance to correctly handle overlapping shapes (default value: 0.1).

Computation of edge orientation

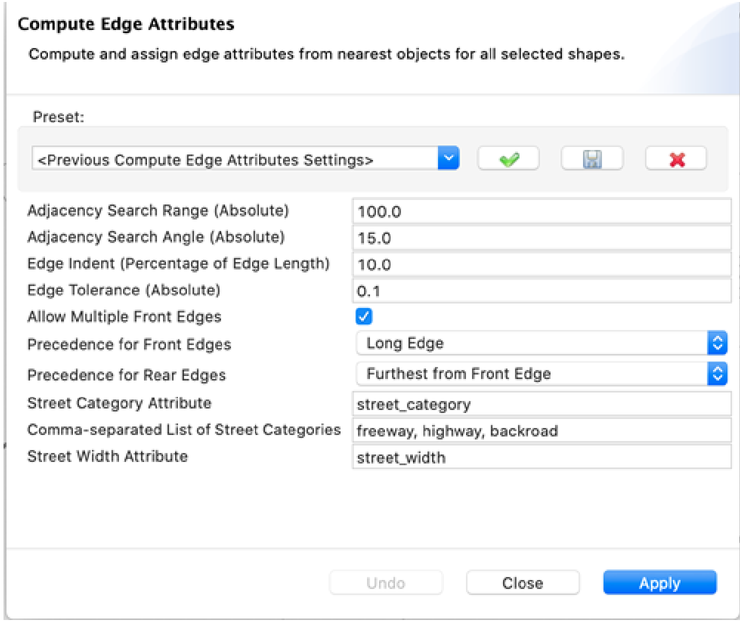
First, the front edge (or the front edges if **Allow Multiple Front Edges** is selected) is determined:

- The algorithm iterates over all edges and selects those facing a street.
- Of those facing a street, the one(s) with the lowest street category is/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 furthest 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 in case 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 / Minor attribute is used.
- `Comma-separated List of Street Categories:` This allows the user to define an ordered list of street categories which are used for sorting and determine the front edges. For example, assume that your street network contains an attribute "street_category" with values "freeway", "highway", "backroad". Then you would set the street category attribute to "category" and the street category list to "freeway, highway, backroad" to make sure 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 needs to be a `string` type. If left empty, the built-in street width is used.



Set the algorithm parameters

Edit terrain

Interactive terrain editing

The **Terrain Edit Brush** and **Terrain Reset Brush** tools allow you to interactively change the elevation of the terrain in your 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 includes the following options:

Terrain Edit Brush	
Brush Size (m)	The brush size in meters defining the radius in which the terrain height is adjusted.
Elevation Picker	Allows you to pick the terrain height at the location of your mouse by clicking a terrain in the Viewport. Press `(y)` to turn the elevation picker toggle on and off.
Height (m)	The height in meters in which the terrain is adjusted.
Smooth Borders	Applies a smooth transition method at the brush border between the edited and the original terrain. • **None**: No smoothing is applied. • **Smooth Range**: 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 **Range**). • **Constant Gradient**: Applies a slope with a constant up or down gradient defined in degrees (see **Gradient**).
Range (m)	• Radial range in meters in which to apply smooth transition. • Available for the **Smooth Range** and the **Constant Gradient** methods.
Gradient (°)	• The gradient in degrees of the border slope. • Available for the **Constant Gradient** method.
Easing	• Applies a non-linear smoothing in order to avoid sharp edges at the inner and outer fronts of the border region. • When the toggle key is off, the smoothing is applied linearly. • Available for the **Smooth Range** method.

Reset 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 **Terrain Reset Brush** tool options.

Terrain Reset Brush Tool Options

The **Terrain Reset Brush** tool includes the following options:

Terrain Reset Brush	
Brush Size	The brush size in meters defining the radius within which the terrain is reset.
Smooth Borders	Applies a smooth transition method at the brush border between the edited and the original terrain. • **None**: No smoothing is applied. • **Smooth Range**: 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 **Range**). • **Constant Gradient**: Applies a slope with a constant up or down gradient defined in degrees (see **Gradient**).
Range (m)	• Radial range in meters in which to apply smooth transition. • Available for the **Smooth Range** and the **Constant Gradient** methods.
Gradient (°)	• The gradient in degrees of the border slope. • Available for the **Constant Gradient** method.
Easing	• Applies a non-linear smoothing in order to avoid sharp edges at the inner and outer fronts of the border region. • When the toggle key is off, the smoothing is applied linearly. • Available for the **Smooth Range** method.

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

Terrain	Choose which terrain to reset or select **All**.
Constraint	• **Everywhere**: All terrains are completely reset. • **Inside selected shapes only**: Only the terrain vertices intersecting the currently selected shapes are reset.
Add border	If enabled, a small border region around the shapes is reset, too. Only meaningful if **Constraint** is set to **Inside selected shapes only**.

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:

Terrain	The terrain which should be aligned or All.
Raise terrain	If enabled, terrain vertices below selected shapes are aligned.
Maximal raise distance	If distance between terrain and shape is smaller, terrain vertices below shape are raised.
Lower terrain	If enabled, terrain vertices above selected shapes are aligned.
Maximal lower distance	If distance between terrain and shape is smaller, terrain vertices above shape are lowered.
Add border	If enabled, a small border region around the shapes is aligned, too.
Write cut/fill volumes to attributes	If enabled, for each selected shape cut/fill volumes in cubic meters are approximately calculated. The values are written into the object attributes (fields "cutVolume" and "fillVolume").
Smooth borders	Applies a smooth transition method at the border between the edited and the original terrain. • **None**: No smoothing is applied. • **Smooth range**: Applies smoothing to the terrain in which the relative height difference at the border is interpolated within the specified range defined in meters (see **Border range**). • **Constant gradient**: Applies a slope with a constant up or down gradient defined in degrees (see **Border gradient**).
Border range (m)	• Radial range in meters in which to apply smooth transition. • Available for the **Smooth range** and the **Constant gradient** methods.
Border gradient (°)	• The gradient in degrees of the border slope. • Available for the **Constant gradient** method.

Border easing	• Applies a non-linear smoothing in order to avoid sharp edges at the inner and outer fronts of the border region. • When the toggle key is off, the smoothing is applied linearly. • Available for the **Smooth range** method.

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 (e.g. 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 overview

CGA stands for "Computer-generated Architecture". It is a programming language used to instruct CityEngine how to automatically generate 3D geometry and, especially, architecture.

The basic elements are:

Rule — The building Instructions are called rules. Each rule corresponds to one instruction or transformation step. When executed, each rule takes geometry as input and alters and/or replaces it to generate an output. This output becomes the input of a following rule. The first rule that is executed is the **Start Rule** and 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, simply drag and drop a CGA file onto the shape in the **Viewport**.

Generation — The model creation is launched by clicking the Generate icon in the toolbar or press `Ctrl + G`. The start rule is executed and the resulting 3D geometry (model) is shown in the **Viewport**.

Note:
To better understand the basic elements in CGA modeling check out the section "Model Generation with Rules of Tutorial 1: Essential Skills.

CGA and manual modeling

Procedural techniques are applied in many areas of 3D modeling. These approaches proved to be very efficient and time-saving when large amounts 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 interesting use case. By instructing a CGA rule to calculate sizes and dimensions of a model during generation, KPIs (Key Performance Indicator) 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 elaborate design decisions.

Manual modeling, on the other hand, is fast when a one-of-a-kind model is needed and a visual representation is only 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 the CGA generation by interactive tools.

Work with rules

The tutorials and example projects you can download within the CityEngine main menu (**Help** > **Download Tutorials and Examples...**) as well as **ESRI.lib** are full of versatile rules that you can use in your projects as is or as starting points for your own customized rules. To get started with 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**.
2. Click **New** > **CityEngine** > **CGA Rule File...**.
   Alternatively, right-click in the **Navigator** the rule folder 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**. 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 auto-generated meta 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 has originally been 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 put in front of each parameter definition. This way, these values become editable attributes in the **Inspector**. 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 a 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 we defined before.

Note:
- Press `Ctrl + S`, and select **File** > **Save** or right-click in the **CGA Editor** and select **Save** to save your CGA file. You must save your changes in order to take effect for the model generation.
- To open CGA files, double-click the file in the **Navigator** or select **File** > **Open** in the main menu.

Learn more about the CGA Editor.

Assigning rules and generating models

1. Create a couple of 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** the fields "Rule File" and "Start Rule" are filled out now.
   - The attributes `maxHeight` and `minHeight` appear in the **Inspector**.
5. Click **Generate models** tool (`Ctrl+G`) Generate .

   In the **Viewport** you now see extrusions to different heights.

Select some of the shapes and change one of the attributes in the **Inspector**. This automatically triggers a regeneration of the selected shapes.

Note:
The changes in the **Inspector** 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 such as when closing and reopening the scene. To "shuffle the dice" and get new random values you need to set a new seed on the shapes. You can do this by clicking the **Update Seed** tool (`Ctrl+Shift+G`) in the toolbar.

Learn more about CGA in Tutorial 6: Basic shape grammar.

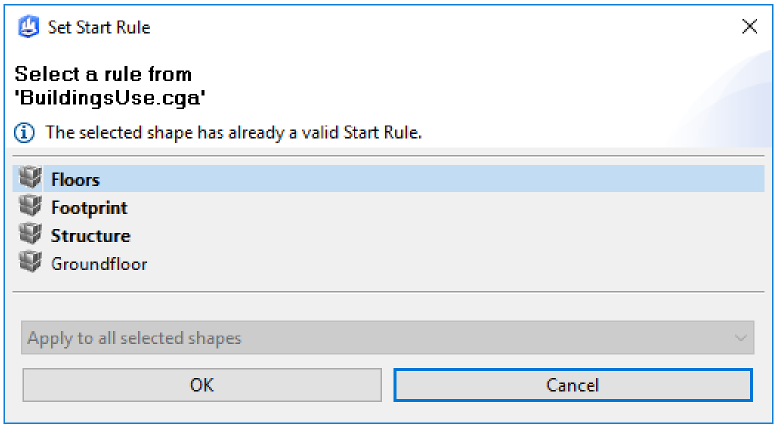
Setting the Start Rule

In addition to the rule file, a shape requires are valid start rule to trigger the model generation. If no valid start rule is found while a rule file is being assigned, the **Start Rule** dialog opens.

A Start Rule is required for rule file model generation. The start rule can also be manually typed in the **Inspector** or set from using the **Start Rule** dialog by clicking **Select...** in the **Inspector** .

Set Start Rule dialog

The Start Rule Dialog displays all rules that can be applied as start rules. Rules that are marked as start rules are displayed in bold in the dialog below (see also CGA annotations):



Setting the start rule

Select the rule that you want to assign as start rule. Additionally, you can choose the following:

- **Apply to all selected shapes**: Set selected start rule to all shapes.
- **Apply only to shapes with invalid Start Rule**: Set selected start rule to all shapes with invalid or empty start rule. This option is only active if applicable.

Note:
CityEngine tries 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 detail.

Default Start Rules

- Shapes generated inside 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, ...). These start rules can be reset to their initial value with **Graph** > **Reset Shape Attributes**
- Shapes which are imported from a .obj file (arbitrary geometries) have their start rule set to the obj group name by default.

CGA essentials

Coordinate systems

3D coordinate systems are used to describe the position and orientation of objects in space. While CityEngine can handle many different types of (georeferenced) coordinate systems, CGA modeling works with Cartesian coordinate systems only. A coordinate system is defined by its origin and the orientation of the three orthogonal axis. While one global coordinate system would be 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.

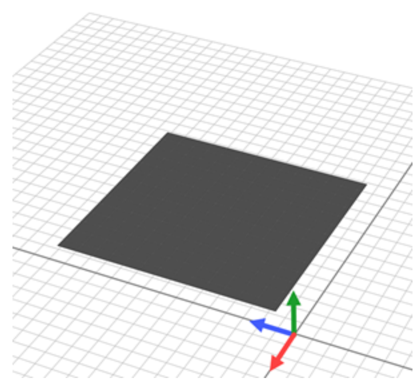
Think of a car model placed on a street segment. It makes sense to define the position of the wheels relative to the car and the car relative to the street. The origin and the orientation of the "car" system are defined with coordinates of the "street" system. This way the two coordinate systems are relative to each other and the coordinates of one system can be translated into coordinates of the other one.

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.

There are multiple different coordinate systems relative to each other in CGA:

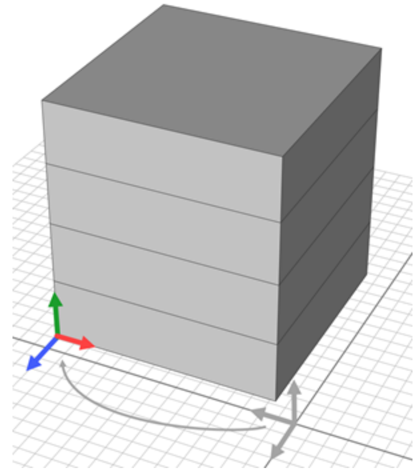
<table>
<tr><th colspan="2">World Coordinate System</th></tr>
<tr><td>

Is defined per scene.
The world coordinate system is global, there can be only one per scene.
Can be georeferenced, but doesn't have to be.
The position of initial shapes is defined in world coordinates.

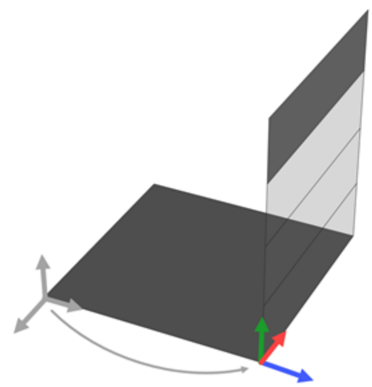
</td><td></td></tr>
<tr><th colspan="2">Object Coordinate System</th></tr>
</table>

<table>
<tr><td>

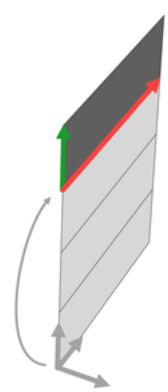
- Local coordinate system defined for each initial shape.
- The origin is placed at the first point of the initial shape's first edge.
- The axes are oriented such that the x-axis is directed along the first edge, the y-axis is directed along the first face's normal and the z-axis is perpendicular to the former two.

</td><td></td></tr>
<tr><td colspan="2">Pivot Coordinate System</td></tr>
<tr><td>

- The pivot system is described in object coordinates.
- In many cases the pivot origin and orientation is identical with the object coordinates. Still it comes into play e.g. with component splits.

</td><td></td></tr>
<tr><td colspan="2">Scope Coordinate System</td></tr>
<tr><td>

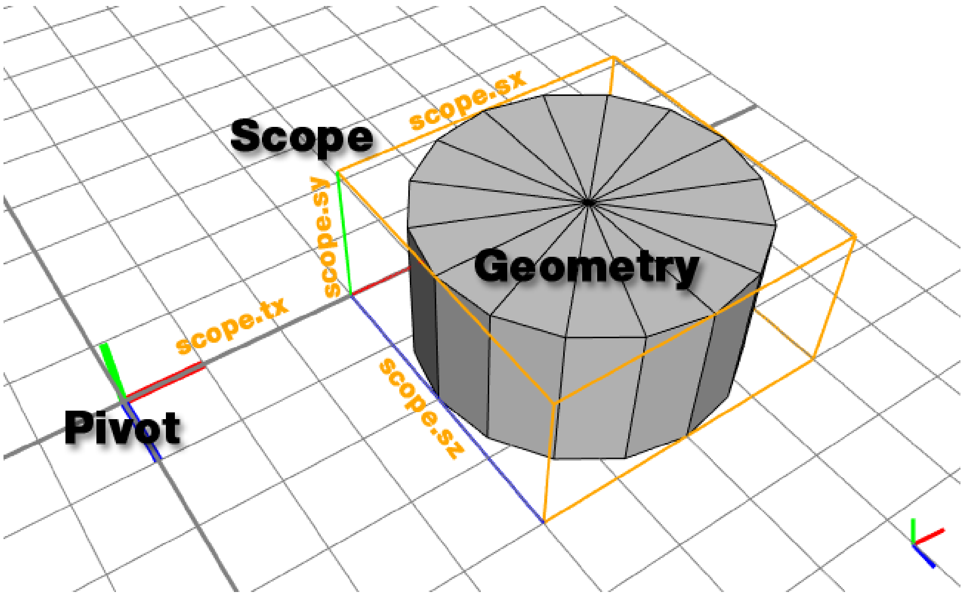
- Think of the scope as a bounding box that is placed and oriented in the pivot system.
- In addition to its characteristics of a coordinate system, the scope has also a size. The size is defined by a width, height and depth.
- Using transformations you can define the position, orientation and size of the scope.

</td><td></td></tr>
</table>

Note:
- When developing CGA rules you can visualize the coordinate systems and shapes of a model using the Model Heirarchy.
- You can query the pivot and scope attribute attributes.

CGA shapes

CGA Shapes are the central ingredient 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:

Shape Symbol	The name of the shape. Used to find the matching rule that is used to generate the successive shapes.
Pivot	The pivot describes the shape's coordinate system and is defined by: • a position vector p (`pivot.px`, `pivot.py` `pivot.pz`) • an orientation vector o (`pivot.ox`, `pivot.oy` and `pivot.oz`. The pivot is given in object coordinates, relative to the initial shape's origin; see Coordinate Systems.

Scope	The scope represents the oriented bounding box for the shape in space relative to the pivot and is defined by three vectors: • Translation vector t (`scope.tx`, `scope.ty` and `scope.tz`) • Rotation vector r (`scope.rx`, `scope.ry` and `scope.rz`) • Size vector s (`scope.sx`, `scope.sy` and `scope.sz`)
Geometry	The geometry contains the information about the corners edges and faces that make up the "form" of the shape. The geometry can be any sort of polygonal mesh. In addition, information about color, material and textures (shader attributes) are stored in the geometry as well.
Parameters	Each shape can have an associated parameter list. The ordered parameter list is implicitly defined in the rule which creates the shape. Three parameter types are supported: • Boolean • Numeric (internally represented with double-precision float) • String

Rule application

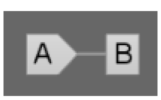
The basic idea of a rule is to replace a shape with a certain shape symbol with a number of new shapes. Formally:

```
PredecessorShape --> Successor
```

Here is a simple example:

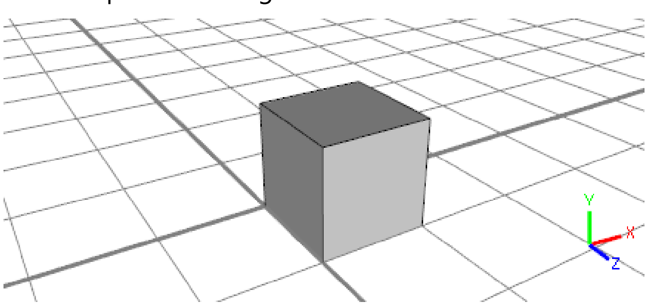
```
A --> B
```

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 done and not processed anymore. If there is no rule matching symbol B the generation process is finished. The resulting structure is called the `shape tree` and looks like this:



Shape tree

In the shape tree above, A is the root shape and B is a leaf shape. Leaves are very important because the sum of all leaves represents the generated model. Inner nodes are not visible in the final model.



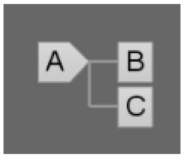
Generated model

In this simple example, we assume shape A's geometry, scope and pivot are set up such that the shape represents a unit cube in the origin; because B is a copy of A, B looks exactly the same (see the picture above).

A rule can have more complex successors, e.g., the right side of the rule can consist of multiple shape symbols and shape operations:

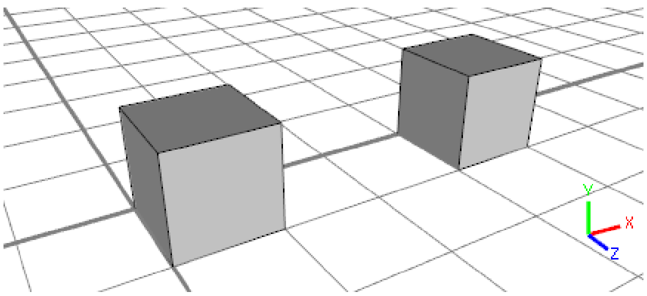
```
A --> B t(3, 0, 0) C
```

This successor is now executed from the left to the right. Again, B is an identical copy of A. Then the current shape is translated by 3 units in x-direction (i.e., the `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:

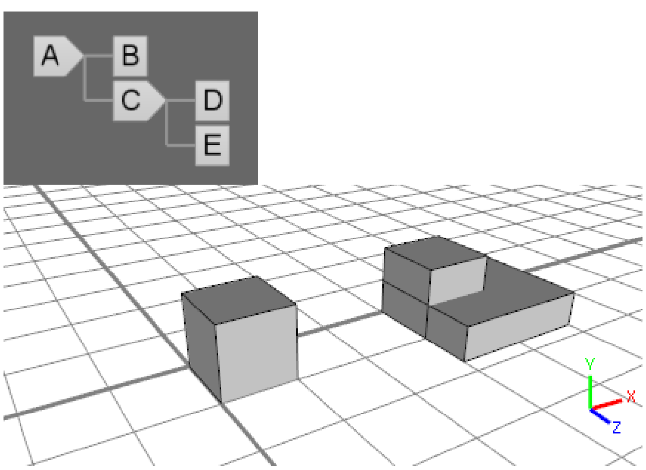


Leaves B and C make up final model

If we add this rule for C:

```
C --> D s(2, 0.5, 1.75) E
```

The generation process will add 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:



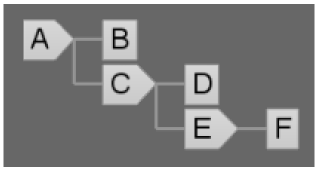
Shape tree and model with children D and E added

Note:
Now, the leaves (B, D, E) are not on the same level (i.e. 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
```

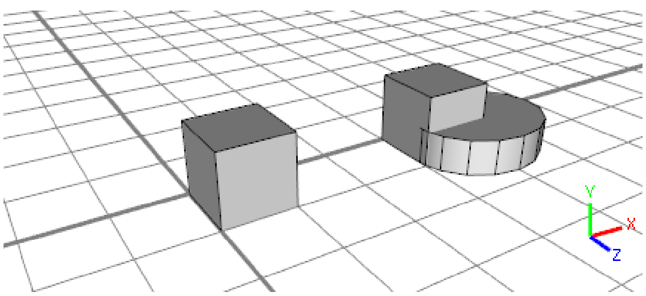
After starting the generation again, shape E is not a leaf anymore but now has a child shape F.



Shape E with child shape F

The geometry of shape F is not a cube anymore but was replaced with the mesh read from file "cylinder.obj".

Note:
The size (i.e. the `scope.s` vector) of shape F is still the same as the one 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 rule F is defined, the generation is stopped at this point. However, the CGA editor will issue a "Undefined Rule" warning. This can be suppressed by adding a period after F, thus explicitly marking F as a terminal shape:

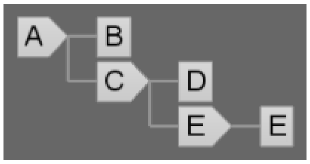
```
E --> i("cylinder.obj") F.
```

Anonymous Leaf Shapes

For convenience, rules like the E rule above can be truncated:

```
E --> i("cylinder.obj")
```

In this case (i.e. 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 overview

This section is about CGA Features and covers typical constructs, CGA syntax and handy features. CGA Operations are covered in Shape operations overview.

Rule with parameters	Pass multiple strings, numbers or boolean values from a rule to its successors.
Conditional rule	Call different successor rules based on conditions.
Stochastic rule	Call different successor rules at random. The likelihood of each successor can be controlled by percentages.
CGA attributes	Global variables used to store values in multiple rules. They can be modified in the rule file or the **Inspector**.
CGA styles	A style is a specific combination of values for a set of attributes. You can switch between styles in the **Inspector**.
CGA functions	Encapsulate code so that it can be used by multiple rules. Helps to avoid duplicated code.
Local variables	Store intermediate values in local variables.
Rule file import	Import rule files to get access to all of its rules, functions, and attributes.
Comments	Make your code easier to understand for your peer programmers (or yourself in the future).

Rule with parameters

Rules can be parameterized, i.e. 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)
```

During execution of rule `Lot`, a new shape with shape symbol `Footprint` and float parameter "20" is generated. Also, the `height` parameter will have the value 20.

Note:
Arbitrary mathematical expressions can be built 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` as well as the `Floor` rule 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 makes sure that only the matching one is used during shape creation (i.e. during execution of the `Lot` rule above, 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 rule `Footprint` takes one parameter, `type`, of type string. If the string is equal to "residental", the first successor is taken (i.e. 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 (i.e. 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 "do not generate a shape").
- 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 and the statements must build a consecutive block and can not be interrupted with successors (like a block, and they are very different than `if` statements in well-known programming languages).

The `case` and the `else` statements must build a consecutive block and can not be interrupted with successors. They are similar to `switch-case` blocks and different from if statements in well-known programming languages.

Stochastic rule

In analogous manner to conditional rules, the CGA Shape grammar permits stochastic rules, i.e. 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% chance the first successor is chosen and a new `Lot` shape with parameter "residential" is generated, a 20% chance for the second successor and a 50% 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 always needs to 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 individually be changed on a per shape basis.

This can be done in the **Rules** pane in the **Inspector**.

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)
```

Here, 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"
```

Here, there is a 50% chance `landuse` evaluates to "residential" and a 50% chance it evaluates to "industrial". For each shape the conditional/stochastic attributes are evaluated once and stay constant during the generation process.

In the same way, the `rand()` function can be used:

```
attr height = rand(30,50)

Lot --> extrude(height) Envelope

Envelope -->
    case height < 40 : SmallBuilding
    else: LargeBuilding
```

For each shape which has the start rule `Lot`, `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 Attributes, sources, and connections for details how to use different sources to control input of rule attributes.
- The display of attributes in the Inspector can be controlled by CGA annotations.
- Interactive handles can be used to edit attributes in the 3D view.

Change attributes in Inspector

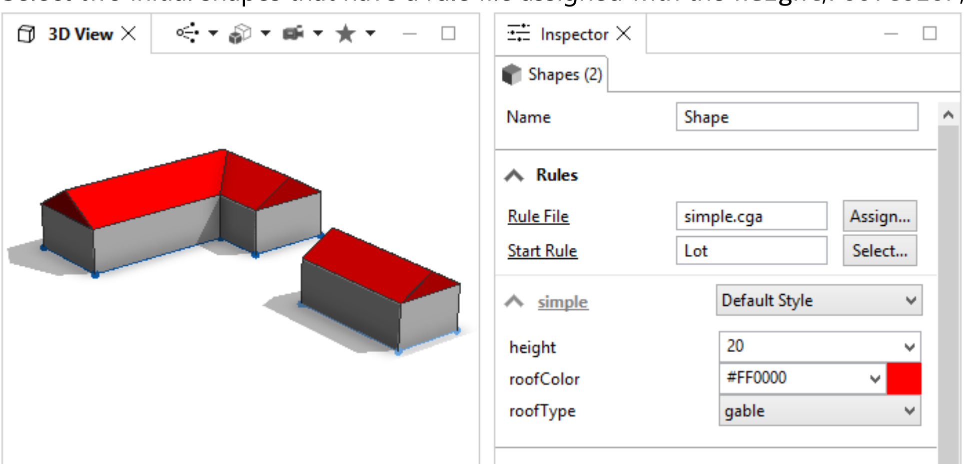
The values of functions marked with attr can be set individually for each initial shape in the **Inspector**. For instance, the definition below yields an entry in the **Rule Parameters** section.

```
@Range(min=10, max=40)
attr height = 20
```

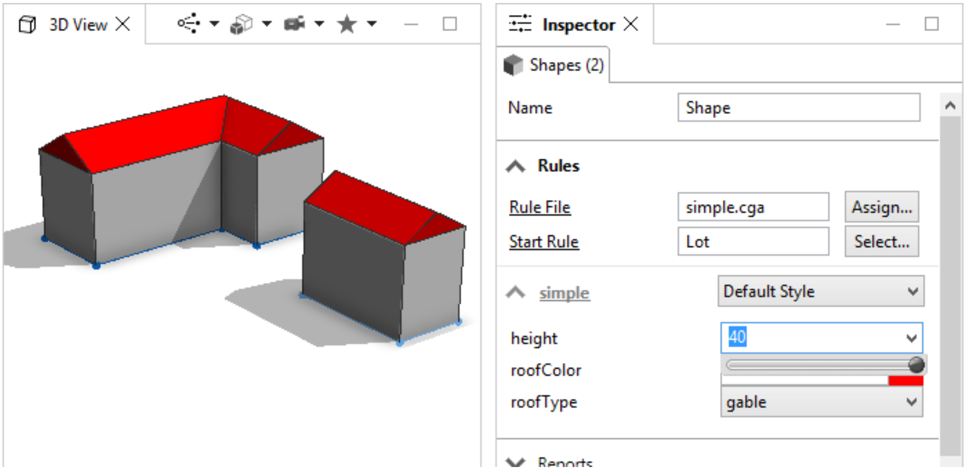
Multi assign

It is possible to manually assign a value to multiple initial shapes in a convenient way. The following is an example of assigning 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.
   Both selected shapes have their height value changed.



3. Select one shape (or model) only, and change the `height` back to the rule default 20.
4. Select both models.
   The value of `height` is displayed as a question mark **?**, stating 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 helps you to save, manage and retrieve favourite settings for a given rule.

Create a new style

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**.
3. Open the **Default Style** drop-down menu and click **Add new Style**.
   This opens the **Create and apply new style** dialog with the following options:

Style name	Use a unique name for your style. An existing style with the same name will be overwritten.
Based on	The reference style which the new style is based on. This can either be the Default Style or the current style (if one is applied).
Description	An optional description for the style.

4. Click **OK**.
   - A new style has been created and it is applied to the selected shape.
   - The new style has been added to the CGA file.

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 in the **Rule Tab** of the **Inspector**.
3. Select **Preview & select styles....**.
   The **Style Manager** 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 cross to delete a style.

Toolbar options

- **Tiles view/Scroll view**: Display CGA styles as tiles or list.

- **Without user attributes/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, 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.
- Instead of using the wizard styles can also be added and modified in the **CGA Editor**, 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`, which was set to 1.2 in the Default Style is overwritten to 2.2 in the style `Facade_Wide`.

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
```

A style can be extended from an existing style. An extended style will inherit all definitions from its parent style. In the example above,

- style `Facade_Wide_2` inherits `Door_Height` from its parent `Facade_Wide` (value results in 2.8).
- style `Facade_Wide_3`, which does not extend a parent, will inherit its value for Door_Height from the default style (resutling in 2.5).

Note:
All styles, if extended or not, implicitly extend the default style.

CGA functions

Functions are used to encapsulate evaluations which are used several times in the rules. Unlike rules, functions are typed (i.e. 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 which uses `getHeight` might look like this:

```
Lot --> extrude(getHeight(geometry.area)) Envelope
```

Note:
In contrast to attributes, functions are evaluated in every call. This means that a function such as

```
height = rand(30, 50)
```

makes sense only for dedicated purposes because it returns a different value every time it is used.

Const functions

Functions can be made constant with the `const` keyword. Const functions behave the same as `attrs`, the only difference is that const functions are internal to the rule file and can not be mapped in the Inspector.

Local variables

Local variables efficiently store values of any type. Within the scope of a rule or a function they can be used 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 this example we extend the rule `Lot` introduced in the previous section CGA functions.

```
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 details about the comp operation). If the height is less than 35 the roof angle is set to 45 degrees, otherwise it is set to 10 degrees. We 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 we 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

Beside rules, local variables can also be defined 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

A sequence of local variables can be defined. An expression may reference a local variable which was defined prior in the list. This makes local variables an easy 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 given prefix, see CGA Reference.

The following rule files both define a `Facade` rule, each one is texturing 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 `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 attribute `height` in structure.cga is overwritten by the attribute `height` in main.cga. Therefore, a building of `height` 10 is created.

Note:
If structure.cga was used standalone, a building of `height` 5 would 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

Comments can be added to CGA source code by either line comments with the characters // or #:

```
// a comment
# another comment
```

or block comments with /* ...... */

```
/* block comments
   can be used to write
   multi-line comments
*/
```

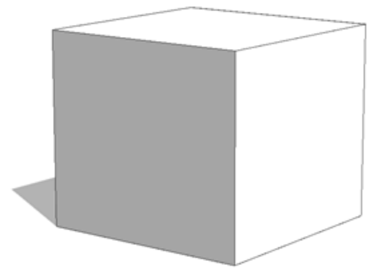
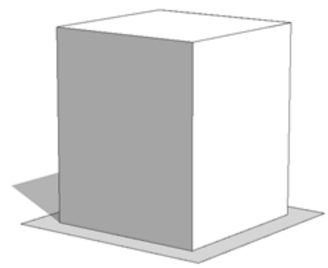
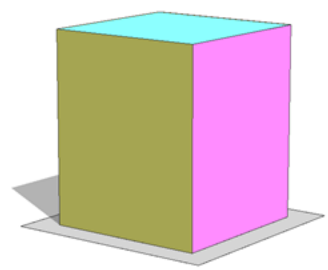
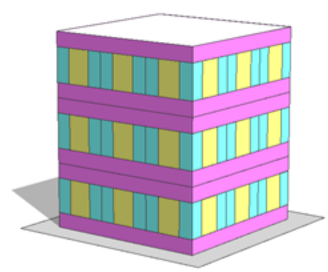
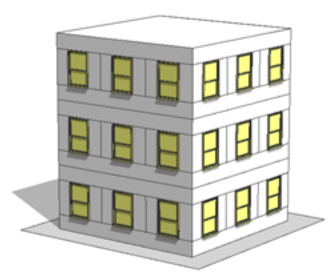
or inline comments

```
Lot -->
    Garden House /*Garage*/ Fence
    ...
    comp(f){ front : F  | /* side : S  | */ top : T }
```

Essential shape operations

Essential shape operations overview

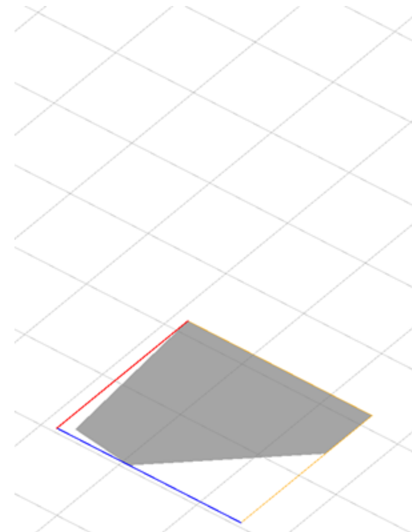
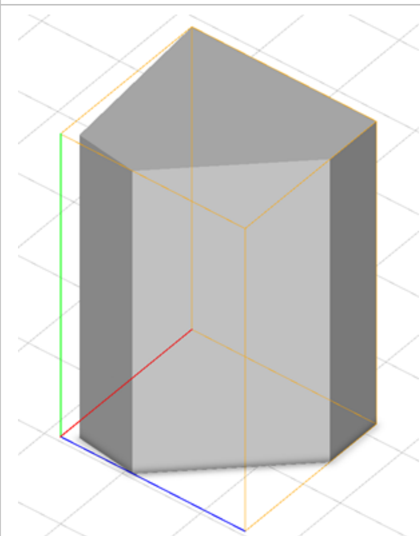
Shape operations are the actions that you can apply to shapes. Below is a list of essential operations with descriptions. You can find a list of all available operations in the CGA reference.

<table>
<tr><td>Extrusion</td><td>Extrudes the input shape to a given height.<pre><code>Lot --> extrude(13) Mass</code></pre></td><td></td></tr>
<tr><td>Transformation</td><td>Used to place, orient, and size the scope and geometry.<pre><code>Mass --> s('0.75, '1, '0.75)
         center(xz)
         Block</code></pre></td><td></td></tr>
<tr><td>Component split</td><td>Splits a shape into its sub-components, for example, a volume into its faces.<pre><code>Block --> comp(f) { side :
Facade | top : Roof. }</code></pre></td><td></td></tr>
<tr><td>Subdivision split</td><td>Splits a geometry along scope-axes.<pre><code>Facade --> split(y) { 1 :
Facade. | ~3 : Floor | 1 :
Facade. }*
Floor --> split(x) { 1 :
Floor. | ~2 : Window | 1 :
Floor. }*</code></pre></td><td></td></tr>
<tr><td>Insert assets</td><td>Loads an asset and inserts it in the current scope.<pre><code>Window --> i("window.obj")</code></pre></td><td></td></tr>
</table>

Extrusion

Extrusion is typically the first step to generate a 3D building from a 2D footprint. This operation increases the dimension, i.e. a two-dimensional building footprint can be extruded to a three-dimensional mass model, see extrude operation in CGA reference.

Example

`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 might be desired to extrude along a world coordinate axis rather than the polygon normal. This and other extrusion variants are explained in the extrude operation.	

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`. It is also possible to 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`. Hence, in contrast to the translate and rotate operations, the parameter values are not added but overwritten. Furthermore, note that the size operation sets the size in absolute values (e.g., meters or yards) and does not perform a relative scaling.
- `center(axes)` — translates the scope of the current shape such 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 directions (of the previous shape on the shape stack) the translation is performed.

Relative operator

For the `t()` and `s()` operations it is possible to conveniently transform the absolute values `tx,ty,tz` or `sx,sy,sz` to values relative to the scope size using the operator `'`.

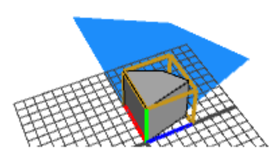
```
s('0.5, '1, '1)
t('2, 0, '3)
```

This is equal to:

```
s(0.5*scope.sx, 1*scope.sy, 1*scope.sz)
t(2*scope.sx, 0*scope.sy, 3*scope.sz)
```

Examples

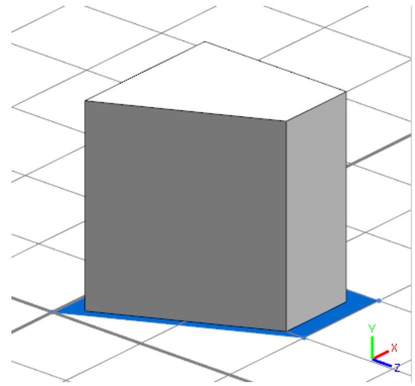
Setting the size

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 `'`, 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

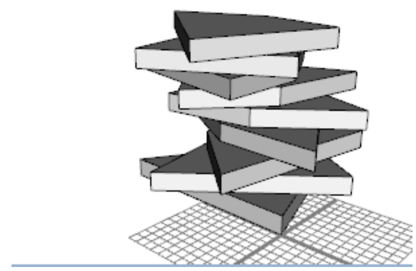
Each split shape is first rotated around its scope origin and then centered.

```
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)`

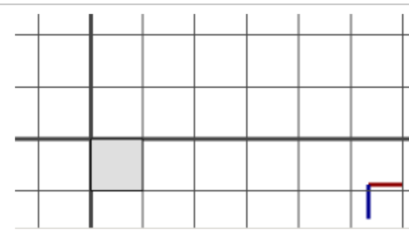
instead of the `r()` `center()` sequence gives the same result.



Translate - Rotation concatenation

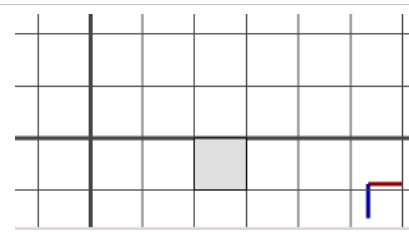
This is the shape we start with.

```
A --> primitiveCube()
```



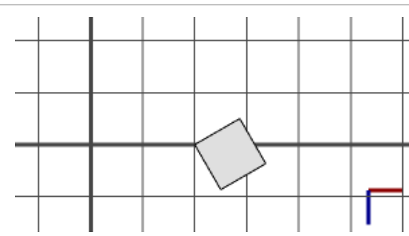
First a translation of two units along the x-axis.

```
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)
```

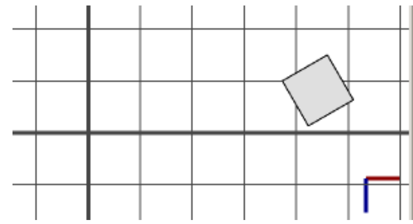


<table>
<tr>
<td>

And another translation of 2 units along the x-axis:

- translations are along the scope's x-axis, i.e. the rotation changes the global translation direction.
- the relative operator `'` is used - here 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)
```

</td>
<td></td>
</tr>
</table>

See the t operation, s operation, r operation, and center operation in the CGA reference for further detail.

Component split

A typical operation is to decompose an architectural design into geometric components. In the CGA shape grammar, the component split permits breaking down shapes into shapes of lesser dimensions.

The operation below splits a predecessor shape, based on its geometry, into its components and executes a set of operations on each component.

`comp(component){selector : operations | selector : operations ...}`

The parameter `component` 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 rule below creates a new shape B for `all` faces of shape A's geometry:

```
A --> comp(f) { all : B }
```

Similarly we use

```
comp(e) { all : B }
```

and

```
comp(v) { all : B }
```

to split into edges and vertices respectively.

Selectors

To access only selected components, we use operation calls such as:

```
comp(f) { 3 : B }
```

to create a shape consisting of the original shape's third face. Such calls are not very generic and require the user to be aware of the topology of the predecessor shape's geometry. Therefore, as an alternative, we use selectors:

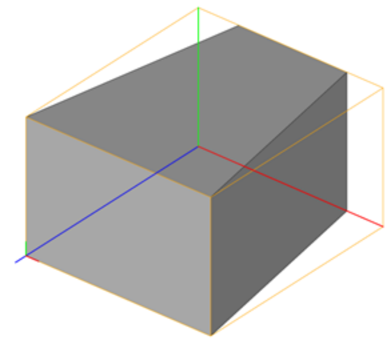
```
Building --> comp(f) { side : Facade }
```

This CGA grammar selects only the vertical `side` faces of `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 the the orientation of the scope).

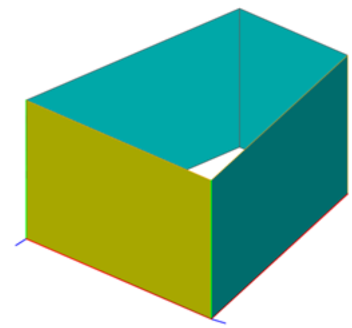
Example

<table>
<tr><td>

```
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 shows in the same colors.
</td><td></td></tr>
<tr><td>

```
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.
</td><td></td></tr>
</table>

Typically, the facades are then sudivided further into floors. Each of the new `Main` and `Side` shapes has its pivot and scope positioned and oriented such that the facade rules can be written conveniently.

See the CGA component split operation for more information.

Subdivision split

The split operation can be used to model shapes and to set up geometry by splitting a larger geometry object to smaller ones. The split operation is central to creating designs with the CGA shape rules. The basic definition for split is:

```
split(axis){size : operations}
```

See CGA split operation for detailed information.

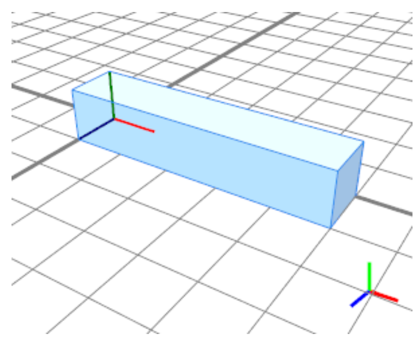
The following examples show the different types of the split operation:

- absolute split,
- relative split,
- floating split and
- repeat split.

The following are examples that start with an introductory rule and proceed 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 we resize the initial scope and insert a cube in that scope. The scopes' 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 scopes' 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.

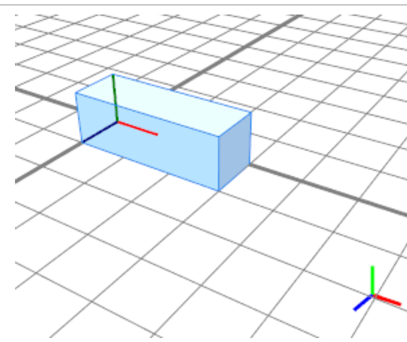


Absolute split

The absolute split cuts the geometry at absolute values.

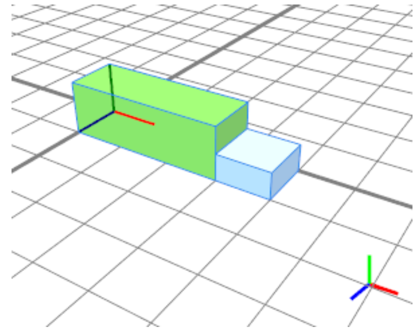
The operation `split(x) { 3 : Blue }` splits the geometry along the x-dimension of the scope. The cube of length 5 is cut at an absolute value of 3 units and replaced with the successive cube shape `Blue`.

```
Lot --> s(5, 1, 1)
        primitiveCube()
        split(x) { 3 : Blue(1) }
```



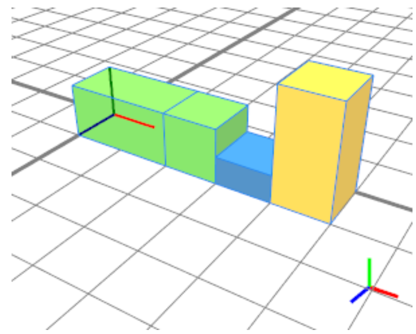
This example shows what happens if two absolute split values are used (3 units in green and 1 unit in blue). After the split the total resulting length is 3+1=4.

```
Lot --> s(5, 1, 1)
        primitiveCube()
        split(x) { 3 : Green(1)
                 | 1 : Blue(0.5) }
```



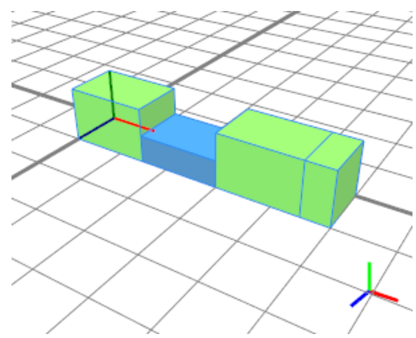
This example shows the effect of absolute values within a (limited) scope. The resulting shape `Yellow` has a length of 1 instead of 2 since there is a total length of 5. The rightmost shape `Blue` will not be generated.

```
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)  }
```

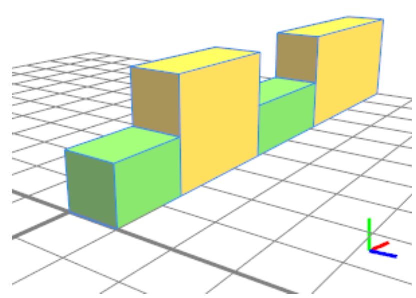


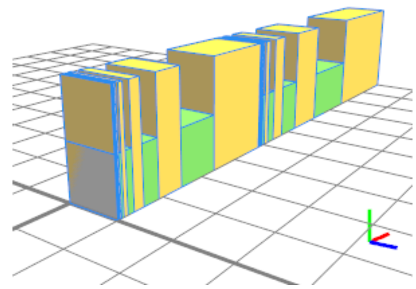
Relative split

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.

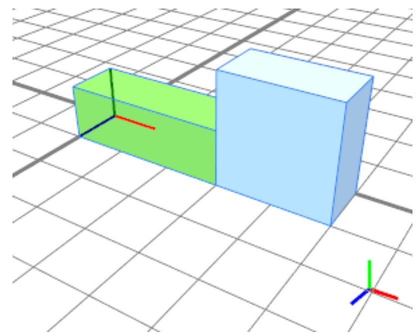
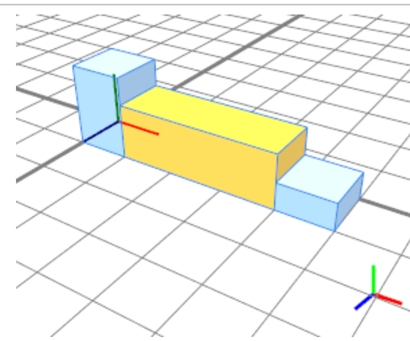
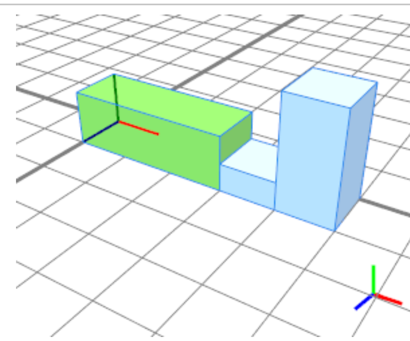
```
split(x) { '0.382 : A | '0.618 : B }

A --> split(x) { '0.382 : Green(1)  |
                 '0.618 : Yellow(2) }
B --> split(x) { '0.382 : Green(1)  |
                 '0.618 : Yellow(2) }
```

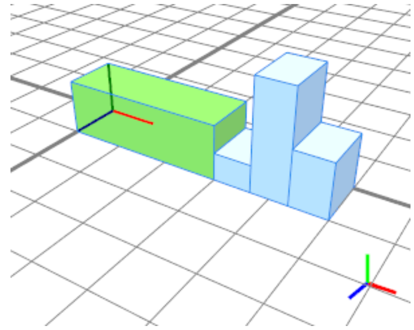
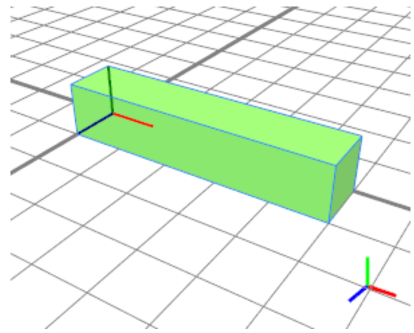


This example shows a recursive split using the Golden Ratio. The recursion stops when the geometry gets too small for a further split.
`split(x) { '0.382 : A | '0.618 : B }`
`A --> split(x) { '0.382 : A | '0.618 : C }`
`B --> split(x) { '0.382 : B | '0.618 : D }`
`C --> split(x) { '0.382 : Green(1) | '0.618 : Yellow(2) }`
`D --> split(x) { '0.382 : Green(1) | '0.618 : Yellow(2) }`
	

Floating split

A floating split adapts values such that the entire space is filled. Floating splits are denoted by an 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, i.e. 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

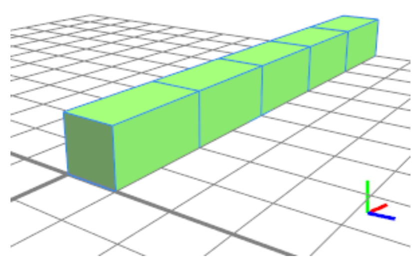
The repeat split is used to repeat geometry within a given scope. It creates a rhythm: the repetitive use of a group of geometric objects in order to establish a recognizable pattern. A typical scenario is the alternating arrangement of windows and columns in the facades of common high rise office buildings. An asterisk * after the bracket denotes the repetitive split.

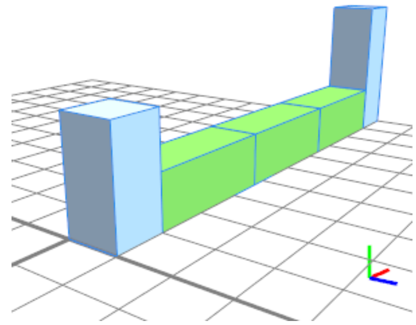
In the first example, a scope of length 10 is filled repetitively with a green shape with absolute length 2. Exactly 5 cubes of length 2 fit in the scope.

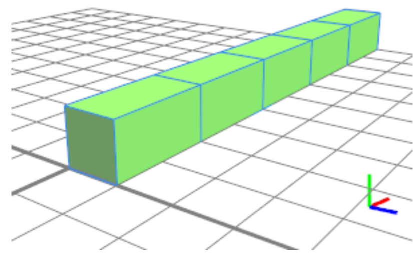
```
Lot --> s(10, 1, 1)
        primitiveCube()
        split(x) { 2 : Green(1) }*
```

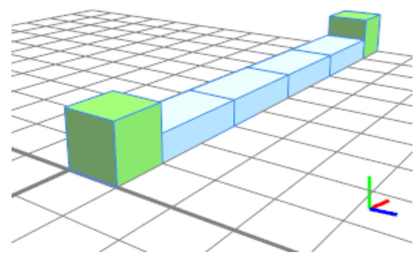
A floating value of 2.1 generates the same result because a split to 5 successor shapes (with a length of 2 each) still approximates the target length of 2.1 best.

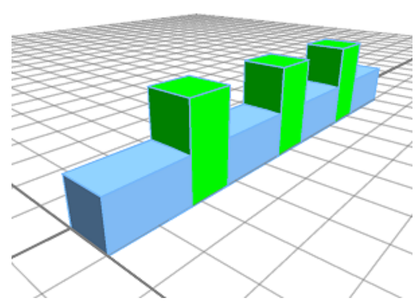
```
split(x) { ~2.1 : Green(1) }*
```



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≈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 which 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) }* }`
	

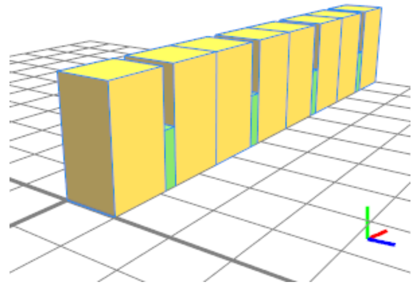
Repeat split - Patterns

Repeat operations can be used to create patterns. The following example shows that geometric objects can be grouped into a bracket and repeated.

<table>
<tr><td>

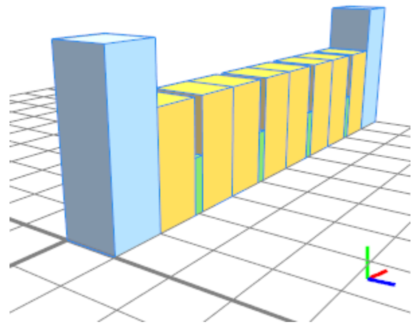
A pattern is created and repeated.

```
split(x) {  ~1 : Yellow(2) |
          0.25 : Green(1)  |
            ~1 : Yellow(2) }*
```

</td><td></td></tr>
<tr><td>

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)      }
```

</td><td></td></tr>
</table>

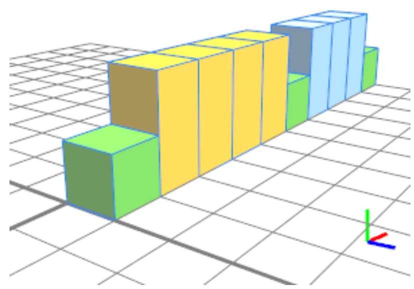
Repeat split - Parallel

Repeat splits can be performed in parallel.

<table>
<tr><td>

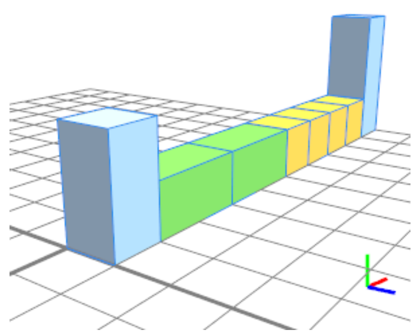
Two repeat splits (yellow and blue) bordered by floating splits (green).

```
split(x) { ~1 : Green(1)      |
         { ~1 : Yellow(2) }*  |
           ~1 : Green(1)      |
         { ~1 : Blue(2)   }*  |
           ~1 : Green(1)      }
```

</td><td></td></tr>
<tr><td>

Two successive repeat splits (green and yellow) bordered by absolute splits (blue).

```
split(x) { 1 : Blue(2)      |
         { 2 : Green(1) }*  |
         { 1 : Yellow(1) }* |
           1 : Blue(3)      }
```

</td><td></td></tr>
</table>

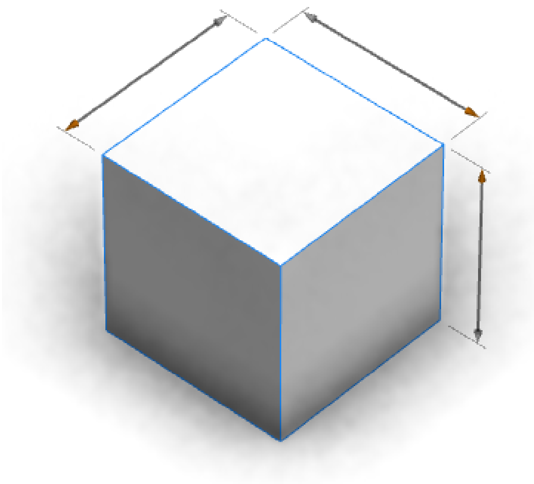
Insertion of assets

A typical last step is the insertion of assets, i.e. polygon meshes such as windows, doors etc. This is described in detail in the CGA insert operation.

Handles

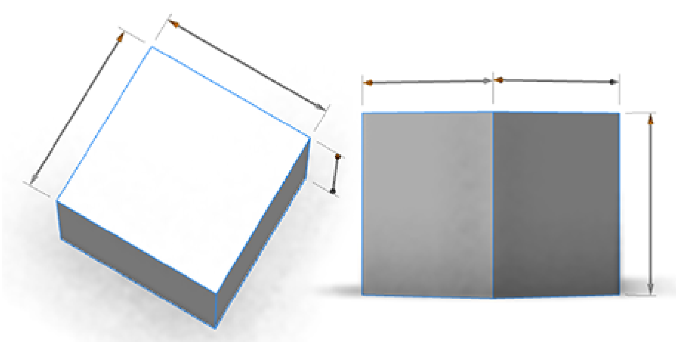
Handles overview

Handles allow users to edit the value attributes directly in the **Viewport**.



Handles with a model

The handles are positioned so that they are always visible, whichever camera angle is used.

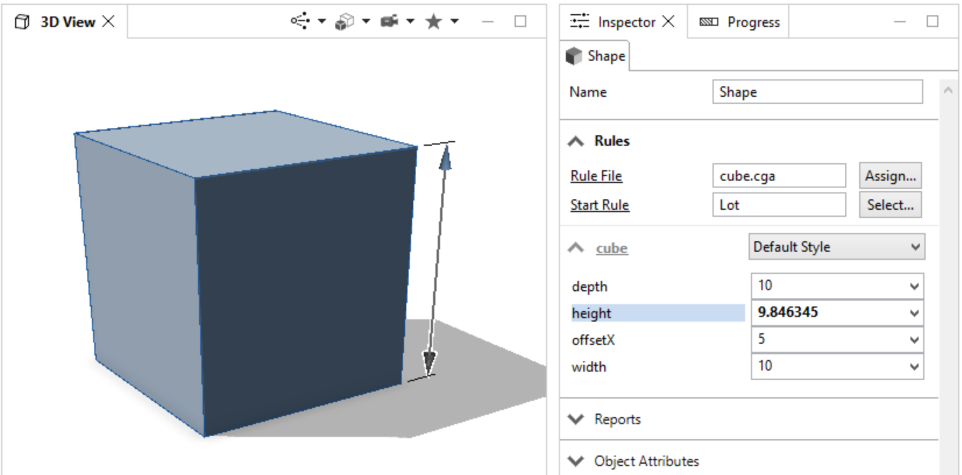


Handles in multiple camera angles

You can enable or disable the handles in the **3D View** toolbar under **View settings**
or in **Edit** > **Preferences** > **Scene**. By default handles are enabled when present.

Using Handles

When the mouse cursor is placed over a handle, CityEngine highlights both the attribute in the **Inspector** and the associated scope in the view.



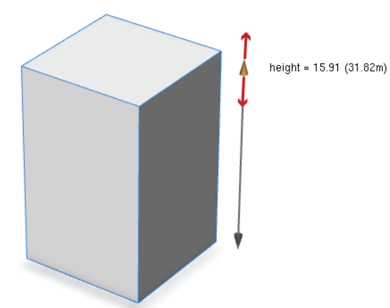
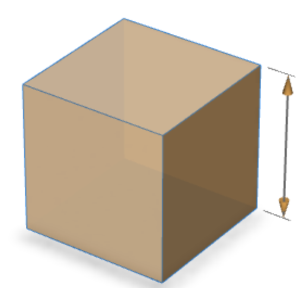
Handles in the Inspector

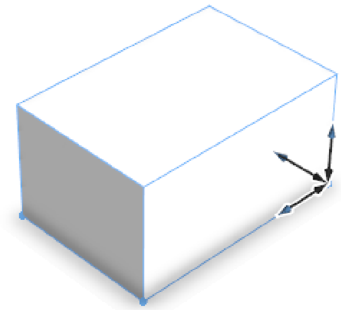
Tip:

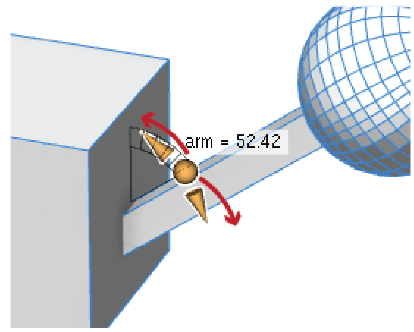
Small handles or handles attached to small parts of the model are sometimes not shown. To show these handles, press the control key.

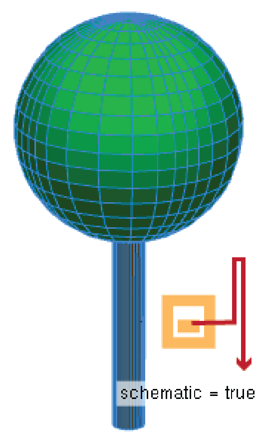
Handle types

There are several different types of handles for editing different types of attributes, such as linear distances, angles, Boolean switches, multiple choice or colors. They are manipulated by dragging, or clicking, the colored (orange) elements.

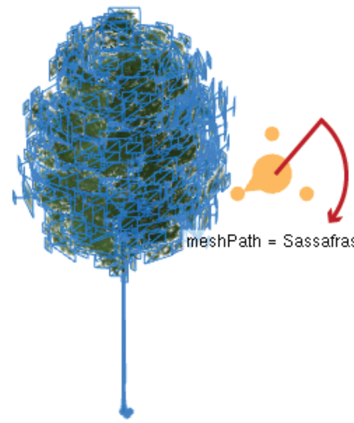
Linear	
Linear handles adjust distance attributes. They are edited by dragging the colored markers. When dragging the handle, the value of the attribute is shown. If the value of the attribute is different to the length of the linear handle, the length of the handle is shown in parenthesis.	
Some linear handles can be dragged from either end. These handles have two colored arrows.	

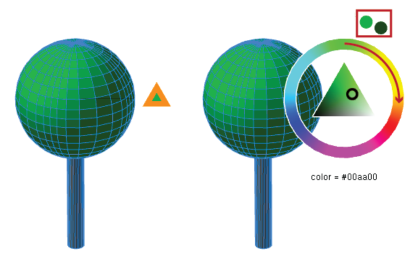
Move	
Move handles adjust position or offset attributes. They are similar to linear handles, however the length of the move handles is always constant, and dragging one end also moves the opposite end by the same amount.	

Rotational	
Rotational handles adjust angle attributes. They are edited by dragging the colored sphere.	

Toggle	
Toggle handles adjust Boolean (on/off) attributes. They are edited by clicking the colored square, or dragging the square up and down.	

Selector

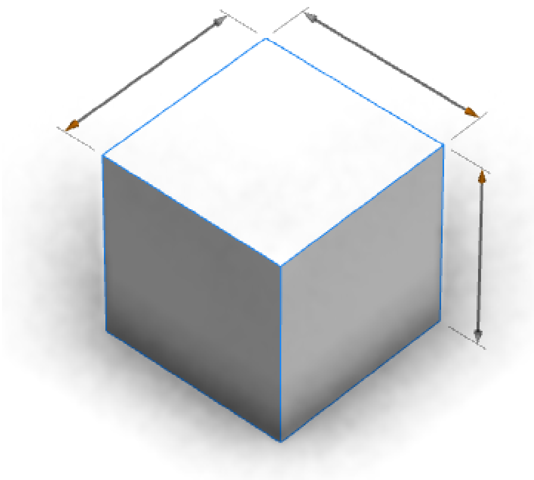
Selector handles edit multiple choice values. They are edited by clicking the colored circle, or by dragging in a clockwise or anticlockwise direction around the circle.	

Color	
Color handles edit color attributes. The are edited by clicking and dragging on the colored triangle. When a drag starts, the hue circle appears. Dragging within (or outside) the hue circle adjusts the hue, while dragging inside the triangle adjusts the saturation and brightness. The selected color is shown by a circle within the triangle. Alternately, clicking the triangle, and dragging to a color si watch selects that color.	

The range of handles may be limited by the rule (using @Range and @Enum annotations). In this case the values of the handles are constrained when editing.

Create handles

Handles are created by adding a @Handle annotation to an attribute.

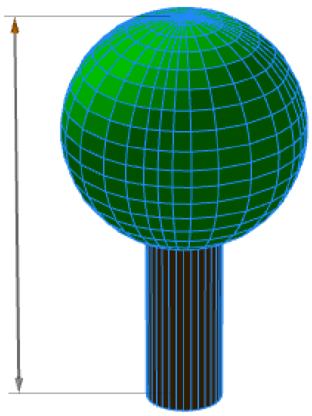


Model with @Handle annotation added

The above handles 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 option `shape=Cube` attaches the handle to the shape `Cube`. The option `axis=x|y|z` specifies a scope direction for the handle. Thin gray extension lines connect the handle to the shape that it is measuring. Handles can be applied to a wide range of objects.



Tree model with handle

The above handle was created using the annotation:

```
@Handle(shape=TreeCenter, axis=y, reference=center, slip=screen)
attr height = 30
```

Handles that do not move with the camera can also be created, for example:

```
@Handle(shape=TreeCenter, axis=y, reference=origin, slip=inside, occlusion=false)
attr height = 30
```

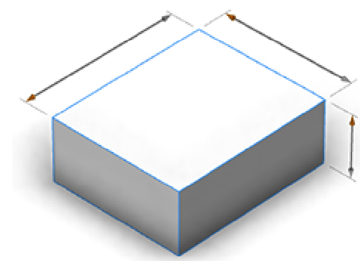
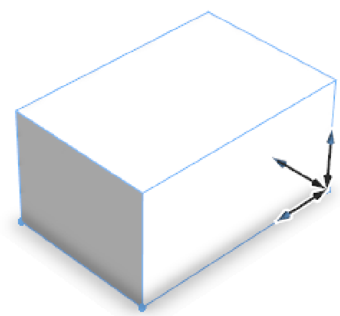
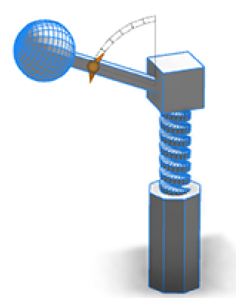
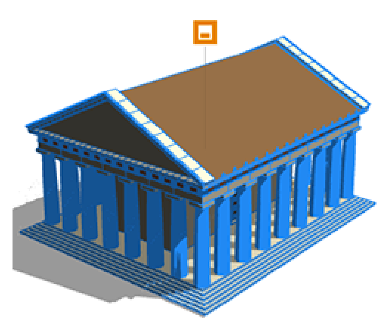
The advanced options `reference` and `slip` specify the handle's position and movement as the camera moves. In this case the height handle rotates smoothly around the tree with the camera.

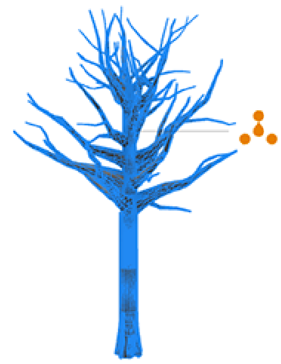
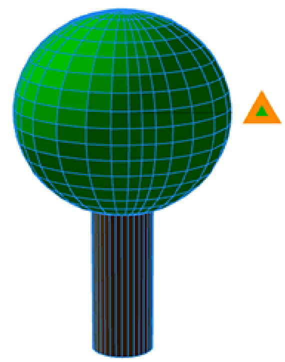
Note:
Adding many handles, or adding handles to complex models may lead to low framerates on low-end hardware.

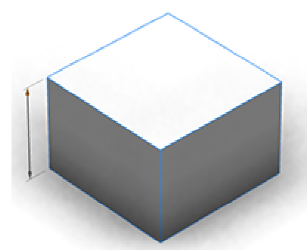
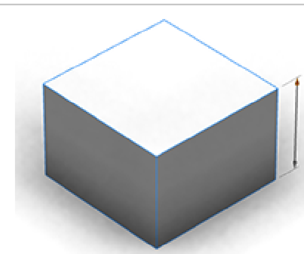
General options

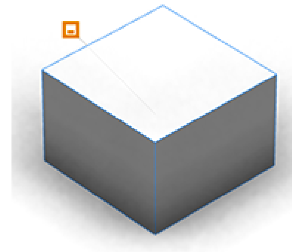
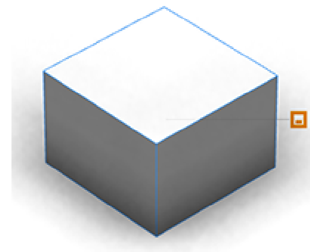
The default options in the below table are marked with an asterisk. If any option is omitted, the default value is taken. The only option that must be specified is `shape`.

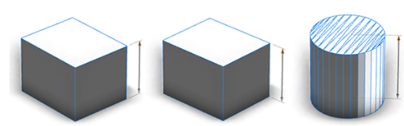
shape
`shape=shape_name^argument_count`
Selects the shape of the rule with name `shape_name`. The parameter `argument_count` specifies the number of arguments that the rule takes.
If `^argument_count` is omitted, `^0` is assumed. An `*` allows referencing anonymous leaf shapes, e.g.: `shape=Box*^3`.

type	
`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 given below:	
`type=linear` Handles with a linear type are used for editing float values that represent distances.	
`type=move` 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.	
`type=angular` The angular type is for float attributes that represent angles.	
`type=toggle` The toggle type is for boolean attributes.	

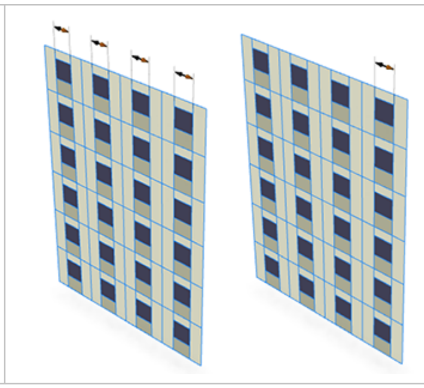
`type=selector` The selector type is for attributes with the @Enum annotation.	
`type=color` The color type is for editing color string attributes.	

align	
`align=topLeft|left|bottomLeft|bottom|bottomRight|right|topRight|top|default*`	
Selects a screen direction as the offset direction preference for a reference handle. `default` selects the nearest location outside of the model's silhouette.	
linear example of `align=left`	
linear example of `align=right`	

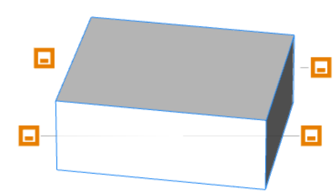
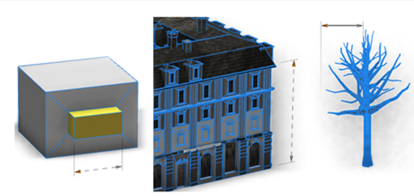
toggle example of `align=topLeft`	
toggle example of `align=right`	

slip	
`slip=scope*|screen|inside`	
This option specifies the way in which reference handles may be moved outside the model's silhouette. `slip=scope` moves the handle in directions given by the scope axes, `slip=screen` moves the handle parallel to the camera, and `slip=inside` disables the offsetting behvior.	
Different `slip` are appropriate for different shapes. For example, the length of cylindrical objects is best specified using `slip=screen`, while the dimensions of a cuboid should use `slip=scope`.	
`slip=scope` causes the handle to align itself to the edges of the scope as the camera moves. In contrast `slip=screen` maintains a constant offset direction, independent of camera location. It is recommended that cuboid objects use `slip=scope`, while cylindrical or spherical objects will use `slip=screen`. While it is possible to use either of these options in any case, following these recommendations presents a consistent and intuitive interface to users.	

repeat
`repeat=chain*|none`
Chained handles will be clustered into a single continuous chain. Linear handles of several different attributes with the same orientation may be grouped onto one chain if there is sufficient space.

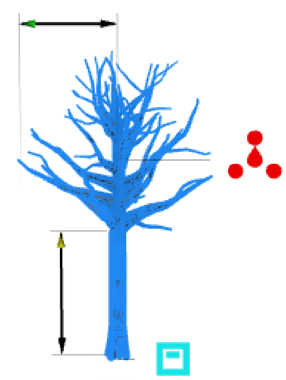
Left: repeat=chain, right: repeat=none If `repeat=none` is set, CityEngine determines an appropriate location for a single handle, preferring locations with short extension lines and long linear handles.	

minDisplaySize
`minDisplaySize=pixels`
If the size of a handle is below `pixels`, it is not shown. Users can override this behavior by pressing or control key.
The size of a handle with linear type is the screen-length. The size of an angular handle is the distance between zero degree and the handle. The size of toggle, selector, or color handles is the screen length of the shape's scope's shortest edge.

extensionLines	
`extensionLines=scope*|silhouette|fade|off`	
Specifies the style of the extension lines associated with a handle.	
Clockwise from bottom right: extensionLines=scope, extensionLines=fade, extensionLines=off and extensionLines=silhouette.	
The image below shows typical use cases for the different extension line types. `scope` (left) is used to highlight embedded features, `silhouette` (middle) with obvious features to avoid cluttering the geometry and `fade` (right) for irregular shapes.	

translate
`translate={translate_x,translate_y,translate_z}`
Specifies a scope-relative translation of the handle. For example, `translate={3,0,0}` translates the handle by 3 * `scope.sx` in the direction of the scope's x axis.

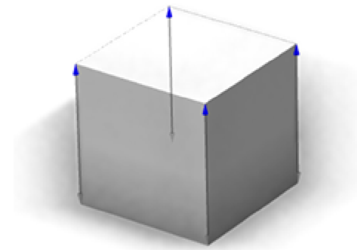
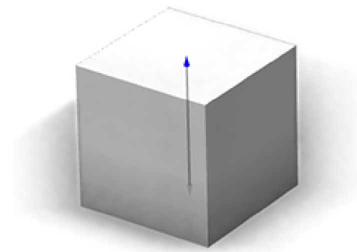
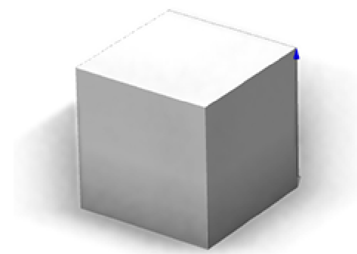
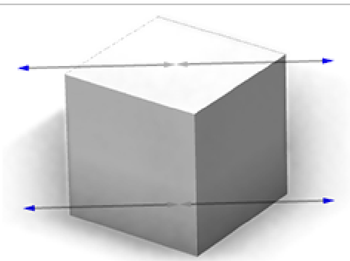
occlusion
`occlusion=true*|false`
Enables or disables occlusion handling for the handle. When enabled handles are only shown when their reference positions are visible. However, for complex models this may cause the handle to flicker. Using `occlusion=false` will resolve this issue, and never hide the handle because of occlusion.

color	
`color=hexCodeString`	
Normally the color of handles is given by the viewport hightlight 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"`.	
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".	

Linear type options

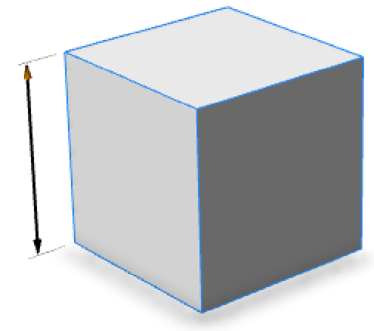
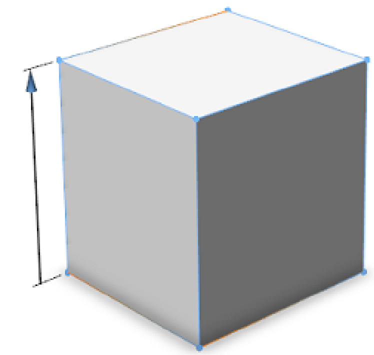
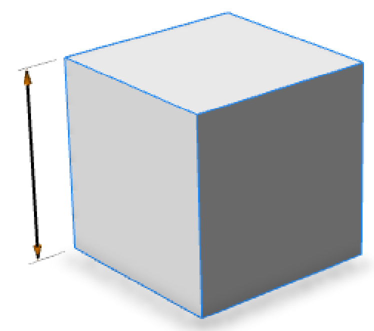
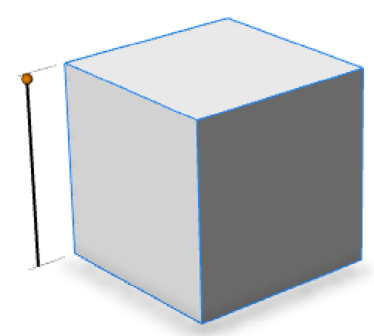
Linear handles are assumed to be associated with a scope edge. The length of the linear handle is given by the scope edge specified by `axis`.

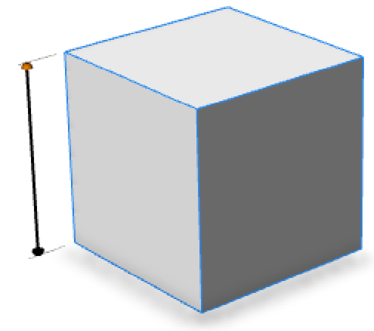
reference
`reference=edges*|center|origin|radial`
Specifies the location of the reference handles. **Note:** The location of radial handles depends on the camera position.
Handles are moved to their offset position using the direction given by the `slip` option.

`reference=edges` Typical use cases are boxes or cuboid geometries (as shown in the first example on this page).	
`reference=center` Suitable for the length of cylindrical or spherical objects (such as the tree's height in the second example of this page).	
`reference=origin` 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.	
`reference=radial` Can be used for the width of cylindrical objects (such as the dead tree model above).	

axis
`axis=y*|x|z|x-|y-|z-`
The linear handle is positioned along the specified scope axis of the shape.

skin
`skin=doubleArrow*|singleArrow|diameterArrow|sphere|hemisphere`
Linear handles have several options for the rendering of terminators. These options do no change the behavior of the handle, with the exception of `skin=diameterArrow`, which scales symmetrically about its center when dragged.

`skin=doubleArrow`	
`skin=singleArrow`	
`skin=diameterArrow`	
`skin=sphere`	

`skin=hemisphere`	

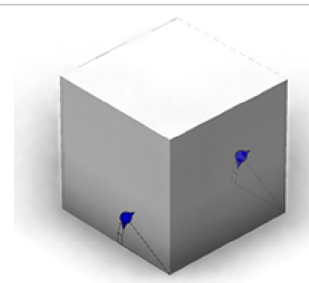
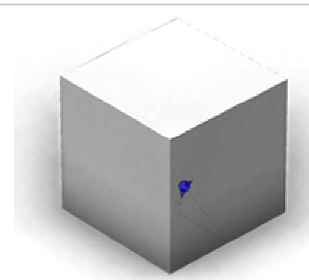
Move type options

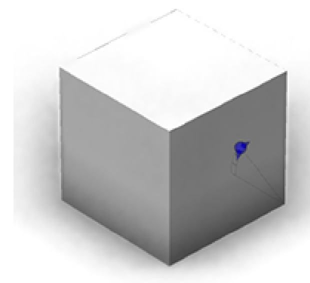
Move handles are similar to linear handles, and are also associated with a scope edge. However, the length of the move handle is always constant, in contrast to linear handles where it is given by the scope edge. When dragging one end of the handle, the other end also moves by the same amount. This makes them suitable for attributes representing positions or offsets.

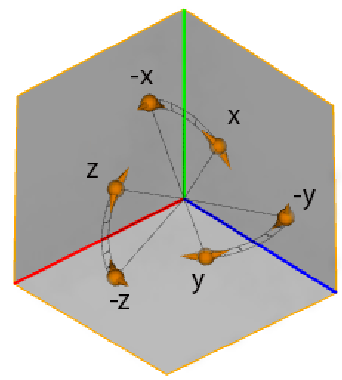
`axis` and `skin` are similar to linear handles. For `reference`, only `center` and `origin` is supported. Also, the `slip` value is restricted to `inside`.

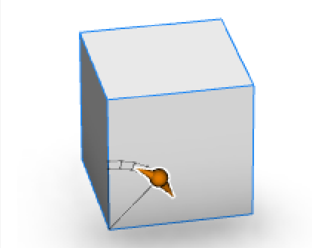
Angular type options

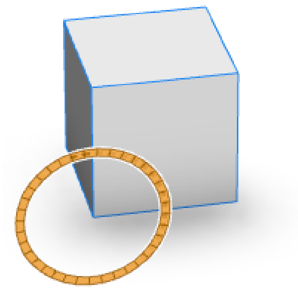
Angular handles allow the user to manipulate angles. Typically a combination of `axis` and `translate` are used to position the handle in the appropriate location. By default, angular handles do not offset themselves (`slip=inside`). This behavior can be altered by specifying another `slip` value.

reference	
`reference=edges|center*|origin`	
Specifies the location of the reference handles. In the below images, the shape origin is in the bottom right of the screen.	
`reference=edges`	
`reference=center`	

`reference=origin`	

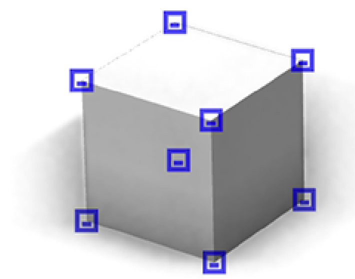
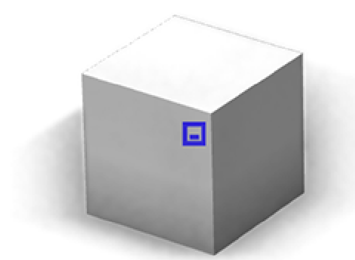
axis	
`axis=y*|x|z|x-|y-|z-`	
Selects the rotation scope axis.. A negative axis reverses the direction of rotation.	
The red, green and blue lines are the x, y and z axes. For example `axis=x` rotates around the x axis towards the z axis. The angular handles each have an attribute value of 30 degrees	

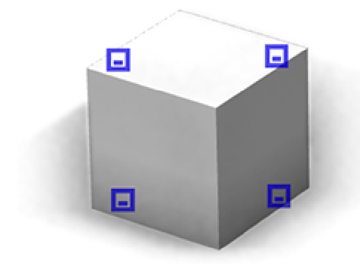
skin	
`skin=doubleArrow*|ring`	
Rotational handles have two skin options: `doubleArrow` or `ring`.	
`skin=doubleArrow`	

`skin=ring`	

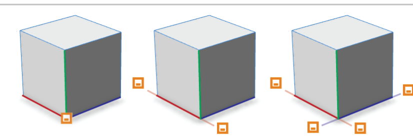
Toggle, Selector and Color Type Options

Toggle, selector and color types share layout options.

reference	
`reference=edges|center*|origin|radial`	
Specifies the location of the reference handles. In the below images, the shape origin is in the bottom right of the screen. **Note:** The location of radial handles depends on the camera position.	
`reference=edges`	
`reference=center`	
`reference=origin`	

`reference=radial`	

axis

if slip=scope	
`axis=x|y*|z|xy|xz|yz`	
Specifies the directions in which the handle may move when `slip=scope` is set. For example, handles with the option `axis=xy` may move along the x or y axes.	
Left: the reference position for the toggle handle at the origin, and the three axes. Middle: `axis=x`, the handle can move alone the x axis. Right: `axis=xy`, the handle can move along the x or the y axis to find the best location.	

if slip=screen
`axis=x|y*|z`
In the case of `slip=screen` the handle moves in the same manner as a linear handle with the same axis parameter.

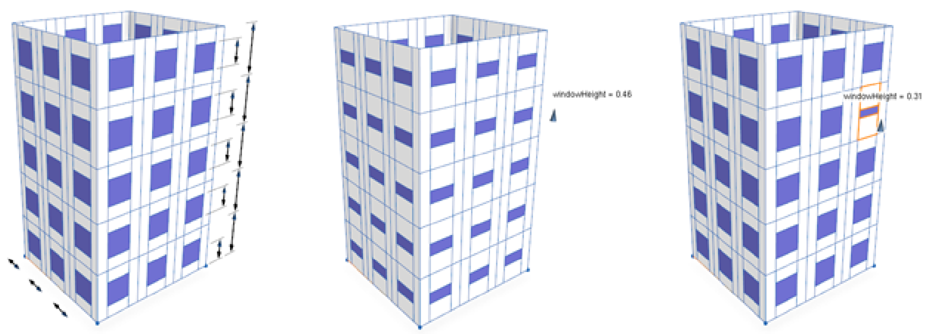
if slip=inside
`axis` does not affect the placement of the handle if `slip=inside`.

Local edits

Local edits overview

By using handles attributes of a selected CGA model can be edited in the **Viewport** in an intuitive manner. 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, imagine a building where the window height of all windows 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.



Left: Initial Building, Middle: Global Edit, Right: Local Edit.

To do this, first activate the **Local Edits** tool , and then click on a window in the **Viewport**. Changing an attribute in the **Inspector** or using handles will now only affect this window. To exit the **Local Edit** tool, switch to the **Selection** tool .

Use local edits

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 displayed to make global edits.
   The global handles of the selected part disappear. Handles for local edits are displayed instead.

- The **Local Edits** tool is available automatically for all attributes that have a handle annotation. See handles for more information.
- Press Shift and click to select multiple parts of a building. See Multiple selections for details.
- You can use the right-click menu for the **select higher/lower level** and **next/previous pattern** to select logical groups such as rows and columns of parts. See Local edits on patterns for details.
- Attributes with local edits are marked orange in the Inspector. See Manage local edits for details.

Work with local edits

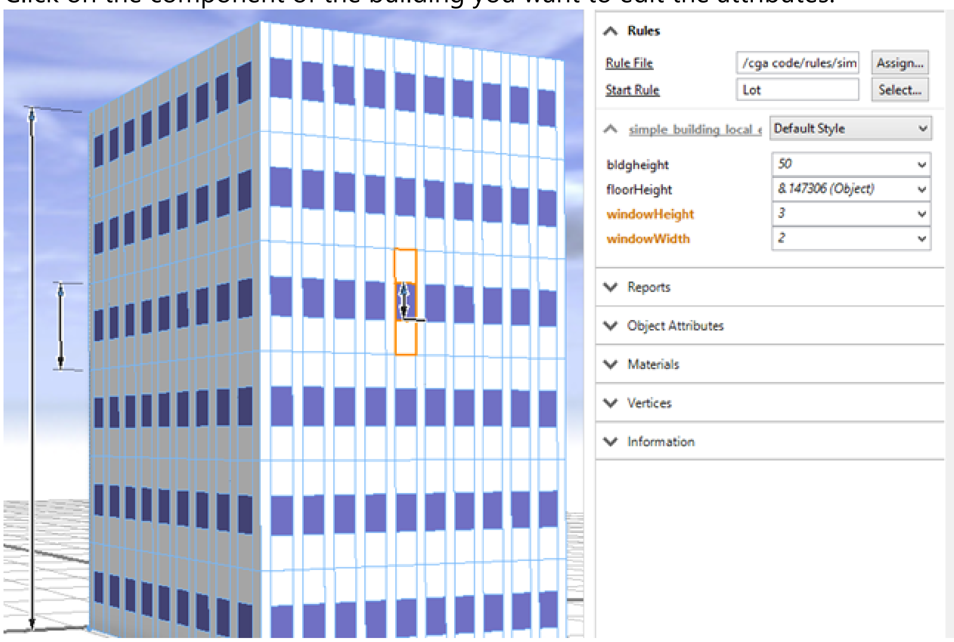
The following page gives you detailed descriptions of **Local Edit** behavior and functionality.

Local edits on single selection

The area of effect of a local edit depends on the selection. You can select single or multiple parts of a building.

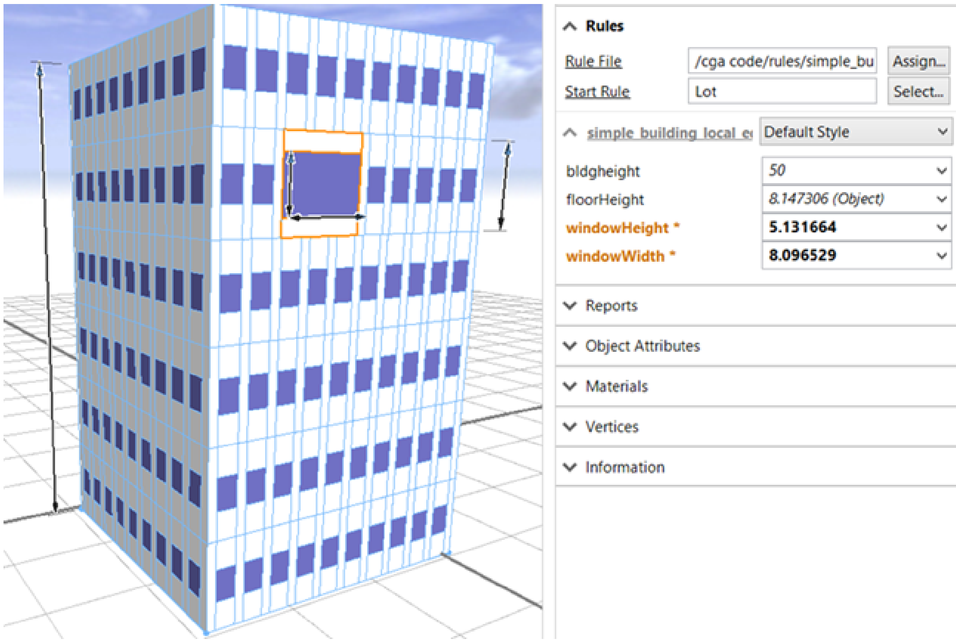
When you select components of a building with the Local edits tool, an orange highlight shows the current area of effect in the **Viewport**. The editable attributes in this area are also highlighted in orange in the Inspector.

1. Click the Local Edits tool to activate local editing.
2. Click on the component of the building you want to edit the attributes.

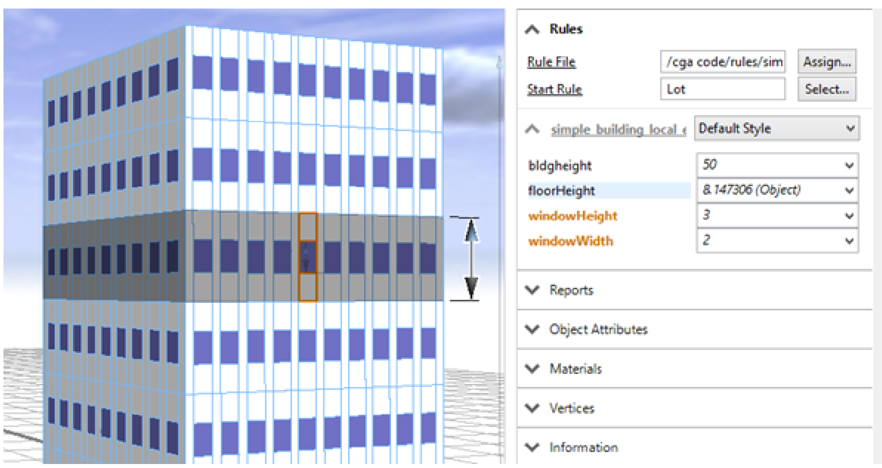


3. Edit the attributes by adjusting the handles of a highlighted attribute, or make the edits directly in the **Inspector**. You are affecting the attributes only for the current highlighted selection.

After making local edits, the edited attributes will have orange highlights with a * symbol in the Inspector. The * symbol indicates that the attributes have local edits applied in the current selection. The **Inspector** also displays the attribute values in bold after changes are made as it normally does in the **Inspector**.



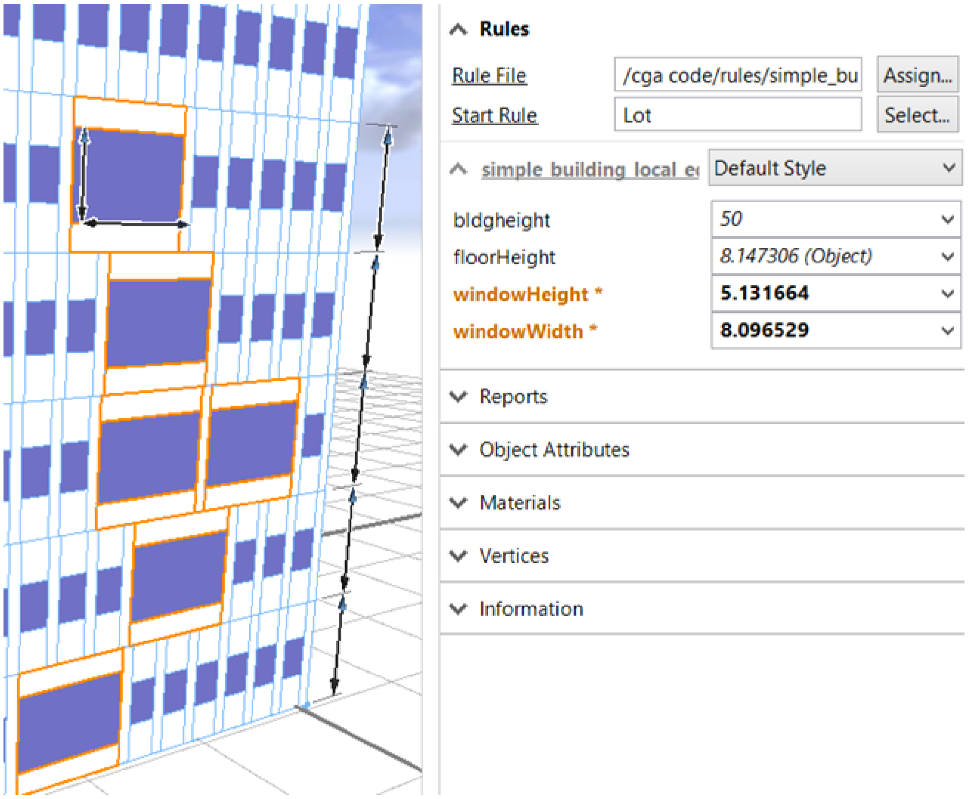
You can also edit attributes that are outside the current orange highlight, such as `floorHeight`. This will have a greater area of effect. To see the area, hover over the corresponding handle.



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. Hold `Shift` and click to add elements to the selection.
   Edits will only affect attributes of the selected components.



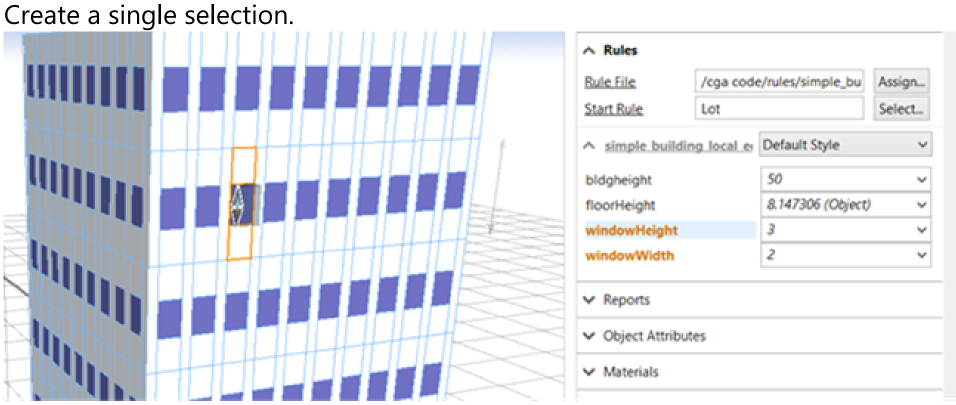
2. Hold `Ctrl + Shift` and click to remove items from the selection.

Local edits on higher levels

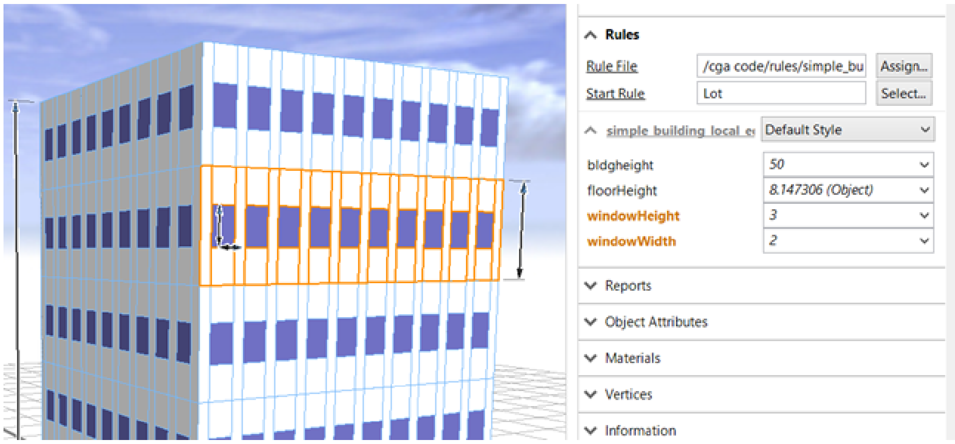
During rule application, a shape tree is created with multiple levels, such as windows, floors, and facades. See Shape Tree for more details.

This shape tree can be used 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 can be selected using the **select higher level** mechanism as follows:

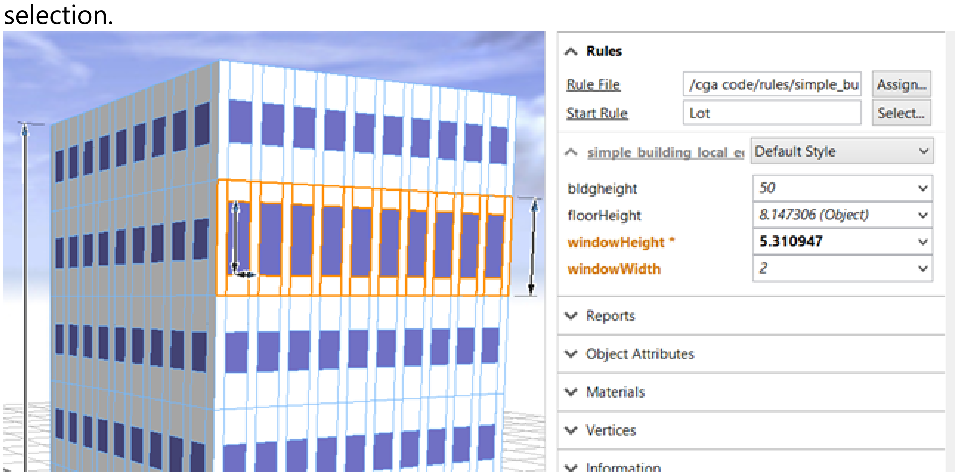
1. Create a single selection.



2. Right-click the local selection and choose **Select higher level** or press `PageUp` to select the next level up.



3. You've selected all the windows on that higher level. The `windowHeight` and `windowWidth` attributes are highlighted in orange in the Inspector which indicates they can be locally edited.
4. Edit the `windowHeight` attribute with the handles or the Inspector. This will change all the windows in the selection.



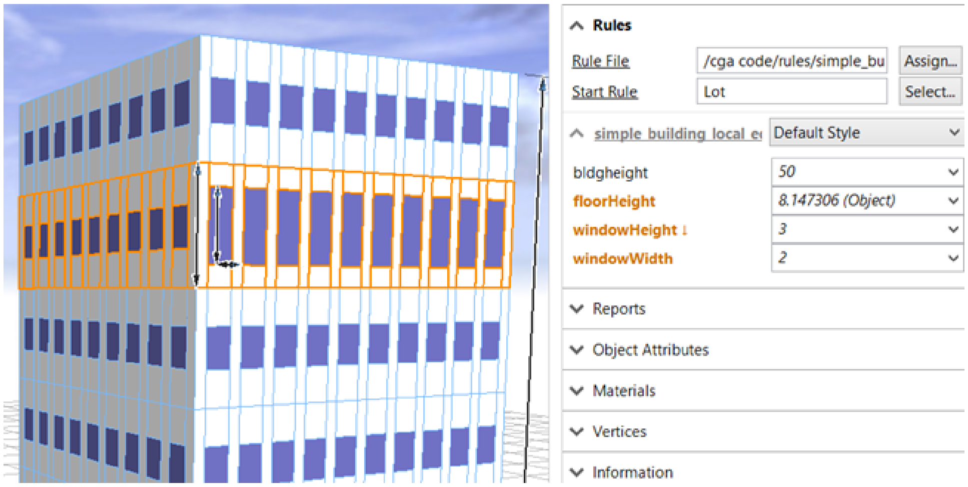
The Inspector now displays the following:

- `windowHeight` and `windowWidth` are highlighted attributes, 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.

Local edits on multiple levels

By stepping even higher up, edits can be performed on multiple levels. When multiple edits apply, always the lowest edit, which is nearest to the leaf shape, has precedence.

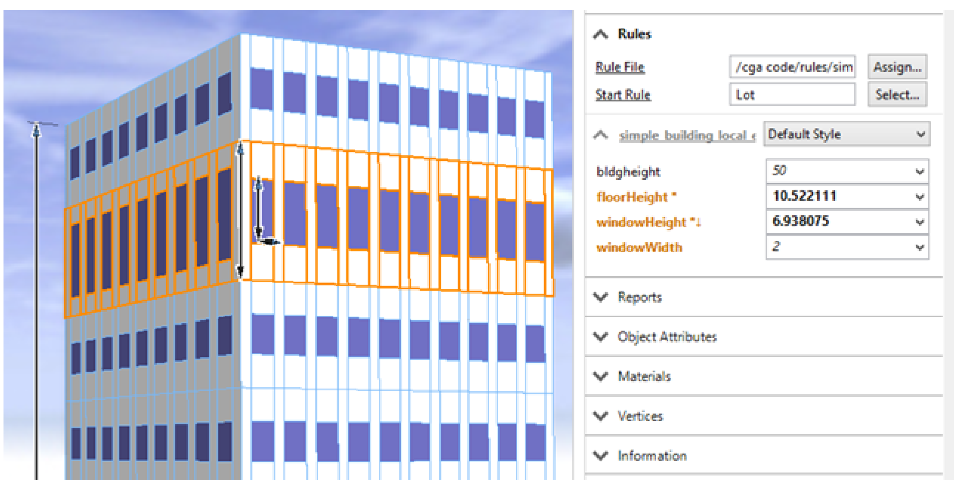
1. Press `Pageup` again to select the next higher level.



You can see in the Inspector that `floorHeight` is now also highlighted.

- `floorHeight` , `windowHeight`, `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` attribute.
   Edits to `windowHeight` don't affect the previously selected level because the lower level already has an edit on it which takes precedence. However, the other levels are affected, because they do not have a lower edit.

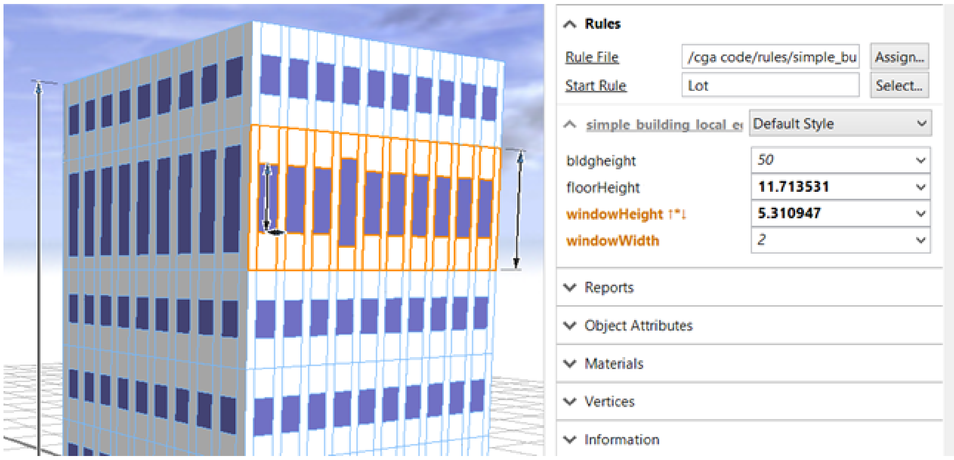


The **Inspector** displays the following:

- `/floorHeight`, `windowHeight`, `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. Use the `Pageup` or `PageDown` keys to change the current level and display the * symbol in the **Inspector** 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** that indicates local edits at the current selection level, a higher level, and a lower level.

1. Press `PageDown` until you get to the lowest level.
2. Edit the `windowHeight` attribute.
3. Press `Pageup` to go to a higher level.



The **Inspector** displays the following:

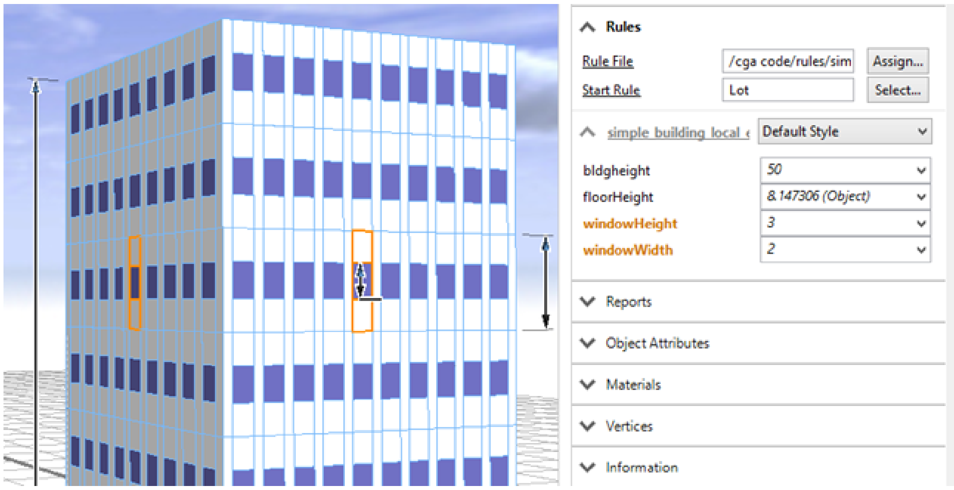
- `windowHeight`, `windowWidth`- indicate local edits will only affect the current selection.
- `floorHeight` - Local edits have been applied at a higher level, but the attribute is not currently selected.
- `windowHeight↓*` ↑ - Local edits have been applied at the current, lower, and a higher level.

Note:
To select different level components, you may need to press `PageUp` or `PageDown` 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 `PageDown` 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.

Local edits on patterns

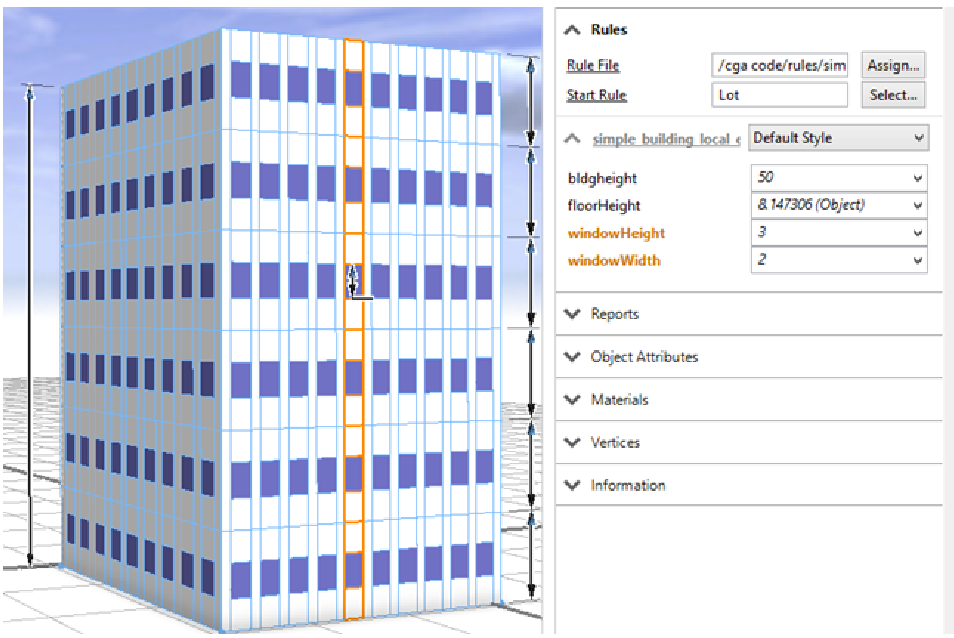
Use pattern selections when you want to select objects based on a pattern. Patterns are automatically detected.

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 will estimate a pattern to your selection and make 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 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

Discovering Local Edits

To see if local edits are applied to a specific selection, first activate the Local Edit tool

. When making a selection, the Inspector now indicates if 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 `PageUp` and `PageDown` to change the selection level. Stepping all the way up will highlight all performed local edits in the Inspector with a ↓ sign.

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** to get 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 needed referenced assets and data. Rule packages are an easy way to share procedural rules in a handy single file. For finding the package content, the rules are analyzed.

Note:
You can create very complex asset dependencies in CGA. Therefore, it cannot be guaranteed that all assets are included in every case. When in doubt, add the files manually.

Creating a rule package

Rule packages can either be stored locally on your machine or published to ArcGIS Online. See Online and Enterprise for more information.

1. Locate the a CGA rule in the **Navigator**.
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.
   - **Upload package** and specify a name for the rule package
   - **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 in order 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 your rule package.
   If any issues are discovered, an error will be reported. You must validate and resolve all errors before you can save it to disk or share it to ArcGIS Online.
9. Click **Share** to to upload or save the rule package.

Note:
Publish to ArcGIS Online:
- If you haven't already you will be asked 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 Sharing data on a different Portal

Importing

Import data overview

Drag and drop — Drag a file onto the desired folder in the **Navigator** to add it to your workspace. Drag it into the **Viewport** to add it to the current scene. In this step, the import options defaults are used that are suitable for most cases. This is the most simple import method.

Import dialogs — Suitable for batch import and when default settings need to be adjusted. This is the import method for advanced users.

From ArcGIS Online — Using the **Navigator**, you can connect to your ArcGIS Online (or portal) account and download data into your project.

From Get map data — This feature lets you import an extract of Esri world terrain and satellite imagery. Additionally building footprints and street networks from OSM (Open Street Network) can be imported directly into the current scene.

Importing projects into your workspace is not part of this chapter.

Import by drag and drop

1. Drag a file onto the desired folder in the **Navigator** to add it to your workspace.
2. Drag it into the **3D Viewport** to add it to the current scene. In this step, defaults are applied that are suitable for most cases.

File type	Drag and drop import behavior
DAE (COLLADA)	Import as static model
DWG (Autodesk)	Import as static model
DXF (AutoCAD)	Import as shapes and/org graph segments
FBX (Autodesk)	Import as shape or static model
FGDB (Esri ile Geodatabase)	Import all supported layer types, as shapes, point shapes or graph networks
glTF (Khronos Group)	Import as shape or static model
IFC (buildingSMART)	Import as static model
KMZ / KML (Keyhole Markup Language)	Import all referenced dae objects as static models
OBJ (Wavefront)	Import as shape or static model
OSM (Open Street Map)	Import all layers, as shapes and/or graph segments
SHP (Esri Shapefile)	Import as shapes, point shapes multipatch shapes or graph networks
USD (Universal Scene Description)	Import as shape or static model

Drag and drop import of image data

Dragging an image file from the **Navigator** into a **3D Viewport** brings up the **Terrain** dialog.

Placement of data after drag and drop import

- Georeferenced data such as kml, gdb and shp is placed at its georeferenced location.
- Data with no georeferenced information is placed at the drop spot, taking the data's center as pivot.

Import by dialog

CityEngine allows you to import files into your scene through the following options:

- In the **Navigator**, right-click and select **Import...**
- In the main-menu, select **File** > **Import...**

In contrast to Drag and Drop import, each import format will present a file dialog with additional options.

Depending on the file format, the following list gives the resulting layer type when importing in CityEngine:

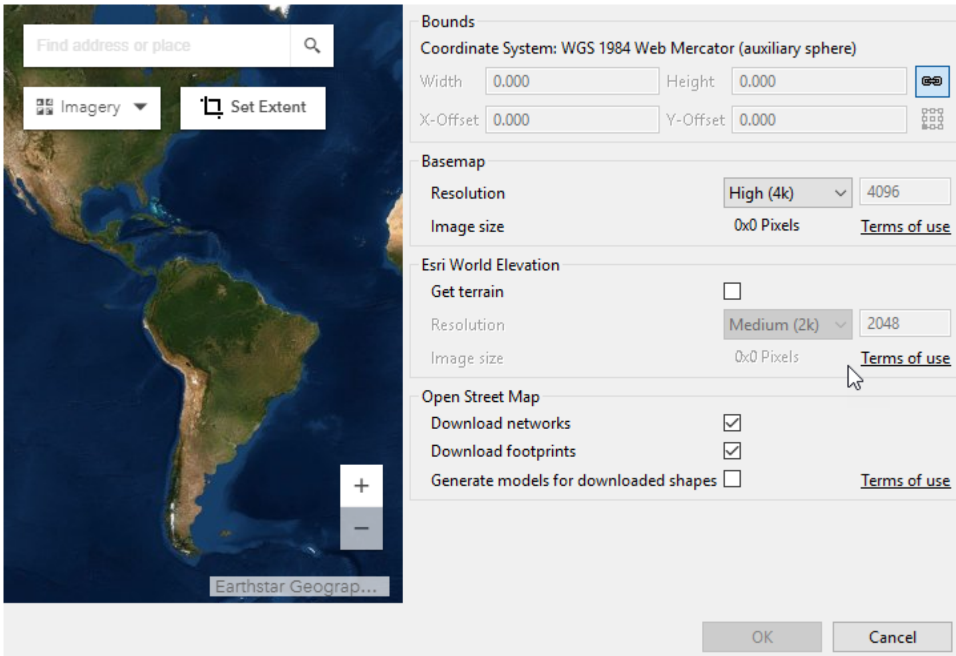
File format	Resulting layer type
DAE (COLLADA)	Shape, Static Model
DWG (Autodesk)	Shape, Static Model
DXF(AutoCAD)	Shape, Graph
FBX(Autodesk)	Shape, Static Model
FGDB(Esri File Geodatabase)	Shape, Graph
glTF (Khronos Group)	Shape, Static Model
IFC (buildingSMART)	Shape, Static Model
KMZ / KML (Keyhole Markup Language)	Static Model
OBJ (Wavefront)	Shape, Static Model
OSM (Open Street Map)	Shape, Graph
SHP (Esri Shapefile)	Shape, Graph
USD (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:

- Select **Create a city from map data** from the CityEngine welcome screen to create a new scene from downloaded data.
- Select **File** > **Get Map Data** from the main menu to import downloaded data into an existing scene.

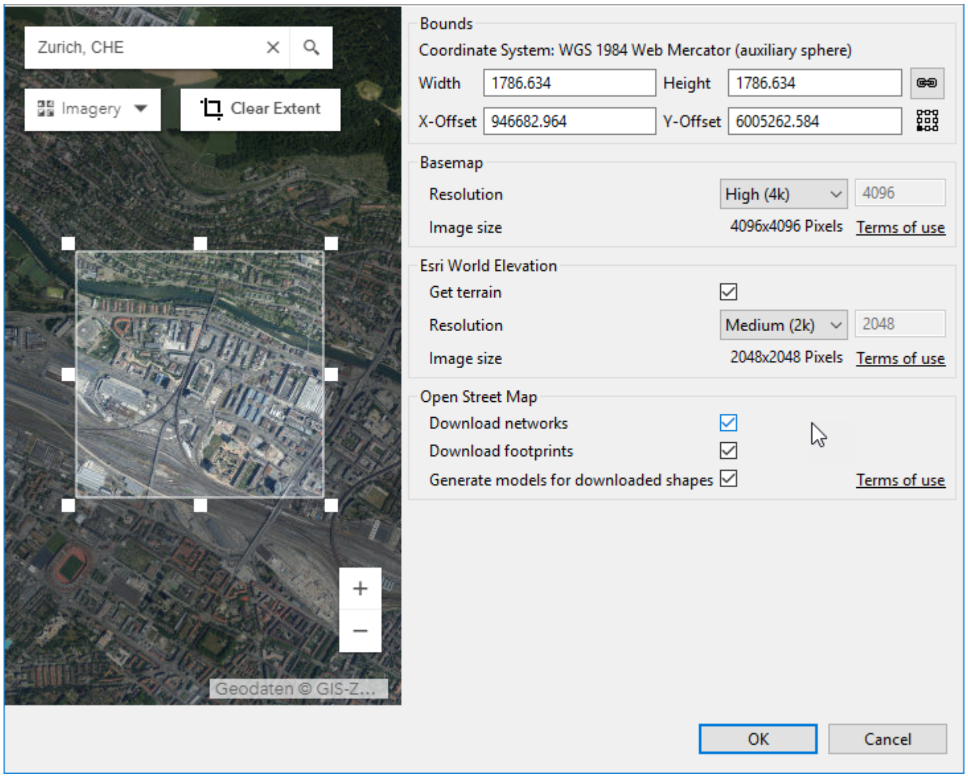


Get map data dialog box

Download data

To download map data into CityEngine, follow these steps:

1. Navigate to the area of interest.
2. Click **Set extent** to specify the extent of the downloaded data.



The extent is set for Zurich, Switzerland

You can also do any of the following:

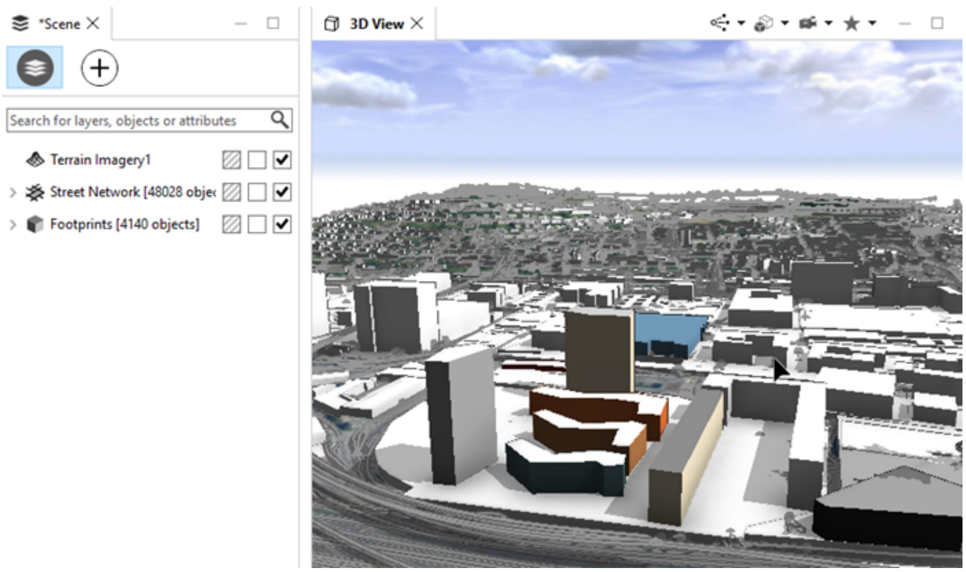
- Change the size and location by adjusting the rectangle.
- Click **Clear Extent** to remove the extent.
- Search for a location.
- Click **Select Basemap** to change the type of basemap from the drop-down menu.

Note:

- There are minimum and maximum limits for the extent.
- The settings are disabled until you set an 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 to inform you to change the location of the extent or create a new scene.

3. Specify the settings for the downloaded data. See Get map data settings for more information.
4. Click **OK**.
   - If you choose to download OpenStreetMap (OSM) data, an additional dialog box appears on which you can configure the OSM settings. When the **OSM** dialog box appears, you have the choice to accept the default settings and click **Finish** or to 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 your scene. For your convenience, default rules from ESRI.lib are assigned to the OSM data (see OpenStreetMap for more information).
- Data is saved to the `/maps` folder of your current project.



Zurich data is downloaded in a scene

Tip:

The following is some basic information to 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 your scene.
  - After running the tool, the graph networks are merged into one graph layer in CityEngine.
  - You can import the `map.osm` file located in 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, it is often needed to manually clean up the networks to improve the quality of the data.

See the Import OSM street data section in Tutorial 4: Import Streets for more information.

Get map data settings

The **Get Map Data** tool includes the following options:

Bounds	• **Coordinate System**—The CityEngine scene coordinate system. ▪ In a new scene, the **Get Map Data** tool sets the scene coordinate system to WGS 1984 Web Mercator. ▪ After the data is imported into the scene, the scene coordinate system is set to the downloaded data. ▪ In an existing scene, the **Get Map Data** tool automatically converts the downloaded data to the current scene coordinate system. • **Width**—The width of the extent. • **Height**—The height of the extent. • **X-Offset**—The x-coordinate of the selected vertex of the extent. • **Y-Offset**—The y-coordinate of the selected vertex of the extent. • —Maintains the aspect ratio when manually entering values. • —Specifies which vertex of the extent rectangle to reference for the x- and y-coordinates.
Basemap	• **Resolution**—The basemap has the following resolution options: ▪ **Low (1k)**—Low resolution. The longer side of the extent is 1024 pixels (Low resolution). ▪ **Medium (2k)**—Medium resolution. The longer side of the extent is 2048 pixels. ▪ **High (4k)**—High resolution. The longer side of the extent is 4096 pixels. ▪ **Custom**—Manually sets the longer side of the extent in pixels. The maximum size is 4096 pixels. • **Image size**—The dimensions of the extent in pixels. **Note:** The ratio of the image size dimensions matches the ratio of the extent width and height. Therefore, the image size dynamically changes as the extent size changes.

<table>
<tr>
<td>Esri World Elevation</td>
<td>
Esri provides a comprehensive elevation map of the world offering terrain at different scales. See Elevation Coverage Map for more information.

Get terrain—Include elevation data with your basemap.
Resolution—The elevation has the following resolution options:

Low (1k)—Low resolution. The longer side of the extent is 1024 pixels.
Medium (2k)—Medium resolution. The longer side of the extent is 2048 pixels.
High (4k)—High resolution. The longer side of the extent is 4096 pixels.
Custom—Manually sets the longer side of the extent in pixels. The maximum size is 4096 pixels.

Image size—The dimensions of the extent in pixels.

Note:
Elevation data is only available for subscribed users or those with an organizational ArcGIS Online account.
</td>
</tr>
<tr>
<td>Open Street Map</td>
<td>

Download networks—Include street data in your map download.
Download footprints—Include polygonal shapes such as building footprints.
Generate models for downloaded shapes—Automatically generate models from the shapes after download.
The models are generated after the Get Map Data tool applies the following ESRI.lib rules to the shapes:

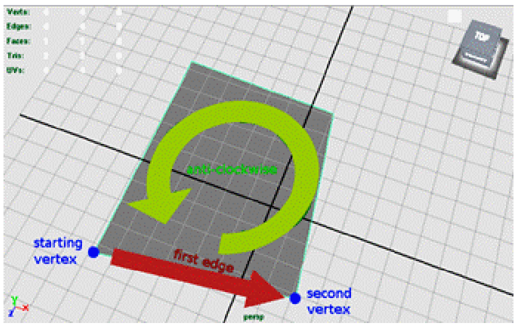
Networks—<code>Street_Modern_Standard.cga</code>
Footprints—<code>Building_From_OpenStreetMap.cga</code>

Caution:
You may experience long download times if the extent is too large.
</td>
</tr>
</table>

Prepare data for import

When using imported polygons as initial shapes for building generation, the following points should be considered when creating the source data in a third party application:

- Polygons drawn counterclockwise result in shapes facing upwards. 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 would like to create different facades for the front, side and the back, of your building.



Polygons edge orientation

- Depending on the file format, CityEngine interprets group, entity, or object names as names for start rules. Consequently, ensure to group your meshes and assign names that are valid CGA shape grammar identifiers (e.g. no special characters, space, etc.).
- Some tools might have different units for exporting. Check the exported file in a text editor to see the actual dimensions. Consequently, most import wizards offer an option to apply a uniform scaling of the imported data.

Import DAE (Collada)

Collada is an open-standards 3D geometry exchange format based on XML. Currently, it is mainly used in conjunction with KML/KMZ as an embedded container for geo-referenced 3D models. You should consider it a legacy format for all other use cases as it lacks in support for modern materials (e.g. PBR) as well as in import/export support in the major 3d apps. We recommend glTF as an alternative.

Import settings

COLLADA import has a the following options:

File	Press **Browse...** to open a file dialog to select an .dae 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	Size factor applied to the imported object.
Offset	Adds the specified offset to the imported object. • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

Import DWG (Autodesk)

Autodesk DWG is a proprietary 3D geometry format native to Autodesk AutoCAD. Even though it doesn't have modern material support it 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 try to convert unsupported objects (like polylines) using one of the following commands: `AECTOACAD`, `CONVTOMESH`, `CONVTOSOLID`, `CONVTOSURFACE`.

Import settings

The DWG import has the following options:

File	Press **Browse...**to open a file dialog to select a .dwg 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 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 factor applied to the imported object.
Offset	Adds the specified offset to the imported object: • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

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:

- as shapes: Circle, LwPolyline (must be closed), Polyline (must be either closed or polyface mesh) and Insert.
- as graph segments: Line, Arc, Circle, Polyline, LwPolyline and Insert.

The importer allows you to browse layers and individual entities contained in the DXF file. One can select the entities to import by using drag and drop.

Import settings

DXF import has the following options:

File	Press **Browse** to open a dialog to 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	In the x- and y- offset fields, one can specify an offset in order to translate the data before importing. This can be useful to center the imported data and to avoid floating point precision problems. By pressing the **Center** button, the offset of the selected DXF entities is automatically computed. **Note:** When changing the set of objects in the entity listings, the offset may also change. Therefore, it may be necessary to press **Center** again.
Scale	A scale factor that is applied to the selected entities.
Entity Listings	In the container on the left side all entities contained in the DXF file are listed. Entities that can be imported as shapes can be moved to the container bottom right. All entities in this container are imported as shapes. Entities can be moved between and inside containers by using drag and drop. **Note:** Each entity is marked with an icon on the left side, indicating if this entity can be imported as graph segment, shape, or both. • Graph only • Shape only • Graph or shape • Entity cannot be imported To delete entities on the right side press **Del**.

Graph settings

Run Generate Bridges Tool after Import	If enabled, the Generate Bridges Tool is executed on a following wizard page.
Run Simplify Graph Tool after Import	If enabled, the Simplify Graph Tool is executed on a following wizard page.

Run Graph Cleanup Tool after Import	Depending on the DXF data it may be necessary to cleanup the graph segments after import. If enabled, the graph cleanup tool is executed on a following wizard page.
Create Street/Intersection Shapes from Graph	If enabled, the shape creation parameter of the graph nodes and segments will be enabled and street shapes are created.
Create Block/Lot Shapes from Graph	If enabled, the shape creation parameter of potentially created street blocks will be enabled and shapes are created. **Note:** Presets can be saved and applied.

Import FBX (Autodesk)

Autodesk FBX is a proprietary 3D geometry exchange format with a freely available SDK. It has recently gained popularity again as the main exchange format for the Unreal and Unity game engines and related AR/VR applications. It is currently lacking in terms of modern material support (e.g. PBR) and metadata transport, but efforts are underway by Autodesk to improve this. For non-AR/VR use cases, we recommend glTF instead.

Import settings

FBX import has the following options:.

File	Press **Browse...** to open a file dialog to 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	Size factor applied to the imported object.
Offset	Adds the specified offset to the imported object. • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

Import FileGDB (Esri File Geodatabase)

The Esri File Geodatabase (FileGDB) is a file-based database for vector and raster data. It can be identified as folder with the suffix .gdb. For example, `myDatabase.gdb`

It is a file-based database with support for many GIS data types such as points, lines, polygons, 3D geometry (multipatch), raster and more. It is the recommended way to exchange GIS data between Esri applications such as CityEngine and ArcGIS Pro.

Import settings

The FileGDB import has the following options:

File	Press **Browse** to open a dialog and browse to the .gdb directory you want to import.
Layer Listing	If the file entry contains a valid gdb path, the upper part of the wizard page shows the layers available for import. These are the available columns in the FileGDB inspector: • **Layer**: Name of the Layer. The checkbox will decide if the layer is actually imported. • **Alias**: Additional name to reference the data; it is often the more commonly known name. • **Type**: If the layer is a feature class, its geometry type will be displayed. Else, the layer type will be displayed.See below for a list. • **Count**: If a feature class or a table, the number of rows will be displayed. • **Readable?**: A layer is readable (= importable), if it is a feature class with non-zero row count and has a supported coordinate system. If you hover the mouse pointer over the sign, a tooltip will tell you the reason for the layer not being readable. • **CS Authority**: Displays the coordinate system EPSG authority ID. • **CS Description**: Displays the coordinate system description.

The following layer types are supported:

- Point
- Polygon
- Polyline
- Multipatch (with textures)
- Table (indirectly, if connected via 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 sign, and will not be imported.

Note:
If a FileGDB is imported into a new scene without any coordinate system yet, the scene will inherit the coordinate system of the first imported layer. All following layers with differing coordinate systems will be 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 settings

Run Generate Bridges Tool after Import	If enabled, the Generate Bridges Tool is executed on a following wizard page.
Run Simplify Graph Tool after Import	If enabled, the Simplify Graph Tool is executed on a following wizard page.
Run Graph Cleanup Tool after Import	Depending on the FileGDB data it may be necessary to cleanup the graph segments after import. If enabled, the graph cleanup tool is executed on a following wizard page.
Create Street/Intersection Shapes from Graph	If enabled, the shape creation parameter of the graph nodes and segments will be enabled and street shapes are created.
Create Block/Lot Shapes from Graph	If enabled, the shape creation parameter of potentially created street blocks will be enabled and shapes are created.

Feature attribute settings

Import and map attributes	• If enabled, all non-geometry attributes of a feature will also be imported. • This mapping controls the width of the street shapes generated from the graph center lines. In the default behaviour, the object attribute `width` is used to determine the resulting street width, and defaults to 8 if no object attribute is found. **Note:** The function code can be edited after import in the inspector when selecting the imported layer. Advanced users can edit the default mapping code by changing the cga code in the file `gdb.ceattr`, located in `/ce.lib/rules/`.
Import database scheme and relationships	If enabled, attributes from tables that are connected to a selected feature class by relationship class(es) are also imported and assigned to the imported shapes as object attributes. **Note:** Each object attribute will keep information about its FileGDB data type (including "Domain") and original related table. This information can be used in the FileGDB exporter to recreate feature classes, relationship classes and tables.

Import textures	If the feature class is of type "Multipatch", its textures will also be imported and assigned to the scene shapes. Each texture will be extracted from the feature class and saved as a jpg or png (in case of transparency). These new texture files will be placed in a new project folder named 'data/[FileGDB name without extension]-data'.
Use selection query and spatial envelope	If the option is 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, their alias and datatype.
SELECT * WHERE	A list with all layer fields can be obtained by hovering the mouse over the layer name. Use these names to build a SQL query for filtering the imported shapes. For example: `SELECT * WHERE edits = 'yes'`
Selection Envelope	Set the size (Width, Height) and reference point (X-Offset, Y-Offset) of the selection envelope.

Working with FileGDB object attributes

- Attributes of features are imported along with the feature itself. After a successful import, these attributes appear in the **Inspector** in 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 as displayed in the **Inspector**. The shape inspector will display 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 will control the generation of models. See also object attributes.

Import glTF (Khronos Group)

glTF (version 2.0) is 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. It is currently a recommended exchange format for new projects. For more details, refer to https://github.com/KhronosGroup/glTF.

Import settings

glTF import has the following options:

File	Press **Browse...** to open a file dialog to select a .gltf or .glb 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	Factor applied to the imported object.
Offset	Adds the specified offset to the imported object: • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

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 settings

IFC import has the following options:

File	Press **Browse...** to open a file dialog to select an .ifc 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** 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	Size factor applied to the imported object.
Offset	Adds the specified offset to the imported object. • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

Import KML / KMZ (Keyhole Markup Language)

KML (Keyhole Markup Language) is an XML based exchange format to describe GIS features on the globe. It has support for points, lines and polygons and is also able to reference 3D models through Collada. It is very popular to produce data for ArcGIS Earth or Google Earth.

KMZ is a compressed version of KML where all referenced Collada and texture files are contained in one handy zip like file.

Within 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', 'Scale'. In a Model the georeference is stored in the `Location` attribute consisting of 'longitude', 'latitude' in degrees and 'altitude' in meters. The geographical coordinate system is always the 'WGS84'.

Note:
When pressing the **Finish** button and the current CityEngine scene's coordinate system is not yet set, a dialog will show up proposing the UTM-zone corresponding to the given longitude.

Import settings

KMZ / KML import has no options. Simply select a ".kmz" file, a ".kml" file or a folder containing one or more ".kml" files by clicking on the **Browse...** button. When importing ".kmz", only a single file may be imported at once.

Note:
For KMZ as well as KML files, the easiest way to import is by first dragging either the KMZ file(s) or the folder containing the KML file and associated files into the CityEngine workspace from your file manager (e.g. Windows Explorer), and then importing it from there. KMZ as well as KML importers work on the CityEngine workspace.

Import OBJ (Wavefront)

Wavefront OBJ is a text-based legacy 3D model exchange format. Despite its severe limitations in efficiency and features (e.g. no modern material support, no geometry instancing) it remains popular due to its simple syntax and manual editability.

Import settings

OBJ import has the following options:

File	Press **Browse...** to open a file dialog to 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	Size factor applied to the imported object.
Offset	Adds the specified offset to the imported object. • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

Import OSM (OpenStreetMap)

OSM files are XML based and typically used to export an extent of the OpenStreetMap GIS service into other applications. An OSM file typically contains geo-referenced descriptions of streets, blocks, parcels and points. OSM is a continually evolving open standard with a strong community.

- **Nodes** The dots that are used for drawing segments between.
- **Ways** An ordered list of nodes, displayed as connected by line segments in the editor.
- **Closed Ways** Closed ways are ways which go in a complete loop. They are used to describe areas like buildings, parks, lakes or islands.

By default, **Ways** and **Closed Ways** are converted into graph segments. However, if a **Closed Way** contains one of the tags `amenity, area, boundary, building, geological, historic, landuse, leisure, natural, place, shop, sport, tourism`, it's converted into a shape with the respective symbols, and displayed on the left side of the ways.

Import settings

The OSM import has the following options:

File	Press **Browse** to open a dialog to select a .osm file to import.
Element Listing	Lists the layers and OSM ways contained in the selected OSM file. Select the element you want to import. **Note:** • OSM files may contain a large number of layers, of which the "highway" layer is usually the most interesting to create street networks in the CityEngine. • You may select multiple OSM layers per import session. Then, all ways which are converted into graph segments will be merged into one graph layer. If you would like to generate several graph layers out of one OSM file, please repeat the process accordingly.
Select / deselect all	Selects / deselects all layers.

Graph settings

Map OSM tags	If enabled, street and sidewalk widths are mapped from tags contained in the osm file. See below.
Run Generate Bridges Tool after Import	If enabled, the Generate Bridges Tool is executed on a following wizard page.
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 Simplify Graph Tool after Import	If enabled, the Simplify Graph Tool is executed on a following wizard page.
Run Graph Cleanup Tool after Import	Depending on the OSM data it may be necessary to cleanup the graph segments after import. If enabled, the graph cleanup tool is executed on a following wizard page.

Create Street/Intersection Shapes from Graph	If enabled, the shape creation parameter of the graph nodes and segments will be enabled and street shapes are created.
Create Block/Lot Shapes from Graph	If enabled, the shape creation parameter of potentially created street blocks will be enabled and shapes are created.

OSM Tag Mapping

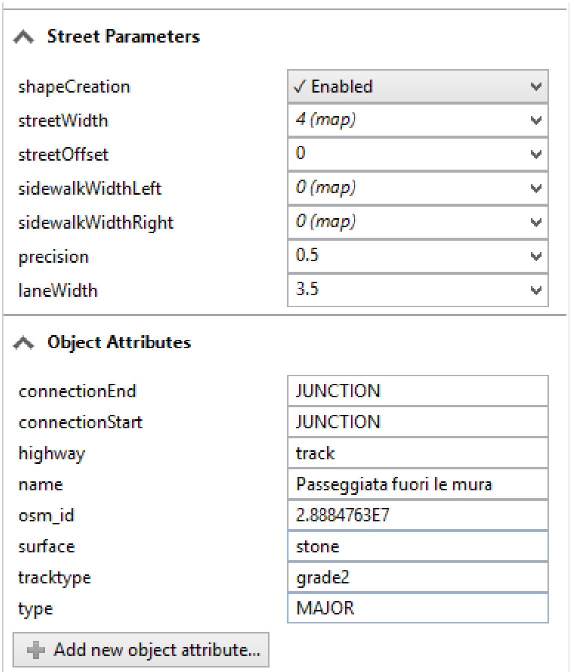
If **Map OSM tags to street widths** is checked, the created layer will contain the following layer attribute code:

Layer Attributes

```
//-------------------------
// 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

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.



Polygons edge orientation

Note:

- The function code can be edited after import in the inspector when selecting the street layer.
- Advanced users can edit the default mapping code by changing the cga code in the file `osm.ceattr`, located in `/ce.lib/rules/`.
- Presets can be saved and applied.

Import SHP (Esri Shapefile)

Shapefile is a legacy format of Esri to describe geo-referenced GIS features (points, lines, polygons and limited multipatches). It has limited support for GIS attributes and should be considered superseded by Esri FileGDB.

Import settings

The shapefile import has the following options:

File	• Press **Browse** to open a dialog to select a .shp file to import. • After opening a file, the dialog detects the shape type of the file. The type is displayed in the header of the dialog (e.g. "Shape Type: POLYGON. Importing shapes.."). The following types can be imported as shapes: Polygon, PolygonZ, PolygonM, Multipatch, Point, PointZ. Other types are imported as graph segments. **Note:** • Shapefiles containing points can also be imported. In this case, a marker (0.1x0.1m quad) is created for each point. • Shapefile polygons containing "negative" polygons that cut holes into polygons are not supported yet, and will be imported as normal shapes instead of "holes".
Coordinate System	• The importer reads the .prj projection file of the shapefile. If successful, the corresponding coordinate system is displayed in the dialog. Otherwise, a pop-up dialog asks the user to choose a coordinate system. • Please make sure 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.

Using Attributes from SHP file	• A SHP file has an accompanying .dbf file that contains associated attributes for elements. After importing the SHP file, these attributes appear in the **Inspector** in the **Object Attributes** tab.
	• To use these attributes with the CGA grammar you need to declare CGA attributes with matching names. A simple CGA rule file could look like this:
	`attr height = 10`
	`Lot --> extrude(height)`
	• After assigning this rule file to the shapes, the CGA attribute `height` appears in the **Object Attributes** tab of the **Inspector**.
	Note:
	The Source field is set to `Object`, which denotes that the CGA `height` attribute is controlled by the object attribute of the lot.

Graph settings

Run Generate Bridges Tool after Import	If enabled, the Generate Bridges Tool is executed on a following wizard page.
Run Simplify Graph Tool after Import	If enabled, the Simplify Graph Tool is executed on a following wizard page.
Run Graph Cleanup Tool after Import	Depending on the shapefile data it may be necessary to cleanup the graph segments after import. If enabled, the graph cleanup tool is executed on a following wizard page.
Create Street/Intersection Shapes from Graph	If enabled, the shape creation parameter of the graph nodes and segments will be enabled and street shapes are created.
Create Block/Lot Shapes from Graph	If enabled, the shape creation parameter of potentially created street blocks will be enabled and shapes are created.

Map shapefile attributes

If enabled, an imported graph layer will contain 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 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
```

Map shape attributes

This mapping controls the width of the street shapes generated from the graph center lines. In the default behaviour, the object attribute `width` is used to determine the resulting street width, and defaults to 8 if no object attribute is found.

Advanced users can edit the default mapping code by changing the cga code in the file `shp.ceattr`, located in `/ce.lib/rules/`.

Import USD (Universal Scene Description)

The Universal Scene Description (USD) importer reads USD assets as static models or start shapes. It supports the file extensions ".usd", ".usda", ".usdc", and ".usdz" and recognizes the USD node types "mesh" and "cube". The importer can read scenes consisting of more than one USD file, see the USD exporter file structure for more information. Additionally for static models, if the USDPreviewSurface material type is present, they are displayed as CityEngine PBR Material.

Import settings

The USD import has the following options:

File	Press **Browse...**to open a file dialog to select a .usd, .usda or .usdc 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 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	Factor applied to the imported object.
Offset	Adds the specified offset to the imported object: • **Center**: Sets the offset such that the object is centered on the scene's origin. • **Reset**: Sets the offset back to zero.

Exporting

Export data overview

Exporting is all about bringing your work out of CityEngine into other 3D authoring tools, publishing it on ArcGIS Online, or wrapping it into a user experience on the web or in VR. Each layer type can be exported into a variety of data types.

- To cover the most common use cases for exporting in CityEngine, the menu **File** > **Export Models...** has been created for you. It lets you export CGA Models, manually drawn shape-models, static models, as well as terrain meshes into a variety of file formats.
- Additional export methods can be found under **File** > **Export...** This dialog covers all use cases of **Export Models...** as well.

The following table gives an overview of what types data can be created from the different CityEngine layer types:

Layer type	Export data types
Model	Meshes
Map	Images, meshes
Graph	Lines
Shape	Meshes

Note:
- Exporting projects and scenes to share data with another CityEngine user or locally between workspaces is not part of this section.
- 360VR Experiences are a special type of export format that can cover entire scenes and are therefore independent of layer types.

Export models

Model export overview

Exporting models consists of these basic steps:

1. Select the objects in the **Viewport** or **Scene Editor** you want to export.
2. Click **File** > **Export Models...** in the main menu or press `Ctrl + E`.
3. Select the preferred file format.

Note:
The model exporter is completely independent of the already generated models in the current scene, and can therefore export arbitrarily large scenes. This means that you do not have to generate a scene prior to model export.

Supported formats and typical usage

3D Formats	Features/Typical Usage
ABC (Alembic)	Alembic is an open, interchange framework designed for VFX content production pipelines.
DAE (Collada)	Loads in a large number of DCC tools and rendering engines with support for asset-instancing, layered-textures & file-referencing.
DATASMITH (Unreal)	The Unreal Engine (Datasmith) exporter provides simple data transfer from CityEngine into the Unreal Engine Editor. It supports modern materials (PBR) as well as geometry instancing.
DWG (Autodesk)	Autodesk DWG is a proprietary 3D geometry format native to Autodesk AutoCAD. Even though it doesn't have modern material support it is widely used in the BIM and CAD industries.
FBX (Autodesk)	Provides export into Autodesk Maya, 3ds Max, MotionBuilder and other DCC tools equipped with an FBX importer. It Includes support for layered textures (multi-textures) and per-texture UVW transformations (scale, translation, rotation).
glTF (Khronos Group)	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.
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.
OBJ (Wavefront)	Transfer models with solid coloring or single texture layers to any DCC tool or rendering engine. The Material definitions are exported into accompanying MTL files. It does not support instancing.
USD (Universal Scene Description)	USD is able to efficiently and flexibly store very large CityEngine scenes with minimal loss of information which makes it suitable as an interchange format for further processing in VFX pipelines.

3D Formats	Features/Typical Usage
VOB (e-on Vue)	Native object format for polygonal meshes of e-on Software's Vue. CityEngine supports Vue 8.5 and higher. We recommend using the latest version.

3D GIS Formats	Features/Typical Usage
FileGDB (Esri File Geodatabase)	Common file format for GIS workflows. Export to multipatch/multipoint/point/polyline geometry types into a File-Geodatabase. Textures are supported.
KMZ/KML (Keyhole Markup Language)	Exports into georeferenced earth browsers like ArcGlobe or ArcGIS Earth; such models are shared e.g. in Google Warehouse.

ArcGIS Online formats	Features/Typical Usage
MSPK (Mobile Scene Package)	An Esri Mobile Scene Package is a custom, web-optimized file type to share complete scenes on the ArcGIS Platform.
SLPK (Esri Scene Layer Package)	Esri Scene Layer Package (.slpk) is a custom, web-optimized format that can be shared and published on ArcGIS Online and viewed with the Web Scene Viewer.
3VR (360 VR Experience)	Exports panoramic photos of CityEngine scenes.
3ws (CityEngine Web Scene)	CityEngine Web Scenes can be viewed in a web browser using CityEngine Web Viewer. **Note:** The 3ws format is deprecated and may not be supported in future releases of CityEngine.

Material formats	Features/Typical Usage
CGA Material	The material descriptions of a CGA model are collected and written into individual subdirectories. Each subdirectory then contains the whole material including the used textures and can simply be copied into e.g. a folder in assets. The name for the subdirectory and the .cgamat file is the material.name attribute (but made unique if there are several different materials with the same name). The <matName>.cgamat file is a CSV file, same as the result of the CGA function `getMaterial(used, changed)`.

Customized exports	Features/Typical Usage
Python (Script Based Exporter)	Allows for execution of arbitrary Python commands during batch export.

General export options

These options are available for all export file formats. Certain formats contain additional settings which will be described in the corresponding section.

Note:

- In the **Export Wizard**, each export option widget displays a tooltip with a description.
- See below for a comparison of the different formats with respect to material and shading features.

General settings

Output Path	Path to the export location. The path must exist.
Base Name	The base name of the exported files. Various suffixes will be appended depending on the other export settings.
Export Geometry	• **Models with Shape Fallback** (Default): If model generation fails, it will export the start shape geometry • **Models**: Ignore the shape, if the model generation fails. • **Shapes**: Only export the start shape geometry.
Terrain Layers	• **Do not export any terrain layers** • **Export all visible terrain layers** • **Export all selected terrain layers (Default)** • **Export all terrain layers**
Simplify Terrains	Uses the reduceGeometry operation to simplify the terrain geometry prior to export. This can take a long time, especially on higher resolution terrains.

Granularity settings

File Granularity	• **One file as long as Memory Budget is not exceeded**: Depending on the memory budget we write one or multiple files. • **One file per start shape**: Exports one geometry file per shape.

Memory Budget	Maximum memory consumption in megabytes. The geometry data is written into a single file if **Memory Budget** is zero. Otherwise a new file is written to disk whenever the memory budget is reached. Remarks: • The geometry size is measured on the raw meshes before any optimization. Therefore, the file size is usually smaller than the setting. • The check is performed per input shape not per rule output shape, i.e. very complex rules may generate more data than the limit setting.
Mesh Granularity	• **Do not merge any meshes**: No meshes will be merged, each mesh will be optimized individually. • **Merge meshes by material**: All meshes with the same material properties will be merged and optimized. • **Reuse asset instances, merge generated meshes by material** (Default): Inserted assets/meshes (see CGA insert operation) will be preserved and instanced. Meshes generated by the grammar will be merged by material.
Feature Granularity	• **One Feature Per Shape**: Creates one feature per start shape in the package. • **One Feature Per Leaf Shape**: Creates one feature per leaf shape in the package.

Geometry settings

Vertex Indexing	Choose between indexed polygon vertices or separated vertex copies per polygon.
Vertex Normals	• **Write vertex normals**: Vertex normals will be written whenever present on the mesh. • **Do not write any vertex normals**: Vertex normals will never be written even if present on the mesh. • **Write face normal if vertex normal is missing**: If the mesh does not contain vertex normals the face normals are taken for the vertex normals. • **Always write face normal**: Ignore existing vertex normals present on the geometry and replace them by the face normals.
Normals Indexing	Choose between indexed polygon normals or separated normal copies per polygon.

Texture Coordinates	• **Do not write any UV's**: No texture coordinates will be exported. • **Only write first layer of UV's**: Only the first layer of texture coordinates (uv-set 0 corresponding to the colormap) will be exported (see Texturing: essential knowledge). • **Write all UV layers**: All layers will be exported.
Local Offset	Local offset is only active when **File Granularity** is set to **One File per start shape** as it is computed for each individual shape. • **None**: Disable local offeset. • **Model Centroid**: Use centroid of the generated model as offset. • **Model Centroid Bottom**: Use centroid projected on bottom bound of the generated model as offset. • **Shape Centroid**: Use centroid of the shape as offset. • **Shape Centroid Bottom**: Use shape centroid projected on bottom bound as offset.
Global Offset	Global offset for generated geometry: Set offset for x, y and z axes (Cartesian coordinate values).
Vertex Precision	These values can be used to reduce the floating point precision of different geometry data prior to the actual file output (used to reduce file size in OBJ).
Merge ... within precision	Coordinates (vertices/normals/texture coordinates) which have an Euclidean distance equal or less than precision will be merged and their corresponding indices will be updated.
Triangulate Meshes	Convert all faces to triangles.
Faces with holes	• **Write as holes** (available for some formats only): Write the actual hole information. • **Triangulate faces with holes**: Triangulate all faces containing holes. • **Discard holes**: Discard the hole information (ignore holes). • **Convert holes to faces**: Convert the holes to actual faces (creates coplanar faces).

Materials settings

Include Materials	If enabled, material definitions (e.g. diffuse color) and textures will be included in the export. Otherwise, the default material will be referenced (if required by the format).

Texture settings

Collect Textures	All referenced textures will be exported into the export target folder and the file references will be adapted.

Create Texture Atlas	Creates texture atlases which combine a set of textures into one, thus reducing the number of textures and materials. All textures except color maps are removed. See CGA material attribute. **Note:** In case of repetition (i.e. uv coordinates outside [0,1]), textures can not be put into an atlas.
Texture Atlas Max Dimension (2^n)	Specifies the maximal dimension of the texture atlases in power-of-2 pixels, e.g. 11 gives a 2048x2048 Atlas.
Texture Atlas Border	Add a border of pixels that repeat the texture content to protect against mipmapping artifacts.

Advanced settings

Write Log	Write log file with detailed information and statistics about this export session. In case of error, check the log file to find out what went wrong.
File Type	Choose among Binary or Text (ASCII) based file creation.
Embed Textures	Choose to reference the file path (Default) or embed the texture files within the binary file.
Shape Name Delimiter	This character will be used to resolve clashes in the shape names. Example for delimiter "_": {"shape", "shape", "shape"} will be resolved to {"shape", "shape_1", "shape_2"}. Already existing suffices like "_1" will be recognized. **Note:** The name resolver can also be called manually on a selection of shapes through **Edit** > **Make Names Unique...**.
Existing Files	• **Overwrite existing files**: No file check is performed prior to writing the geometry files, all files are overwritten. • **Skip existing files**: Existing files (geometry,material and texture files) are not overwritten.

Other settings

Script	Workspace path to python script for additional python execution parallel to export.

Formats feature comparison

The features in this table are listed with respect to the actual output of CityEngine, not the theoretical features of the individual format. Please also check the list with known interoperability limitations and issues in the Format Recommendations.

Format	Instancing	Shaders	Multi-Textures	Tex Trafo	Triangulation	Referencing
ABC	Yes	No 2	Yes	Yes	Yes	No
DAE	Yes	No	No	Yes	Yes 1	Yes

Format	Instancing	Shaders	Multi-Textures	Tex Trafo	Triangulation	Referencing
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
3ws	No	No	No	Yes	No	No

- **1** Triangulation is necessary when exporting to ArcGIS Earth (see Format Recommendations).
- **2** Alembic does not store any shaders, but it is possible to use CGA reports to write out a shader name as property and then bind it in a downstream tool.
- **3** The glTF format itself only allows triangulated geometry.
- **4** CityEngine always passes triangulated geometry to DATASMITH.

Feature descriptions

Instancing	Support for instancing of geometry. Typically, the assets would be stored separately and in unmodified state from the actual nodes and transformation data.
Shader	Support for custom shader names.
Multi-Textures	Support for multiple texture channels.
Tex Trafo	Support for separate texture coordinate transformations independent from the texture coordinates stored in the meshes (eg. texture rotation).
Triangulation	Does the exporter support optional triangulation upon export?
Referencing	Does the format support referencing of entities between files? This allows for the definition of a primary file to combine all scene elements.

Export ABC (Alembic)

The Alembic exporter efficiently exports geometry and meta-data (materials, CGA reports, object attributes, ...) 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

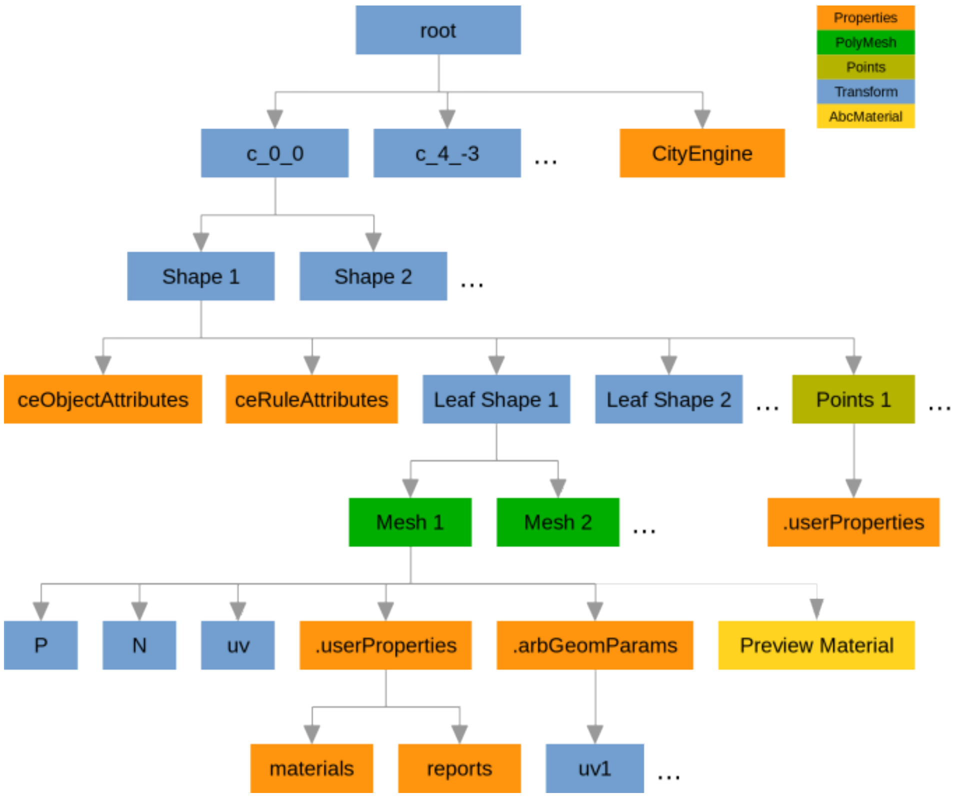
The CityEngine Alembic exporter has been designed with these goals in mind:

- Focus on compatibility with established tools like Side Effects Houdini/Mantra, Autodesk Maya, The Foundry Katana, Pixar Renderman, Solidangle Arnold, ChaosGroup VRay.
- Export of very large models with complex rules while retaining shape granularity (no merging of geometry). This allows for edits (e.g. move, scale) on individual objects (e.g. buildings) in downstream tools without having to re-export the whole scene.
- Preservation of instances, i.e. inserted assets not touched by CGA operations are exported as separate nodes. This allows for limited editing (translate/scale) of these assets in downstream tools.
- Direct export of all material properties and CGA report values on each mesh. This allows the customization of materials using CGA reports in downstream tools.

Export options

The Alembic exporter shares most export options with the other formats, see general export options for details. The only exception is the file name. As Alembic efficiently handles large files (multi GB), the exporter will always write a single file per run.

Alembic node hierarchy created by CityEngine



Alembic node hierarchy in CityEngine

Mappings from Alembic entities to CityEngine entities (refers to figure above, from top left to bottom right):

- The **"c_0_0, ..."** transform nodes are cells of a sparse grid derived from the exported scene. The cell size is adjustable in the export wizard.
- The **"Shape 1, Shape 2, ..."** transform nodes are created from CityEngine scene objects ("initial shapes"). The transformation stores the translation defined by the global/local offset options.
- The **"CityEngine"** property node is unique and receives the current user name and CityEngine version.
- The **"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.
- The **"Leaf Shape 1, Leaf Shape 2, ..."** transformation nodes are created to hold the geometry output of CGA leaf shapes. For uncached meshes (i.e. geometry purely created by CGA) one such node is created with the CGA leaf shape name. For instanced assets, one transform node is created for each mesh.
- The **"Points 1, ..."** nodes receive all single vertex geometries from a shape (e.g. generated by scatter() or comp(v)). One points node for each unique material/report combination will be created. Note: only reports will be attached as user properties.
- The **"Mesh 1, Mesh 2, ..."** nodes are created for each set of polygons with the same instance transformation (or none, in case of uncached geometry) and the same material and CGA report values. The unique material/report combination is useful to customize materials with report values.
- The **"P"** (points), **"N"** (normals) and **"uv0"** (first texture coordinate set) store the actual polygonal geometry.
- The **".userProperties"** property node is used to store CGA materials and reports per mesh or points node.

- The **".arbGeomParams"** property node is used to store additional texture coordinate sets (Alembic only supports one set on the mesh itself).
- The **"Preview Material"** node is of type AbcMaterial and stores the current material properties in the (experimental) Alembic Preview Material v2.1 schema.

Export DAE (Collada)

Collada is an open-standards 3D geometry exchange format based on XML. Currently, it is mainly used in conjunction with KML/KMZ as an embedded container for geo-referenced 3d models. You should consider it a legacy format for all other use cases as it lacks in support for modern materials (e.g. PBR) as well as in import/export support in the major 3d apps. It is recommended to use glTF as an alternative.

Export settings

See the general export options.

CGA mapping to Collada and examples

The following table lists the mapping of the major CityEngine elements to Collada elements.

Note:
The instancing/multi-texture behavior of Collada is similar to FBX.

CityEngine

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/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/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/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 Engine)

The Unreal Engine (DATASMITH) exporter provides efficient data transfer from CityEngine to the Unreal Engine Editor. The main purpose of this exporter is for architectural and design visualization workflows, but will obviously also work for games and other tasks.

Note:
- The Unreal exporter is only available on Windows.
- To use the Unreal exporter in CityEngine, at least the Unreal Engine Launcher needs to be installed as a prerequisite.
- The CityEngine Model Loader Project is not needed anymore and will eventually be removed from the Unreal Engine Launcher.

Export settings

In addition to the general export options, the Unreal Engine DATASMITH has the following options:

Export Twinmotion Compatible	Makes the Datasmith export compatible with Twinmotion. The export will use the “Unreal Base Materials” which are compatible with both Twinmotion and Unreal (the **Use Unreal Base Materials** export setting will be set to `true`). It will also disable features which are not compatible with Twinmotion: • Texture atlasing • Instancing options • Exporting Terrains • Exporting Metadata • Exporting LODs • Exporting scenarios
Global Offset	Global offset for generated geometry for x, y and z axes (Cartesian coordinate values).
Mesh Merging	Different options on how the resulting meshes are merged: • **Per Initial Shape**: Meshes are merged by initial shapes. • **Per Initial Shape by Material**: Meshes are merged by initial shapes by material. This will result in one mesh per material for each initial shape. • **Globally**: Meshes are merged globally. • **Globally by Material**: Meshes are merged globally by material.

Instancing	Different options on how instancing is handled: • **Disabled**: Instancing is disabled. • **Use Instanced Meshes**: Instancing is enabled which results in meshes being shared between Actors if possible. • **Use Instanced Mesh Actors**: Instancing is enabled, and instances are added using hierarchical instanced static mesh (HISMC) actors.
Metadata	Metadata can be exported when the Mesh Merging mode is set to either **Per Initial Shape** or **Per Initial Shape by Material**. Metadata export options: • **All**: Write both object attributes and reports to object metadata. • **Attributes**: Write object attributes to object metadata. • **Reports**: Write generated report data to object metadata. • **None**: Do not include object attributes nor report data. The exported metadata is attached to the root object of each initial shape in Unreal.
Terrain Layers	Terrain layers can be exported to Unreal Engine Landscapes . The following options are supported: • **Do not export any terrain layers** • **Export all visible terrain layers** • **Export all selected terrain layers** • **Export all terrain layers**
Use Unreal Base Materials	Uses the "Unreal Base Materials" as parent for all exported materials. These predefined materials are compatible with both Twinmotion and Unreal. They support the following CityEngine material attributes: • `material.colormap` and `material.color` • `material.normalmap` • `material.opacitymap` and `material.opacity` • `material.emissivemap` and `material.emissive.{r|g|b}` • `material.roughnessmap` and `material.roughness` • `material.metallicmap` and `material.metallic`
Use Texture Atlases	Enables texture atlases that combine exported textures and reduce the number of materials. Texture atlases typically have the biggest impact when used on slower hardware, for example, on mobile VR devices such as the Oculus Quest.
Export LOD	Enables the export of Level of Detail (LOD). Exporting LODs is only enabled when exporting Per Initial Shape.

<table>
<tr><td>LOD Attribute</td><td>The name of the CGA attribute which specifies the LOD. The attribute needs to have an Enum annotation. For each element of the Enum annotation a derivation will be triggered where the LOD attribute will be set to this element value.

The following example
<pre>@Enum(“low”, “medium”, “high”)
attr Lod = “low”</pre>
generates three LODs per Initial Shape.

Note:
If the attribute is defined in multiple CGA files it needs to have the same elements (LODs).</td></tr>
<tr><td>LOD Order</td><td>The semantic order of the LOD Attribute.

• Ascending: The lowest level of detail is defined first.
• Descending: The highest level of detail is defined first.

For example, if the LOD Attribute is defined as follows:
<pre>@Enum(“low”, “medium”, “high”)
attr Lod = “low”</pre>
the order is ascending, as the lowest level of detail is defined first.</td></tr>
<tr><td>Export Scenarios</td><td>Enables the export of CityEngine Scenarios to Unreal Engine Variants. Each scenario can be toggled individually whether it should be exported as a Variant or not.</td></tr>
<tr><td>Default Objects</td><td>Defines how default objects are exported.

• From Selection: Only the selected default objects are exported.
• Visible: All default objects which are visible in the scene are exported and the selection is ignored.</td></tr>
</table>

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 translated to Unreal by passing CGA material attributes to an Unreal Material Instance which is based on a certain primary (or parent) material.

By default the exporter generates appropriate primary materials which handle the translation 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, e.g. `/Game/Materials/CityEngineMaterials/M_CE_MyMaterial`.

Each CGA material attribute is exported and can be accessed in Unreal by 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	Note
`material.color`	Color	Vector3	
`material.opacity`	Opacity	float	
`material.reflectivity`	Reflectivity	float	
`material.shininess`	Shininess	float	
`material.bumpValue`	BumpValue	float	
`material.ambient`	Ambient	Vector3	
`material.specular`	Specular	Vector3	
`material.colormap`	ColorMap	Texture2D	
`material.dirtmap`	DirtMap	Texture2D	
`material.specularmap`	SpecularMap	Texture2D	
`material.opacitymap`	OpacityMap	Texture2D	
`<none>`	OpacitySource	float	For OpacitySource value: • 0: Use color channels (gray level) from OpacityMap • 1: Use alpha channel from OpacityMap
`material.bumpmap`	BumpMap	Texture2D	
`material.normalmap`	NormalMap	Texture2D	
`<none>`	IsPBR	float	• 1: if `material.shader` is set to `CityEnginePBRShader` • 0: Otherwise
`material.metallic`	Metallic	float	
`material.roughness`	Roughness	float	
`material.emissivecolor`	EmissiveColor	Vector3	
`material.metallicmap`	MetallicMap	Texture2D	

CGA attribute	Unreal parameter name	Unreal type	Note
`material.roughnessmap`	RoughnessMap	Texture2D	
`material.occlusionmap`	OcclusionMap	Texture2D	
`material.emissivemap`	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:
The Unreal Engine sorts transparency per actor. Exporting overlapping transparent actors will lead to rendering artifacts. Masked binary opacity is not affected by this limitation.

Default material attribute translation

The default primary materials translate the CGA material attributes roughly to a physically based Unreal Material. If the `material.shader` is set to `CityEnginePBRShader` (this happens automatically if for example an inserted glTF model is exported), the default primary materials in Unreal will 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. Even though 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, DWG has the following option:

Flip vertical axis of textures	Flips the textures along the vertical axis. This is necessary as Autodesk AutoCAD and Autodesk TrueView employ different orientations.

Export FBX (Autodesk)

Autodesk FBX is a proprietary 3D geometry exchange format with a freely available SDK. It has recently gained popularity again as the main exchange format for the Unreal and Unity game engines and related AR/VR applications. It is currently lacking in terms of modern material support (e.g. PBR) and metadata transport, but efforts are underway by Autodesk to improve this. For non-AR/VR use cases, we recommend glTF instead.

Export settings

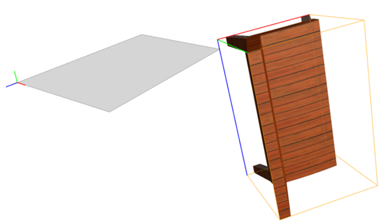
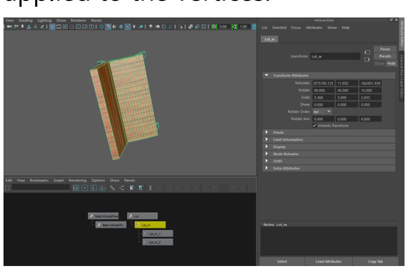
In addition to the general export options, FBX has the following options:

Create Shape Groups	If enabled, a transformation node is inserted for each shape (i.e. building). Meshes will not be merged by material across shapes.
Embed Textures	If enabled, textures are stored inside the binary FBX file.

CGA Mapping to FBX

As the FBX file format is quite verbose, here are some selected examples for this manual. Please refer to the documentation provided by Autodesk (http://www.autodesk.com/products/fbx/overview).

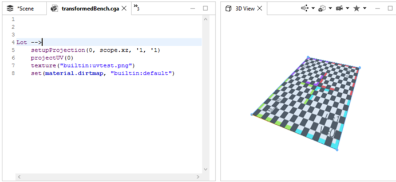
Geometry and transformation data

CityEngine
The pivot is in the origin; the scope (yellow) contains a translation and a rotation. 
FBX Import Autodesk Maya
Exported scene with FBX in Autodesk Maya. Both "One Mesh Per .." options have been disabled to avoid merging any assets. In this case, each asset is parented to a transformation node. Else, the transformation is applied to the vertices. 

Multi-texturing and layered texture nodes

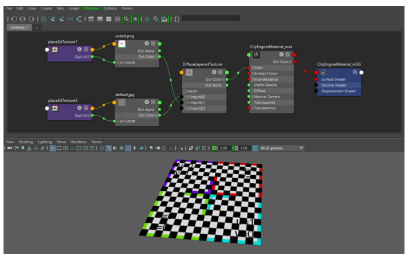
CityEngine

The CityEngine material model multiplies all color textures (color- and dirtmap, if present) with the diffuse color. This example displays the layering/multiplication of "colormap" and "dirtmap".



FBX Import Autodesk Maya

For FBX, CityEngine exports multiple textures as "layered texture nodes" whose blend modes are set to "multiply".

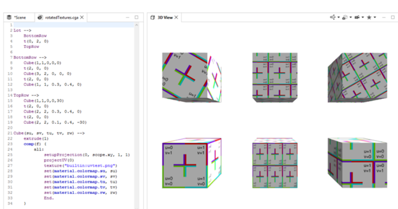


Working with per-texture transformations

CityEngine

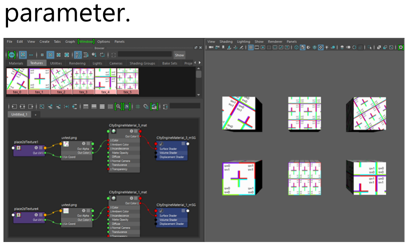
The CityEngine features material attributes to scale, translate and rotate textures independently of the actual texture coordinates stored inside the assets:

- material.{...}.tu/tv = translate/offset texture
- material.{...}.su/sv = scale/repeat texture
- material.{...}.rw = rotate texture around face normal



FBX Import Autodesk Maya

Upon FBX import into maya, CityEngine's tu and tv attributes are mapped to the "Offset" parameter of maya's place2dTexture nodes, su/sv are mapped to the "RepeatUV" parameter and rw is mapped to the "Rotate Frame" parameter.



Export FileGDB (Esri File Geodatabase)

Esri FileGDB is a file-based database with support for many GIS data types such as points, lines, polygons, 3d geometry (multipatch), raster and more. 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 settings

In addition to the general export options, FileGDB has the following options:

Geodatabase Name	The name of the geodatabase directory. The ".gdb" suffix is mandatory.
Export Features	Controls what is exported per selected scene shape. Available values are: • **Models**: Only the output of an attached CGA rule is exported. • **Shapes**: Only the geometry of the shapes is exported, any CGA rules will be ignored. • **Models and Shapes**: Both the shapes and the generated models will be exported. This usually results in more than one feature class.
Export object attributes	Exports all object attributes as feature class fields. Information about data types and domains will be re-used, if available from an earlier FileGDB import. Array values will be ignored if the next option "Export dataset relationships" is not selected and the attribute is not part of an actual relationship.
Export dataset relationships	• All (array) attributes which result from an earlier FileGDB import with relationship import will be exported to the corresponding related tables. • Due to a limitation in the underlying FileGDB API, relationship classes cannot be generated by CityEngine directly. Instead, an ArcGIS workspace xml file is generated. See importing a geodatabase schema for more information.
Emit Reports	Any reports generated by the attached CGA rule will be written into feature class fields.

In addition to the global export options, the FileGDB exporter contains an additional export page with the following per-layer settings:

Export Layer	Include or exclude any selected layers from export.
Layer Name	Adjust the name of the selected layers. This will determine the name of the written FileGDB dataset(s).

Write Strategy	Choose how existing feature classes are handled: • **Replace Feature Class**: deletes the current Feature Class and creates a new one. • **Update Feature Class**: 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. • **Update Feature Class Geometry**: only the geometry of the existing feature is updated.
Geometry Type (for Shapes)	Choose the geometry type for the shapes: • **Polygon** • **Multipatch**

About the naming of exported feature classes

The FileGDB exporter may output multiple feature classes based on a single CityEngine scene layer. For example if you export a shape "wall" of layer "building", you get the following:

- In case of **Export Features** set to **Shapes** or **Models and Shapes**, the shape `wall` will be directly written to the `building` feature class.
- If `wall` has a CGA rule attached which produces polygons, its output will result in a `building_ProcedurallyGeneratedMultipatches` multipatch feature class.
- If `wall` has a CGA rule attached which produces points (e.g. by comp(v)), its output will result in the `building_ProcedurallyGeneratedPoints` or `building_ProcedurallyGeneratedMultipoints` point/multipoint feature class.
- If `wall` has a CGA rule attached which produces edges (e.g. by comp(e)), its output will result in the `building_ProcedurallyGeneratedLines` polyline feature class.

Export glTF (Khronos Group)

glTF (version 2.0) is 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. It is currently a recommended exchange format for new projects. For more details, refer to https://github.com/KhronosGroup/glTF.

Export settings

In addition to the general export options, glTF has the following options:

GLTF Output Format	If set to glTF binary, the file is stored as a .glb file which contains all geometry and texture data. If set to glTF JSON, the file is stored as a .gltf with geometry stored in .bin files and textures as .png files.

CGA mapping to glTF

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` 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 settings

IFC uses the general export options.

Export KML/KMZ (Keyhole Markup Language)

KML (Keyhole Markup Language) is an XML based exchange format to describe GIS features on the globe. It has support for points, lines and polygons and is also able to reference 3d models via Collada. It is very popular to produce data for ArcGIS Earth.

KMZ is a compressed version of KML where all referenced Collada and texture files are contained in one handy zip like file.

Export settings

In addition to the general export options, KMZ / KML has the following options:

Altitude Mode	Controls what altitude tags are written to the kml/kmz files: • **clampToGround**: Aligns the object to the ground, altitude is ignored. • **absolute**: ignores the actual ground elevation. **Note:** For absolute ground elevation and the altitude have to coincide precisely.
Write Compressed Files	If set, the written KML file is put into a KMZ archive file along with the other needed files like the DAE file(s) and texture images.
Heading Correction	ArcGlobe, Google Earth and other earth browsers interpret the Collada DAE file's content differently: for the case of Google Earth the Heading Correction switch has to be turned on for having correct results.
Placemark Locations	• **Exact spatial references per model**: Unprojects placemark locations per shape. Recommended for unconnected objects (such as buildings) for bigger areas. • **Optimized Model placement**: Optimizes placemark locations to ensure correct relative positions (unprojection only in the lead shape, other shapes are referenced relatively). Recommended for connected objects (such as streets) over small areas.

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 exporter will write the file "MyPlacemarks.kml". Next to it, there will be a folder named MyPlacemarks.kml-files containing the associated DAE file and textures. Here is what the KML file will look like:

Example KML File 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 are custom, web-optimized files that can be shared across the ArcGIS Platform. They can be uploaded on ArcGIS Online and viewed with ArcGIS Pro as well as ArcGIS Earth.

If you want to export only parts of the scene, select them in the **3D Viewport** or the **Scene Editor**. If you want to export the whole scene you don't have to select anything. Then start the export by clicking **File** > **Export...** > **CityEngine** > **Export Mobile Scene Package** from the main menu.

Export options for MSPK

The MSPK exporter has the following options:

Output File	The file to export the mobile scene package to.
Export Content	• **Complete Scene** exports all exportable parts of the scene. • **Selection** only exports the elements selected in the Viewport or Scene Editor.
Scene Environment	Choose between local and global scene export. Local scenes are kept in the coordinate system of the scene while global scenes are converted to be viewable on the whole globe.

Upload MSPK to ArcGIS Online

1. Locate the exported MSPK file in the Navigator.
   By default the MSPK file is located in the `data` folder of your project.
2. Right-click the MSPK file and choose **Share As...**
   The CityEngine Mobile Scene Package dialog box appears.
3. Click **Upload package to my ArcGIS Online account**.
4. Name your new MSPK file.
5. Fill in the required fields in the **Item Description** tab.
6. Click **Analyze** to validate your MSPK file for any errors or issues.
   Fix the errors before continuing.
7. Click **Share** to upload your MSPK file to ArcGIS Online.

MSPK files can be uploaded to other ArcGIS platform portals as well. See Sharing data on a different portal.

Export OBJ (Wavefront)

Wavefront OBJ is a text-based legacy 3D model exchange format. Despite its severe limitations in efficiency and features (e.g. no modern material support, no geometry instancing) it remains popular due to its simple syntax and manual editability.

Export settings

- See general export options.
- If the there are issues displaying the exported model correctly in other programs, try setting "Triangulate Meshes" to `true`.

CGA Mapping to OBJ

Geometry

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. Please note, that the per-texture transformations (see material.colormap.{su, sv, tu, tv, rw}) are NOT included in these texture coordinates.
f	The mesh faces/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 will be used. If Mesh Granularity is set to "merge meshes by material", this name is controlled by the material.name attribute. Else, an internal name is set dependending on the operation which 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 Material section below). It is only written, if the "Materials" export option is enabled.
mtllib	Local reference to the corresponding MTL file. It uses the same base name as the OBJ file (but with extension '.mtl') and is written into the same directory.

Materials

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.
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, set to the value of `material.color`.

MTL element	CGA feature
map_Kd	The diffuse texture, set to the value of `material.colormap`. 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 `material.colormap.{su,sv}` are != 1.0 the -s option is appended with the scaling factors. If `material.colormap.{tu,tv}` are != 0.0 the -o option is appended with the translation values.
Ka	Ambient color, set to the value of `material.ambient`.
Ks	Specular color, set to the value of `material.specular` if the material is of type PHONG (`illum` = 4).
d	Opacity, set to the value of `material.opacity`.
map_d	Opacity map, set to the value of `material.opacitymap`.
Ns	The specular exponent of the phong lighting model, also called "shininess". Set to the value of `material.shininess`.
Tf	For Maya compatibility, set to (1.0, 1.0, 1.0).
Ni	For Maya compatibility, set to 1.0.

See also Rendering and export for more information.

Script Based Exporter (Python)

The Script Based Exporter executes a python script during export model generation, without actually writing geometry to a file.

To export to a geometry format and execute an export script simultaneously, use the desired format exporter and specify the script in the export dialog option **Script** based export (Python).

Export settings

Script	Workspace path to python script

Export SLPK (Esri Scene Layer Package)

Esri Scene Layer Package (SLPK) is a custom, web-optimized format that can be shared on ArcGIS Online and viewed with the WebScene Viewer. For more details, refer to https://github.com/Esri/i3s-spec/.

Select the content you want to export in the **3D Viewport**, and start the exporter. Click **File** > **Export Models...** > **Esri Scene Layer Package** in the main menu.

Export settings

In addition to the general export options, SLPK has the following options:

Combine Layers	Instead of writing one package per layer we write one package containing the data in all layers. If you chose to merge all layers the per layer settings will not be shown.
Emit Reports	Any reports generated by the attached CGA rule will be written into attribute fields.
File size	• **Small File — Low Detail**: Creates a small file; all selected models are exported with possible reduced quality. • **Midsize File — Medium Detail**: Creates a midsize file; exported models with medium level of detail. • **Large File — Good Detail**: Creates a large file; exported models with good level of detail. • **Huge File — High Detail**: Creates a huge file; exported models with high level of detail.
Scene Environment	Choose between local and global scene export. Local scenes are kept in the coordinate system of the scene while global scenes are converted to be viewable on the whole globe.

In addition to the global export options, the SLPK exporter contains an extra export page with the following per-layer settings:

Export	Checkboxes denote if a layer's contents are exported.
Layer Name	The layer name, as it is set in the CityEngine scene.

Export tricks and tips

- If a Esri Scene Layer created from a CityEngine export doesn't behave fluent consider reducing the size of the SLPK by
  - choosing a smaller extent for export (select less objects)
  - fine tuning **Texture quality** export options (use compact or half-sized)
  - reducing the geometric complexity of the models (e.g. lower level of detail, less details on streets).
- When preparing Esri Scene Layer Package for a wider audience, keep in mind that a Esri Scene Layer might not run as fluent (or not 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:
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 might not be properly shown in clients.

Publish SLPK on ArcGIS Online

Upload to ArcGIS Online

1. Locate the exported SLPK in the **Navigator**.
2. Right-click the SLPK and choose **Share As...**
   The CityEngine **Scene Package** dialog box appears.
3. Click **Upload package to my ArcGIS Online account** and name your new SLPK.
4. Fill in the required fields for **Item Description**.
5. Click **Analyze** to validate your SLPK 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. If any issues are discovered, an error will be reported. You have to fix the error before you can continue.
6. Click **Share** to upload your SLPK to ArcGIS Online.

SPLK files can be uploaded to other ArcGIS portals as well. See Sharing data on a different portal

Publish scene layer from SLPK

After uploading your package to ArcGIS Online or ArcGIS Enterprise you need to publish the SLPK to a **Scene Layer**. To create a **Scene Layer**, do the following:

1. Go to the **Navigator** and select **Portal** > **My Content**.
2. Find the SLPK you just uploaded and double click it.
   Your browser now opens with the correct item opened for you.
3. Click on **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 not correct.
5. Click **Publish**.
   This opens the item of the **Scene Layer** and after waiting for the service to be created you can open it.

Export USD (Universal Scene Description)

The Universal Scene Description (USD) exporter exports CityEngine scenes into a set of 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

- Focus on compatibility with tools such as SideFX Houdini and its Solaris component, The Foundry Katana, and corresponding USD-compatible renderers.
- Export of large models with complex rules while retaining shape granularity (no merging of geometry). This allows for edits (for example, move or scale) on individual objects (for example, buildings) in downstream tools without the need to export the whole scene again.
- Preservation of instances—for example, inserted assets not touched by CGA operations are exported as separate nodes. This allows for limited editing (translate or scale) of these assets in downstream tools.
- Support for PBR materials via the standard USDPreviewSurface material schema.

Export options

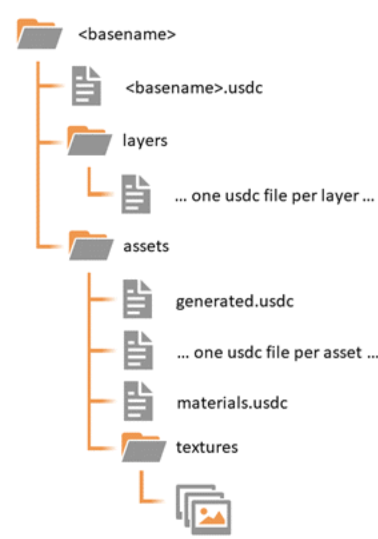
In addition to the general export options, USD has the following option:

File Type	Choose between exporting a hierarchy of USDC files and textures or a single USDZ package.

Layout of exported USD files

For each export session, the USD exporter creates a set of directories and files:

- <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 in turn 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`.



Export VOB (e-on Vue)

Native object format for polygonal meshes of e-on Software's Vue. CityEngine supports Vue 8.5 and higher. It is recommended to 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).

Export 3ws (CityEngine Web Scene)

CityEngine Web Scene (3ws) is a custom, web-optimized format that can be shared on ArcGIS Online and viewed with the CityEngine Web Viewer.

Note:
The 3ws format is deprecated and may not be supported in future releases of CityEngine.

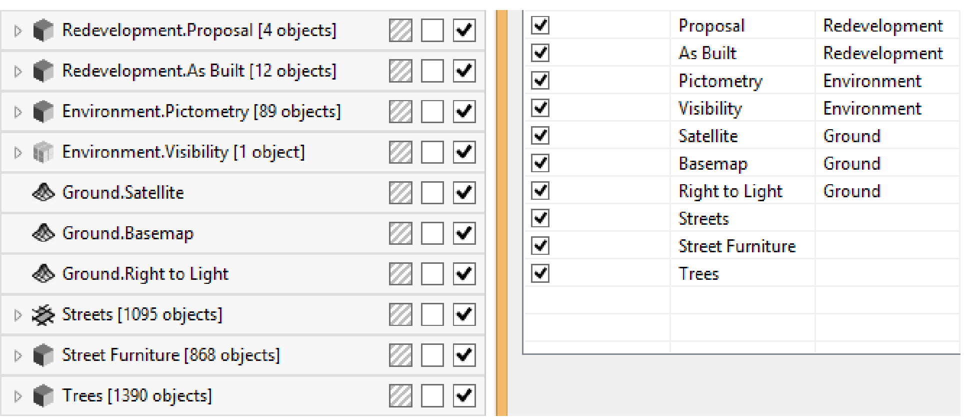
Select the content you want to export in the **3D Viewport**, and start the exporter. Click **File** > **Export Models...** > **CityEngine Web Scene**.

Export settings

In addition to the general export options, the Web Scene export has an additional export page which allows you to control settings of individual layers:

Export	Checkboxes denote if a layer's contents are exported.
Layer Name	The layer name, as it is set in the CityEngine scene.
Layer Group	Layers can be combined to layer groups: All layers with the same group name are collected in a Layer Group. In CityEngine Web Viewer, only one layer of a group is visible at a time (exclusive visibility). Furthermore, all layer groups enable Swipe View in CityEngine Web Viewer.

You can name your CityEngine scene layers to automatically have their layer group set by using the pattern GROUPNAME.LAYERNAME.



Scene Editor with group layer naming

Texture quality settings

Scene settings	Use global option set in **Textures** in first export dialog page. (Only available as per-layer option.)
High Quality	Convert all layer RGB textures and store with high quality
Medium Quality	Convert all layer RGB textures and store with medium quality
Low Quality	Convert all layer RGB textures and store with low quality

Half Sized	Resize all layer textures by 50%; RGB textures are written as medium quality JPG
Compact	Resize all layer textures to 50% size but with a maximum of 256x256 pixels; RGB textures are written as low quality JPG
Original Textures	Use original textures **Note:** Only JPG and PNG textures are supported in Web Viewer. Other textures formats will appear black
No Textures	Do not include textures in layer objects.

Individual layer settings

Layer state settings

Backdrop	Layer will not appear in layer pane, is always visible.
Visible	Layer appears in layer pane and can be set visible or hidden. Initial state is visible.
Hidden	Layer appears in layer pane and can be set visible or hidden. Initial state is hidden.

Interaction settings

Scene settings	Use **Global** option set in **Object Interaction** in first export dialog page.
Locked	Objects in this layer will not be selectable. All objects are combined to a single entity. Therefore locked layers contain a way smaller object count than pickable layers, which can often improve performance in the Web Viewer. As all objects are combined, per-object metadata can not be written (No Metadata option only).
Pickable	Objects in this layer will be selectable, all objects are exported as single entities (features). High object count in a layer can decrease performance, it is therefore recommended to only set layers to pickable where required.

Metadata settings

Scene settings	Use **Global** option set in **Object Metadata** in first export dialog page.
All	Write both object attributes and reports to object metadata.
Attributes	Write object attributes to object metadata.
Reports	Write generated report data to object metadata.
None	Do not include object attributes nor report data.

Texture Quality

Scene settings	Use **Global** option set in **Textures** in first export dialog page
High Quality	Convert all layer RGB textures to JPG with high quality
Medium Quality	Convert all layer RGB textures to JPG with medium quality

Low Quality	Convert all layer RGB textures to JPG with low quality
Half Sized	Resize all layer textures by 50%; RGB textures are written as medium quality JPG
Compact	Resize all layer textures to 50% size but with a maximum of 256x256 pixels; RGB textures are written as low quality JPG

Note:
In all **Texture Quality** modes except "Original Textures" the texture size will be limited to 2048 pixels.

Other 3ws export features

Bookmarks defined in a CityEngine scene will be exported directly to a Web Scene, and are available in the bookmarks menu of CityEngine Web Viewer.

Note:
- Different camera angles are not supported at the moment, Web Viewer displays all views and view bookmarks with the default CityEngine angle (54 degrees).
- The current camera position and direction before starting the export is stored to the Web Scene and used as initial view in CityEngine Web Viewer.

The current shadow and ambient occlusion settings of a CityEngine scene are exported to the .3ws file, and are used as initial values in the Web Viewer. Be careful with enabling shadow and ambient occlusion before exporting, as on some machines Web Viewer might not support direct or diffuse shadows, or will get a performance impact.

Note:
As initial settings are stored during export, changing options (such as shadow settings) in Web Scene preview will not change the intial configuration of the Web Scene.

Export tricks and tips

- Due to browser limitations and to ensure compatibility on less powerful systems we recommend to keep Web Scene size (size of .3ws file) below 15MB. Reduce your Web Scene size by
  - choose a smaller extent for export (select less objects)
  - Use a smaller terrain resolution (512x512 or less). Terrain resolution can be set in Inspector pane of Terrain Layer, Layer Attributes, **terrain resolution u and v**.
  - finetune **Texture quality** export options (use compact or half-sized)
  - or manually convert your textures to lower-resolution JPG or PNG and use texture quality **Original**
  - set export option **Interaction** to **locked** where pickability is not required
  - reduce the geometric complexity of the models (e.g. lower level of detail, less details on streets).
- By default, terrain geometry is simplified during export. This can take a while for complex terrains. To speed up this process,
  - use a smaller terrain resolution (recommended 512x512). Terrain resolution can be set in Inspector pane of Terrain Layer, Layer Attributes, **terrain resolution u and v**.

- disable **Simplify Terrain Meshes** in Main export options → Optimizations (not recommended, as this will increase .3ws file size)
- When setting Texture quality to **Original** Textures, make sure the layer only uses JPG and PNG textures. Other formats are not supported by WebGL, and will appear black in the Web Viewer.
- When preparing Web Scenes for a wider audience, keep in mind that a Web Scene might not run as fluent (or not at all) on other, less powerful systems (less memory, less powerful graphics card). Reduce the exported extent and 3ws file size to ensure wider compatibility.

CityEngine Web Scene Preview

Before publishing a Web Scene, it is a good idea to preview it locally to ensure it contains all the desired data and the layer setup in the exporter options are done correctly.

1. Locate your exported Web Scene (the .3ws file) in the CityEngine **Navigator** (by default Web Scenes are exported to the current project's model folder).
2. Double-click on the .3ws file or right-click the .3ws Web Scene file and choose **Open With** > **CityEngine Web Scene Viewer**
   The Web Viewer opens in your system's default web browser

Preview settings

3D Web Scene Viewer	This option opens CityEngine Web Viewer from ArcGIS Online. An internet connection is therefore required.
3D Web Scene Viewer (offline)	A local version of CityEngine Web Viewer application is used. No internet connection required. Note that the local viewer might not be as up to date as the online version.

The preview URL opened in the browser is of the form

```
http://localhost:61351/ceviewer.html?3dWebScene=/PATH_TO/WEBSCENE.3ws
```

or

```
http://localhost:61351/ceviewer_offline.html?3dWebScene=/PATH_TO/WEBSCENE.3ws
```

for the offline version. The path to webscene.3ws is relative to the CityEngine workspace (web scenes need to exist in the CityEngine workspace to be previewed). A random, varying high port is chosen automatically by CityEngine.

Due to browser security restrictions when loading local files (e.g. the .3ws file) a local webserver is required for the Web Viewer. Starting the Web Viewer with a file protocol (file://...), which happens e.g. when starting the Web Viewer by double-clicking on the viewer.html will therefore not work.

Default browser

When starting the preview Web Viewer application, CityEngine starts the web browser that is set as default browser on the computer. In case you want to preview in a different browser

- change your system's default web browser
- or copy-paste the preview URL to the desired web browser

Misc Preview notes

All initial configuration settings of a Web Scene (e.g. camera position, shadow settings, ...) are done during export. Changes done in Web Scene preview are not stored to the Web Scene, and will not influence the initial configuration.

Upload to ArcGIS Online

1. Locate the exported Web Scene in the **Navigator**.
2. Right-click the Web Scene choose **Share As...**
   The CityEngine **CityEngine Web Scene Package** dialog box appears.
3. Click **Upload package to my ArcGIS Online account** and name your new Web Scene package.
4. Fill in the required fields for **Item Description**.
5. Click **Analyze** to validate your Web Scene 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. If any issues are discovered, an error will be reported. You have to fix the error before you can continue.
6. Click **Share** to upload your SLPK to ArcGIS Online.
7. Click **File** > **ArcGIS Online** and go to **My Content** to find your Web Scene online for sharing.

Note:
If the Web Scene already exists on the portal, the webscene will be updated with the new version, and will keep the same ID.

Package Web Scene and Viewer to a zip file

Instead of publishing to ArcGIS Online you can create a zip file of Web Scene and Viewer application, which you can manually upload to your own webserver.

- Use the same steps as above but choose **Save package to file** in the Web Scene package dialog.
- The created zip file contains two folders
  - `webscenes` contains the Web Scene file .3ws
  - `webviewer` contains the Web Viewer application files

Upload to custom server

Copy both folders to your web server. To start the application, point your browser to

```
http://SERVER.DOMAIN/PATH/TO/WEBVIEWER/viewer.html?3dWebScene=../webscenes/
NAME_OF_WEBSCENE.3ws
```

Once the webviewer application is uploaded and you create additional web scenes, you only need to upload the .3ws files to the webscenes folder and adapt the URL to point to the respective web scenes via the query parameter `?3dWebScene`.

Note:
Due to browser security restrictions when loading local files (e.g. the .3ws file) a local webserver is required for the Web Viewer. Starting the Web Viewer with a file protocol (the URL starts with `file://...`), which happens e.g. when starting the Web Viewer by double-clicking on the `viewer.html` will therefore not work.

Model export application notes

Note #1: Format Recommendations

Please note that some of the listed tools require additional plugins to be able to load all formats.

Tool	Format	Mandatory Options
ArcGIS Earth	KML	
Autodesk Max	OBJ, FBX	(for obj, enable import of smoothing groups in max)
Autodesk Maya	FBX, ABC, OBJ	(Maya cannot import ABC materials out-of-the-box.)
Autodesk MotionBuilder	OBJ, FBX	
Blender	OBJ, FBX, ABC	- Multi-Texturing
Cinema 4D	OBJ, DAE	- Multi-Texturing
Deep Exploration	OBJ, FBX, DAE	
SideEffects Houdini	ABC, USD	
The 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 expensive (i.e., large) assets, it is of advantage to create simplified proxy assets and switch between them with a global LOD (Level of Detail) attribute. To avoid scattering the LOD attribute all over the rules, it is useful to put the conditions into separate "asset loader" rules:

```
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

If you plan to create large models, it is of great advantage to implement global CGA attributes into your rule sets that allow to selectively block the generation of polygon-intensive model features. For example, one could replace some high-polygon greek columns with simple cuboids by using an attribute LOD, together with a corresponding condition in the CGA rules.

Application Example

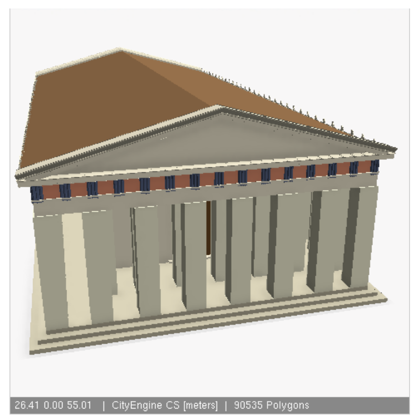
Let's assume you want to render a large scene with Maya/Arnold and you have a CityEngine scene ready with a LOD switch. By exporting the scene with `LOD = 0` to a single .obj file (without any textures) and importing it, for example, into Maya, you are able to quickly setup the lights and camera without overburdening Maya with heavy geometry. Once the environment is ready you can go back to CityEngine and export the whole scene with `LOD = 1` to USD files and link them to the render setup.

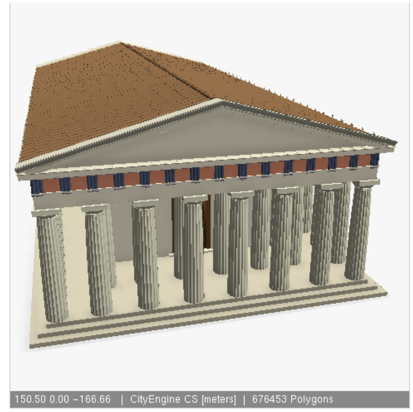
Implementation Example

Below you find a modified version of the Parthenon temple shape grammar example. Note the usage of the LOD attribute:

The temple with `LOD = Low`, ~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`, ~675k polygons	

Export terrain

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

Exporting to geometry

1. Select **Terrain Layers** to export in **Scene editor**.
2. Select **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 model exporter.

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

Exporting to tile package (TPK)

1. Select **Terrain Layers** to export in **Scene editor**.
2. Click **File** > **Export Selected Layers as TPK**.
3. Under **Maps**, choose to export both the basemap and elevation terrain files as TPK's or just select one of them.
   If you chose to export both Basemap and Elevation maps, you will have a {filename}_Basemap.tpk and a {filename}_Elevation.tpk file in your output folder.
4. For **Scene Environment**, you have the option of exporting the terrain for either a global or local scene.
5. Sign in to ArcGIS Online or ArcGIS Enterprise to share the TPK's with your organization or the public.
6. Right-click the TPK file you want to share and click **Share as...**
   This opens the share **Tile Package** dialog. .
7. Fill out the necessary fields and click **Share**.
8. 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.

Exporting to image

To export to image files, do the following:

1. Select **Terrain Layers** to export in **Scene editor**.
2. Select **File** > **Export** > **CityEngine** > **Export Selected Terrains as Image**.
3. For **Format**, choose the image format to export.

Export shapes

To export shapes to an external format, do the following.

1. Select shapes to export.
2. Select **File** > **Export**.
3. Click **City Engine Layers** > **Export Selected Shapes**.
4. Click **Next**.
5. Choose the Esri Shapefile or AutoCAD DXF format.

Tip:
There are two ways you can export shape geometry in CityEngine: (1) use the pure shape exporters, or (2) use the model exporters and choose to only export the shape geometry.

Export to SHP

The Esri Shapefile or simply a shapefile is a popular geospatial vector data format for geographic information systems 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. The shapefile exporter can export shapes and attributes to these files. Additionally, the exporter writes an index file (.shx).

Note:
The same exporter can also be used to export graph segments: For a graph segment specific description please visit: Exporting Graph Segments to SHP.

Export settings

The export dialog consists of the filename and 3D options. Presets can be saved and applied.

File

Press **Browse** to open a dialog to select a .shp file to export.

3D Options

The following table illustrates the influence of this option:

3D Option	Shapefile Type	Data
none	Polygon	2D
PolygonZ	PolygonZ	3D
Multipatch	Multipatch	3D

Note:
The exporter does not write a .prj projection file. The data is always stored in cartesian coordinates in the current scene coordinate system.

Exporting 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, the exporter writes an entity of type closed Polyline.

Note:
The same exporter can also be used to export graph segments: For a graph segment specific description please visit: Exporting Graph Segments to DXF

Export settings

DXF export doesn't have options. Choose or create the ".DXF" export file by clicking on the **Browse** button and click on **Finish**.

Export graphs

Graph segments can be exported to shapefiles, geodatabases and DXF. To export dynamic shapes and models that are based on graphs refer to export shapes and export models respectively. To export graph segments do the following:

1. Select **Graph** segments to export.
2. Select **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 will be available for export.

Export to SHP

The ESRI Shapefile or simply a shapefile is a popular geospatial vector data format for geographic information systems 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. The shapefile exporter can export graph segments and attributes to these files. Additionally, the exporter writes an index file (.shx).

Note:
- The same exporter can also be used to export shapes: For a shape specific description please visit: Export shapes to SHP
- Shapefile support is not available in all CityEngine versions.

Export settings

The export dialog consists of the filename and the 3D options. Presets can be saved and applied.

File

Press **Browse** to open a dialog to select a .shp file to export.

3D options

The following table illustrates the influence of this option:

3D Option	Shapefile Type	Data
none	Polyline	2D
PolylineZ	PolylineZ	3D

Note:
The exporter does not write a .prj projection file. The data is always stored in cartesian coordinates in the current scene coordinate system.

Export to Esri File Geodatabase

The Esri File Geodatabase (FileGDB) is a file-based database for vector and raster data. It can be identified as folder with the suffix .gdb. For example, `myDatabase.gdb`.

Export settings

The export dialog consists of the filename option. Presets can be saved and applied.

File

Press **Browse** to open a dialog to 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, the exporter writes an entity of type Line.

Note:
The same exporter can also be used to export shapes: For a shape specific description please visit: Exporting Shapes to DXF

Export settings

The export dialog consists of the filename and the street width option. Presets can be saved and applied.

File

Press **Browse** to open a dialog to 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 which is written is the total street width, that is the street width plus the sidewalk widths.

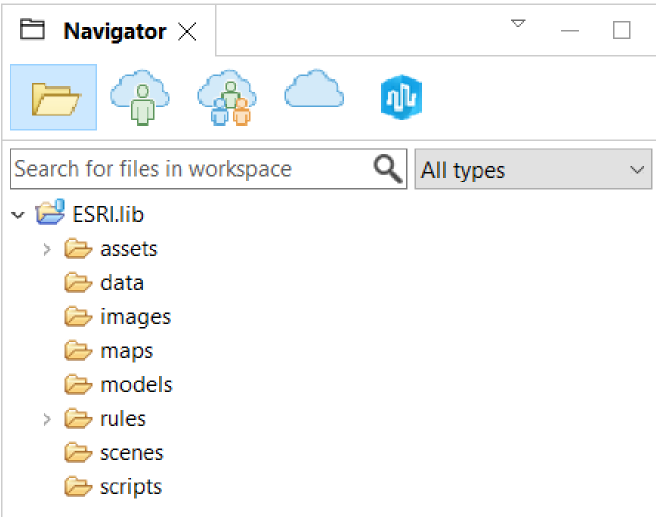
Export 3VR (360 VR Experience)

The **360 VR Experience** (3VR) is a file format to publish and consume panoramic images of CityEngine scenes, see 360 VR Experience Export.

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.



ESRI.lib in the Navigator

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 (left), OpenStreetMap data (middle), and 3D mass models (right)

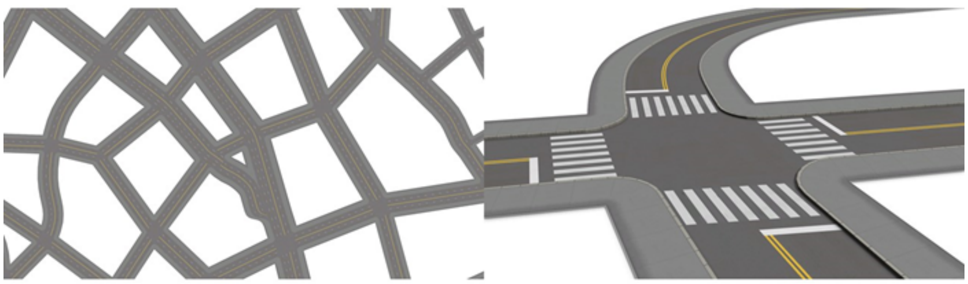
Plants

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 (left), schematic (middle), and fan (right) vegetation models

Streets

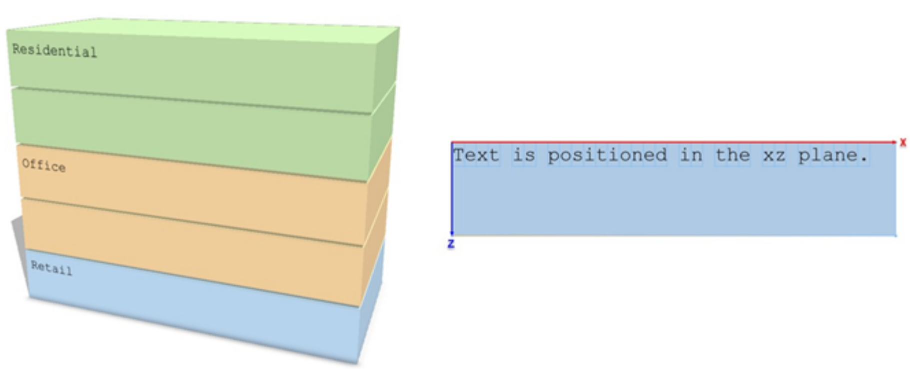
Apply rules to dynamic street shapes to generate textured streets. The `Street_Modern_Simple.cga` rule textures streets with lane markings, while the `Street_Modern_Standard.cga` rule also adds stop markings and crosswalks.



Textured streets with lanes (left) and textured streets with lanes, stop markings, and crosswalks (right)

Text

The `Text.cga` rule inserts text you can use for labeling 3D models. Text is inserted in the xz plane of the scope.

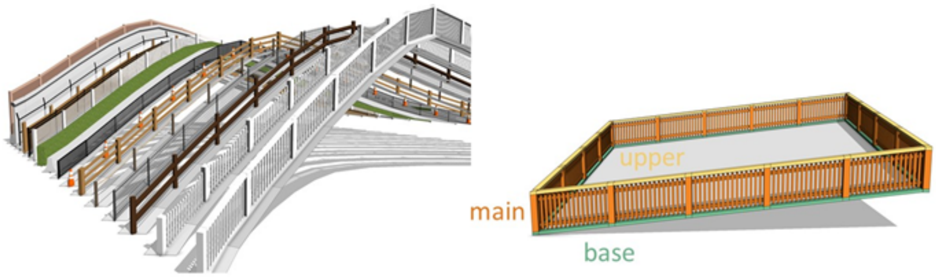


Text rule applied

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

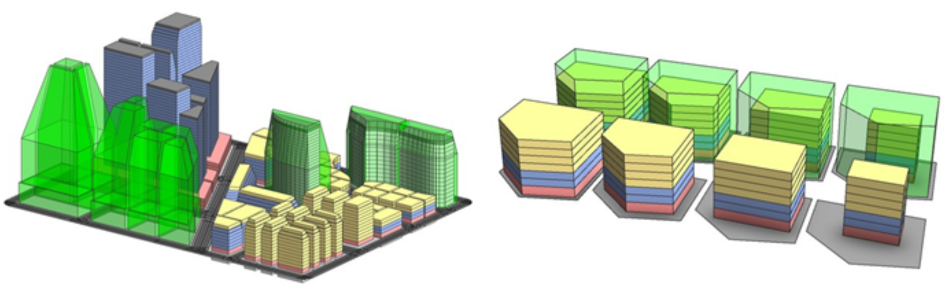
`Fence_On_Polygon.cga` to polygonal shapes to create fences along the edges of the input polygon. You can control which polygon edges have fences through attributes and local edits. Preset styles are available to create a variety of fences from picket fences to highway barriers.



Fences generated along street shapes (left) and polygon edges (right)

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.



Urban rule applied to zoning areas

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.

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 to match the versions of the `ESRI.lib` and Webstyle libraries. CityEngine usually maintains compatibility, but if the versions do not match, the rules might 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 own 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 overview

Analysing and evaluating a design is an essential part of an iterative design process to ensure that qualitative and quantitative requirements are met. CityEngine offers a variety of tools for this.

Dashboards

Visualize quantitative information about CGA models using real-time charts. See 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/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) are factors for determining critical measures when evaluating an urban design proposal. Using CityEngine **Dashboards** you can create meaningful 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**.

- **Dashboards** can consist of multiple pages which 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 clipboard.
- Cards can display data coming from different scenarios.

Check out the **Example Philadelphia 2020.0** in the main menu (**Help** > **Download Tutorials and Examples**). This scene is a good example of how scenarios, CGA reports and Dashboards interact with each other.

Configure dashboard

1. Click **Window** > **Dashboard** in the main menu to configure a **Dashboard**.
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
Report	• Choose from a list of reports created by the applied CGA rule(s) (e.g. Area by flooring type, Area by interior space type, Number of work places, etc.). • The drop-down menu has a text box which allows text filtering among the report names. • A **Key Number Card** can display the values of all reports. • The **Stack Chart**, **Bar Chart** and **Pie Chart** only display the values of group reports. To create a group of reports, you need to create reports with syntax `Group.SubReport`, for instance: ▪ `FAR.Office` ▪ `FAR.Resdential` ▪ `FAR.Commerical` ▪ etc.

Divide by	• Divide a report by another report. • By using the reportName.* notation, corresponding sub-reports (with the same sub-report name) can also be divided.
Multiply by	Multiply the values of a report by a constant number.
Reporter	You can choose to display the values of the reports computed on all the scene or only on the selection. By default a card displays the values on all the scene. • **All Objects**: Default, means values from all objects in the scene are aggregated. • **Selected Objects**: Only values from selected objects are aggregated
Aggregation	The aggregation type defines how to combine all the reported values from the whole scene into a single number for each report or sub-report, that will then be shown in the chart: • Sum • Count • Mean
Unit	The **Unit** displayed next to the value in the card. Has no influence on the calculation or the value displayed.
Minimum and Maximum (for Bar Chart)	Defines the range of the value axis. • **Automatic**: adjusts value automatically based on the displayed values. • **Custom**: Uses a fixed user-defined value.
Scaling (for Bar Chart)	Specify the scaling of the axis: • Uniform (by default) • Logarithmic
Notation	• Decimal (e.g. 48'876'592) • Scientific (e.g. 4.888e+7)
Round values	**Decimal places**: Specify number of decimals to display.
Sorting (Pie Chart)	Sorting of reports: • **Alphabetical**: (report names) • **Ascending**: Lowest value first, highest last. • **Descending**: Highest values first, lowest last.
Legend (Stack Charts and Pie Charts)	• Display a legend in charts. • The legend adjusts automatically to the reported values.

Colors

The colors of the reports in the **Dashboard** are, by default, defined by the color of the scenarios set in the **Scene Editor**. It is possible to specify a color for each report in a CGA file by the hashtag #color. For instance:

```
Lot -->
        report("report_name",1.0)
        report("report_name#color","#ff0000")
```

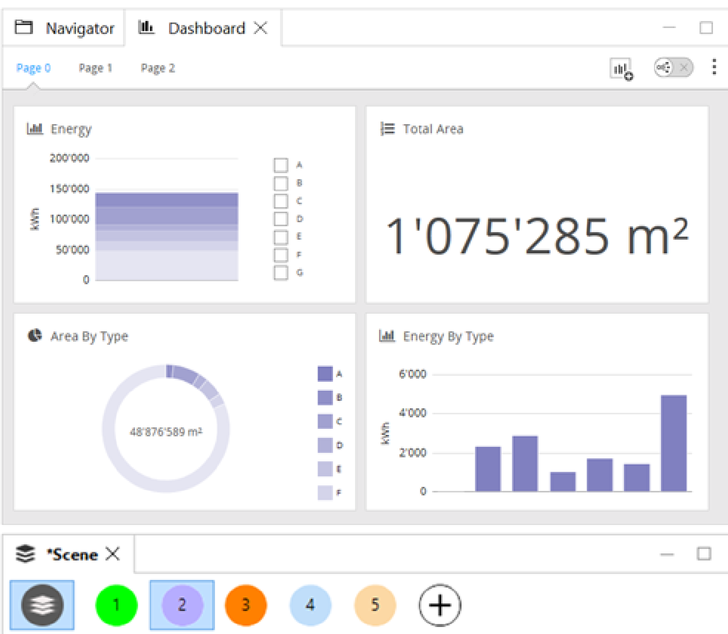
Dashboards and scenarios

Two different modes are available in the **Dashboard** concerning the data to display:

- Active Scenario
- Compare Scenarios

Switching from one mode to anther is done by clicking on 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**

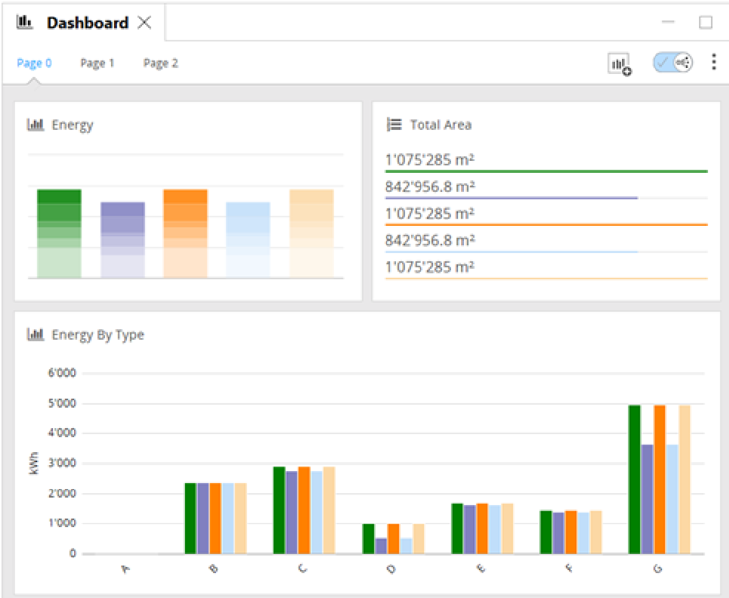
Active scenario mode



Active scenario mode

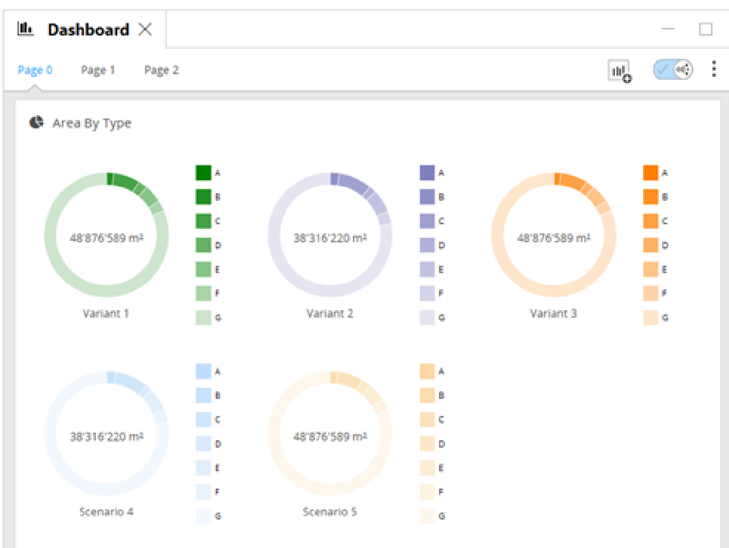
Compare scenarios mode

In the **Compare Scenarios** mode, the cards will display the reports values of all the scenarios of the scene :



Compare scenarios mode

In the **Compare Scenarios** mode, the **Pie Card** will display one pie chart per scenario :



Compare scenarios with pie charts

For more information about scenarios, see scenarios.

Manage Dashboard

It is possible to create many pages in the **Dashboard**. Click on the **Options** tool ⋮ in 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:
  - Report or Group of Reports
  - Scenario
  - All Scene or Selected Objects

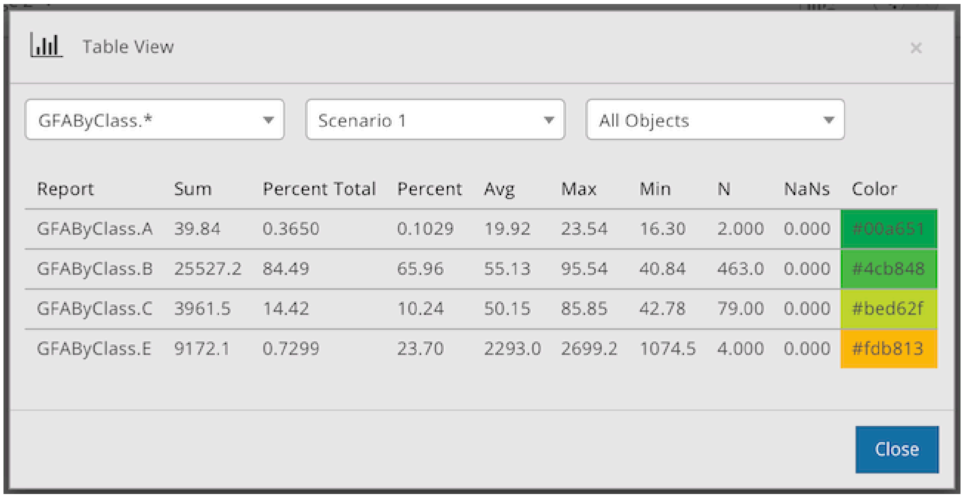
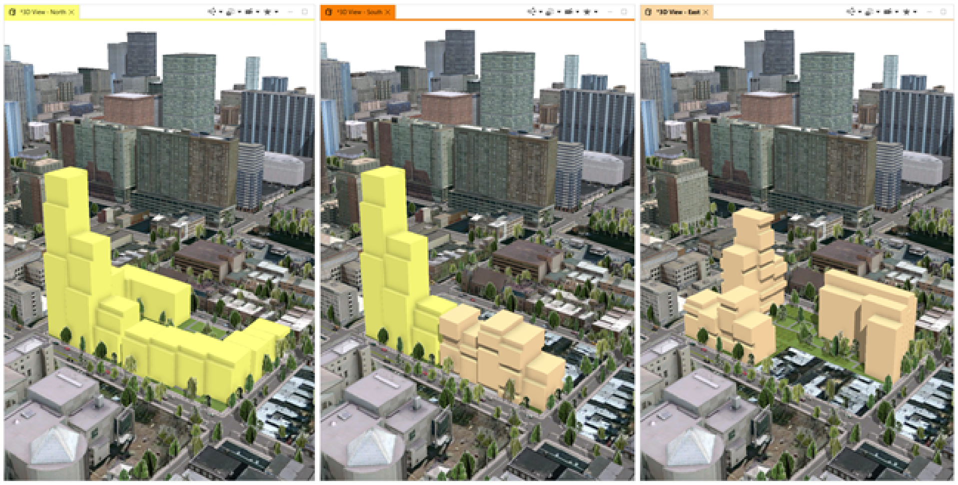


Table view

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 new scenario.
2. Provide the **Name**, **ID**, and **Color**.
3. Click **OK**.

Manage scenarios in Scene Editor

Manage scenarios in the **Scene Editor** toolbar. Change the order, edit properties, and remove scenarios. When you right-click a scenario, you get the following options:

Duplicate...	See Duplicate.
Mirror...	See Mirror.
Edit...	Change the scenario name, ID, and color.

Delete	Remove the scenario.
Move right	Move the scenario to the right.
Move left	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 new 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 the a scenario to duplicate.
2. Click **Duplicate...**
3. Provide a **Name**, **ID**, and **Color**.
4. Click **OK**.
   A new 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 active scenario in the Viewport

The active scenario is the scenario which 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.

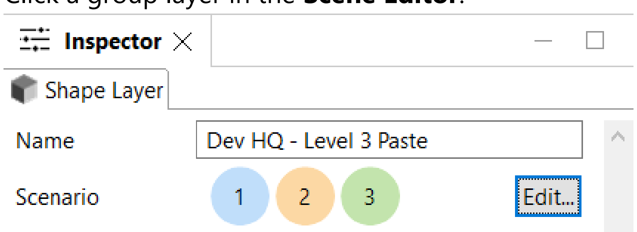
Tip:

- 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.

Managing 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 which indicates the membership status of the group layer.
   a. Under **Object Type**, choose **Default Object** or **Scenario Object**.
   b. Under **Scenarios**, choose which scenarios the group layer is a member.

When you add the membership of layer to another scenario, it has the same affect 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 *.

Measure tools

You can measure distance with the **Measure Distance** tool and areas and paths with the **Measure Area and Path** tool .

Measure distance

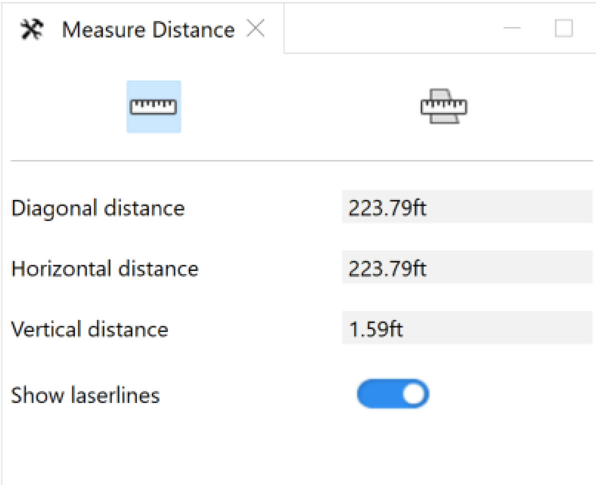
To measure distance, do one of the following:

- Click the **Measure Distance** tool .
- Press `M+D`.
- Click **Analysis** > **Measure Distance** in the main menu.

Click in the scene to place the starting point. Click again to place the end point.

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 includes the following options:

Measure Distance	
Diagonal distance	Diagonal distance between the points.
Horizontal distance	Horizontal distance between the points.
Vertical distance	Vertical distance between the points.
Show laserlines	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, do one of the following:

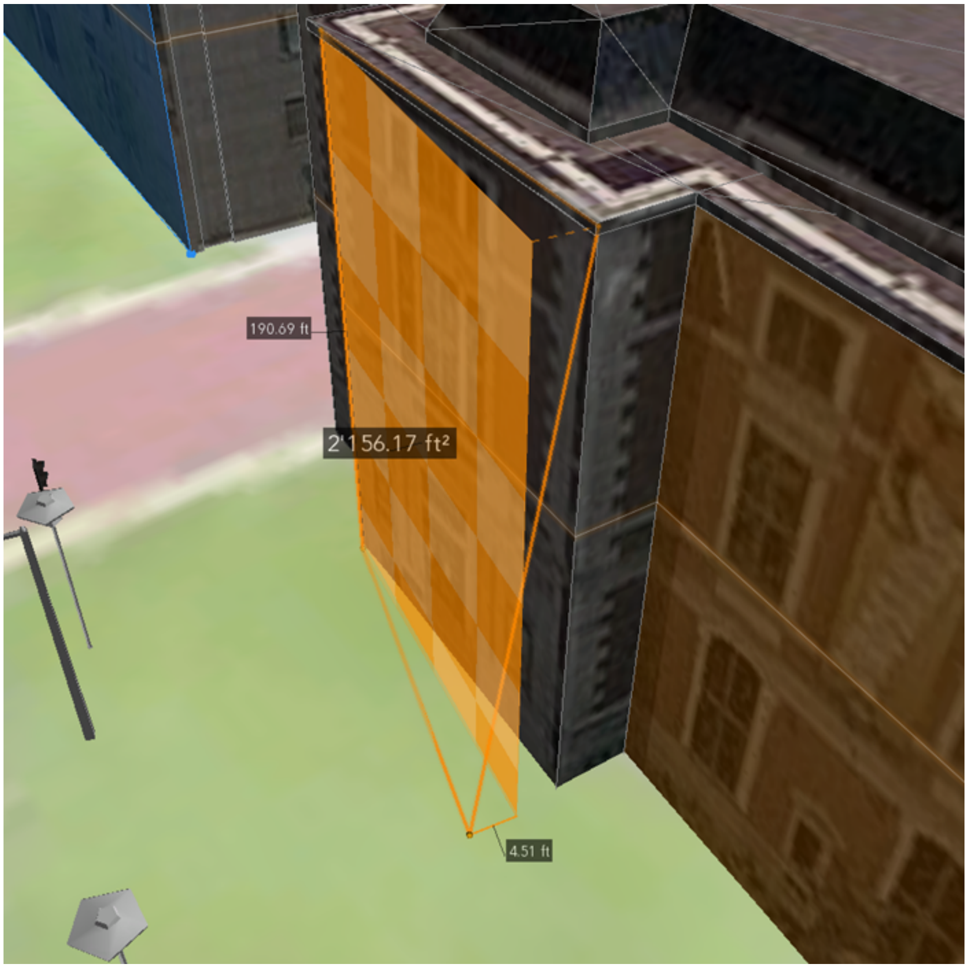
- Click the **Measure Distance** tool and then click the **Measure Area and Path** tool in the **Tool Options** window.
- Press `M+A`.
- Click **Analysis** > **Measure Area and Path** in the main menu.

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 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.

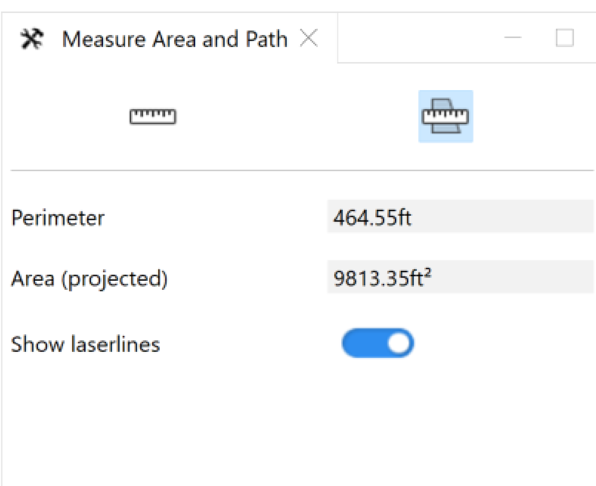
Note:

- 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 includes the following options:

Measure Area and Path

Perimeter	Perimeter length of the polygon.
Area (projected)	Area of polygon.
Show laserlines	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 cursor 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 overview

Visibility analysis tools highlight surfaces and structures that are visible or hidden from an 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**.

Viewshed Creation	Calculates a viewshed that determines the visibility from a camera-like observer for a limited field of view.
View Dome Creation	Calculates a view dome that has the same functionality as the **Viewshed Creation** tool but offers a 360° field of view.
View Corridor Creation	Creates a protected view corridor 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 Manage analysis layers. The visibility analysis tools snap to buildings and terrains to facilitate the positioning of the observer point and the point of interest.

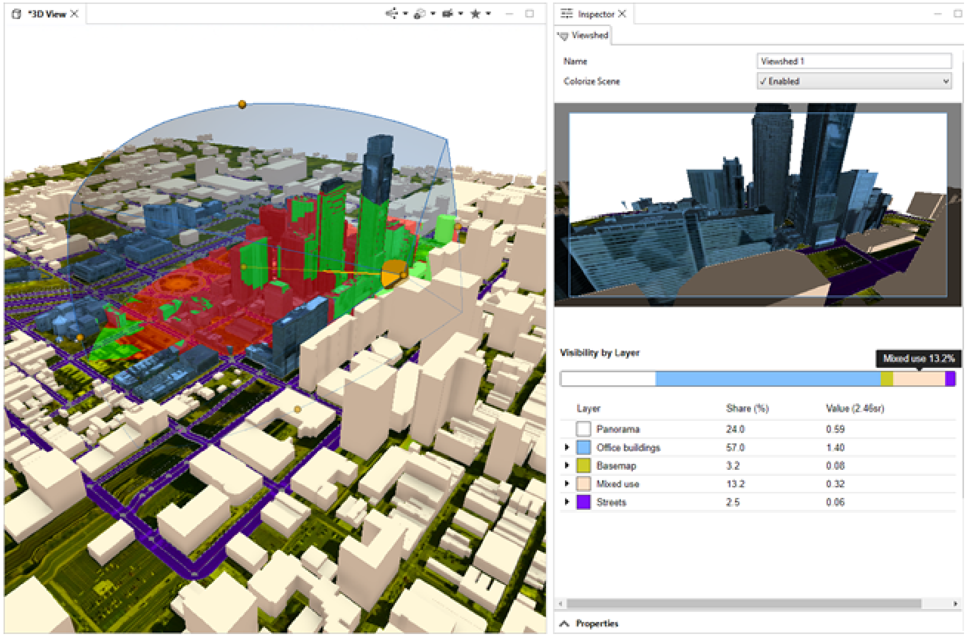
Common features

- 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 it self is edited, the recalculation is triggered automatically for all visibility objects.
- The visual representation in the Viewport offer 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 offers a perspective view rendered from the observer position.
- The **Visibility settings** tool to show or hide all analysis objects.

Visibility by layer

The **Viewshed Creation** and **View Dome Creation** tools offer 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 offers 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 non-geometry areas in the field of view including the sky and areas below the horizon not covered by terrain or geometry.

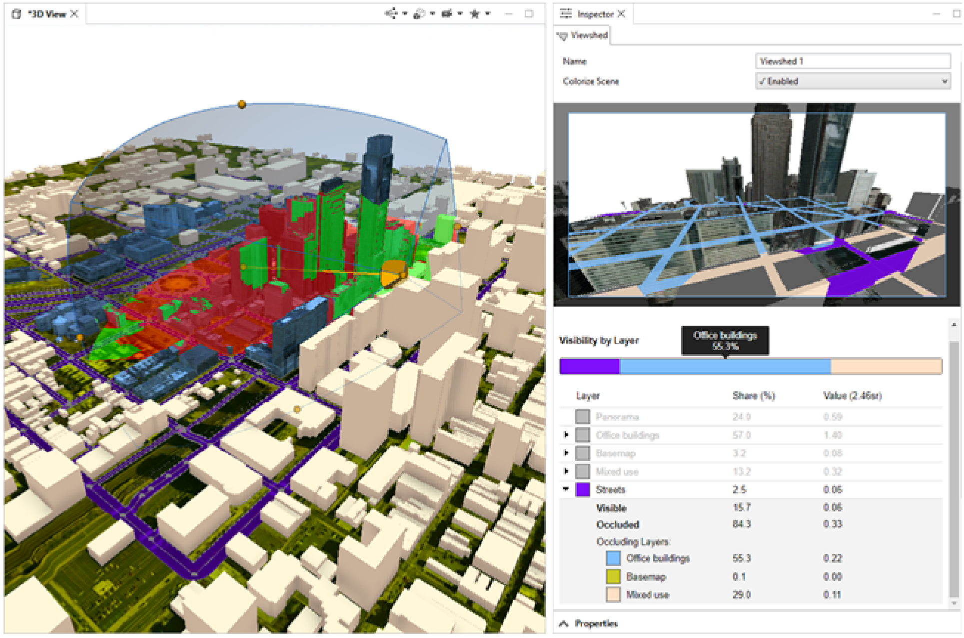


Viewshed in the Viewport with colored geometry and the Inspector with the bar chart and the table view of the statistics.

Note:
Only geometry inside the viewshed or view dome body is taken into account for the calculations. Geometry further 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`.

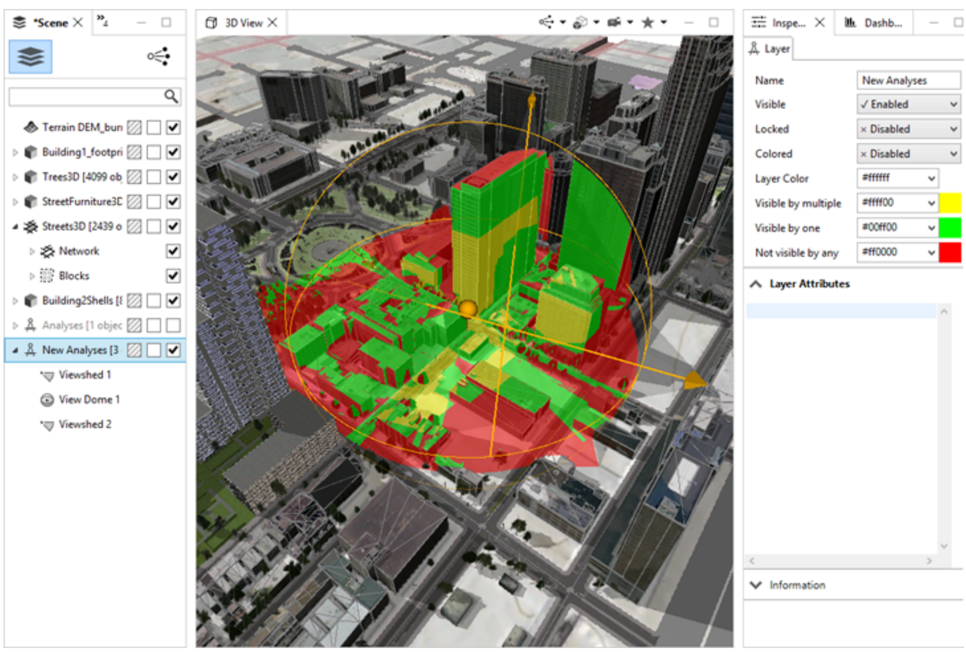
For further analysis, you can expand a layer to see how much of it is visible or occluded within the viewshed or view vome. Furthermore, you get a list of layers that contribute to the occlusion of the expanded layer including percentages and sr values. In the image below, 15.7% of the `Streets` layer is visible and 84.3% 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%.

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 layer Buildings

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 to 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 when you want 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 then be managed in the Scene Editor.

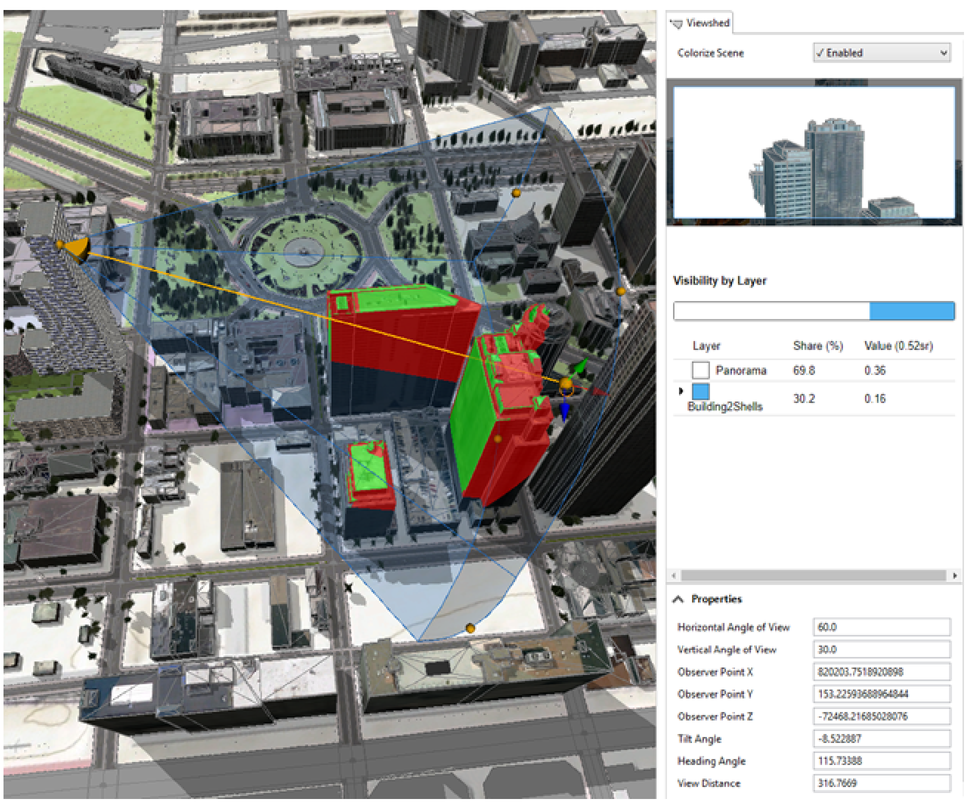
You can open the **Viewshed Creation** tool the following ways:

- Click the **Viewshed Creation** tool .
- Click **Analysis** > **Viewshed Creation** in the main menu.

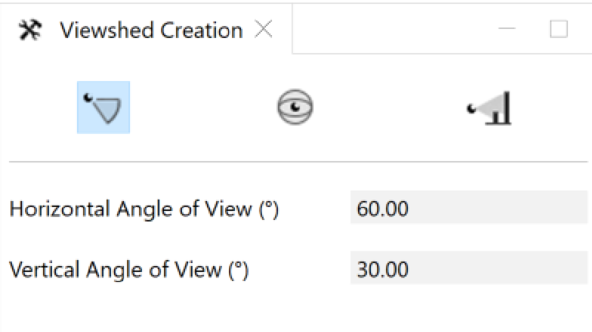
Create a viewshed

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 includes the following options:

Viewshed Creation	
Horizontal Angle of View (°)	The value determines the horizontal angle of view for the next view corridor placed. Once placed, you can adjust the viewshed.
Vertical Angle of View (°)	The value determines the vertical angle of view for the next view corridor placed. Once placed, you can adjust the viewshed.

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:

Horizontal Angle of View	Horizontal angle of view, or field of view, from observer
Vertical Angle of View	Vertical angle of view, or field of view, from observer
Observer Point X	X-coordinate of observer
Observer Point Y	Y-coordinate of observer
Observer Point Z	Z-coordinate of observer
Tilt Angle	Camera view angle from -85 to 85 degrees
Heading Angle	Camera view angle from -360 to 360 degrees
View Distance	Distance between observer and point of interest

Tip:

The **Colorize Scene** property in the Inspector allows you to choose whether to display the colorized 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 more details.

View domes

The **View Dome Creation** tool allows you to see the visibility from a given point in a 360 degrees 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 then 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 you want to display the colorized visible or hidden areas of the view dome in viewports. The preview shows you the 360 degrees 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 the properties in the **Inspector**:

Observer Point X	x coordinate of observer
Observer Point Y	y coordinate of observer
Observer Point Z	z coordinate of observer
View Distance	Distance between observer and point of interest

Tip:
The **Colorize Scene** property in the **Inspector** allows you to choose whether you want to display the colorized 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 more detail.

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.

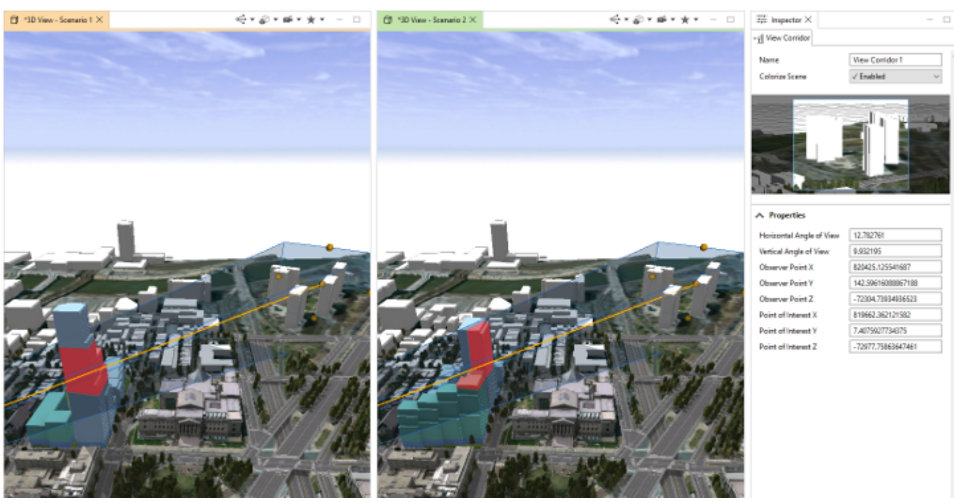
You can open the **View Corridor Creation** tool the following ways:

- Click the **Viewshed** tool and then 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:

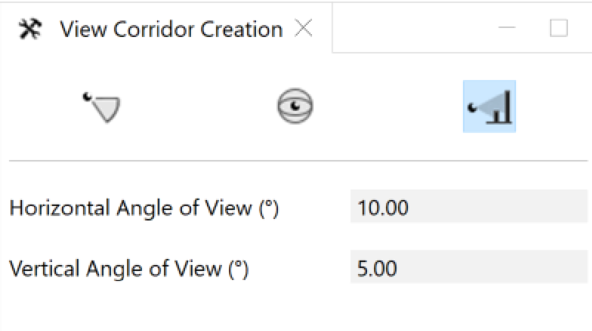
1. Click a point in the Viewport to create an observer point.
   This starts the **View Corridor Creation** tool, which dynamically displays the view corridor as you move.
2. Drag the tool to set the point of interest.
3. Click again to anchor the **Point of Interest** property in the view corridor.
   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 colorized by the **View Corridor Creation** tool.

Tool Options



The **View Corridor Creation** tool includes the following options:

View Corridor Creation	
Horizontal Angle of View (°)	The value determines the horizontal angle of view for the next view corridor that is placed. Once placed, you can adjust the view corridor.
Vertical Angle of View (°)	The value determines the vertical angle of view for the next view corridor that is placed. Once placed, you can adjust the view corridor.

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:

Horizontal Angle of View	Horizontal angle of view, or field of view, from observer
Vertical Angle of View	Vertical angle of view, or field of view, from observer
Observer Point X	X-coordinate of observer
Observer Point Y	Y-coordinate of observer
Observer Point Z	Z-coordinate of observer
Point of interest X	X-coordinate of point of interest
Point of interest Y	Y-coordinate of point of interest
Point of interest Z	Z-coordinate of 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

Online and Enterprise overview

ArcGIS Online and ArcGIS Enterprise are online platforms to share maps and geographic information with others. You can share all kinds of geographic information: maps, map layers, features, editing templates (using layer packages), imagery, and analytic results. Once you signed up on 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.

Get map data

Get map data from ArcGIS Online to add basemaps to your CityEngine scene. You can choose from several types of basemaps and select an area to download into your 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 walks 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 in the CityEngine toolbar. If you don't have an ArcGIS Online account, see get new account to create a new public account.

Note:
To access **All Portal** items in the **Navigator** and share your data with your organization and public, sign in to a **Portal**.

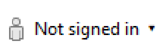
Main menu

To sign in from 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 in exactly the same way 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:

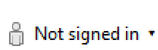
1. Click the **Sign in** menu 

 in the toolbar.
2. Click **Sign in** to open the **ArcGIS Sign In for CityEngine** dialog.
3. Enter your username and password.
   The username is case sensitive, enter it in exactly the same way 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 

 in the toolbar.
2. Click **Manage Portals**.
   The **Portals** dialog under **Preferences** opens.

3. Click **Add new portal...** .
   This opens the **Add Portal** dialog.
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 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 in the **Preferences** dialog, do the following:

1. Right-click the portal and select **Set as active portal** to make the portal active.
2. Click **OK**.
   After you added the portal(s), CityEngine will remember your connected portals the next time you launch the application.

Switch active portal

You can also switch your active portal from the **Sign in** menu in the toolbar. To switch the active portal, do the following:

1. Click the **Sign in** menu in the toolbar.
2. Click **Switch active portal**.
   This opens a drop-down list of all available portals.
3. Click the portal you wish to make active.

Note:
You may need to sign in to the portal you selected.

Get map data (Online/Enterprise)

Get map data from ArcGIS Online and ArcGIS Enterprise to add data to a new or existing CityEngine scene.

Sync feature layers

In CityEngine 2017.1 or later, 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 polygon feature layer

You can create a polygon feature layer in ArcGIS Online or ArcGIS Enterprise directly in CityEngine.

1. Create a new shape layer with the shape creation tools.
2. Right-click the shape layer in **Scene Editor** and select **Create Feature Layer**. to open
   This opens the **Create new Feature Layer** dialog.
3. Fill out 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 can not be changed, such as adding or deleting new attribute fields.
- Creating a new feature layer is not possible with ArcGIS Enterprise version less than 6.1. **Create Feature Layer** is disabled when the active portal is ArcGIS Enterprise with version less 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 **Navigator**.
2. Right-click the polygon feature layer you want to edit.
3. Click **Import**.
   This opens the **Import feature layer** dialog
4. Select which layers you want to import from the feature layer.
5. If you want to import only polygons which 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 then 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, check the **Log** for further details.

Edit 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 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 in order 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 where 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 directly share maps and geographic information across the platform. CityEngine will walk you through the process of preparing, packaging, and sharing the information online.

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 guides you through the sharing process.

Tip:
You can also right-click the file in the **Navigator** and select **Share as...**

You can share to ArcGIS Online or your organization the following types of files:

- Esri Scene Layer Package (slpk)
- CityEngine Web Scenes (3ws)
- Rule Packages (rpk)
- Tile Packages (tpk)
- CityEngine 360 VR Experiences (3vr)

Sharing data on a different portal

By default CityEngine targets the main ArcGIS Online portal at 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

Integrate ArcGIS Urban with CityEngine

Using ArcGIS Urban in ArcGIS CityEngine allows you to do the following:

- Import ArcGIS Urban plans or projects (parcels, zoning and overlays layers, and scenarios).
- Edit the imported layers and scenarios and synchronize the changes with ArcGIS Urban.
- Publish CityEngine models as scene layers to ArcGIS Online and attach them to a scenario in an ArcGIS Urban project.

Import a plan or project

To import a plan or project from ArcGIS Urban to CityEngine, complete the following steps:

1. Set the active portal.
2. Sign in to the portal.
3. Open the Navigator and click the ArcGIS Urban tab.
   A list of available plans and projects displays.
4. Right-click a plan or project and select **Import as Scene**.
   - A CityEngine scene is created.
   - All scenarios, parcels, and zoning and overlays layers are imported into the scene.
   - A section of the basemap, satellite map imagery, and elevation data is imported.

Note:

- The scene is named after the imported project or plan.
- The scene is stored in a CityEngine project named after the city name configured in the ArcGIS Urban settings. If the CityEngine project doesn’t exist, it is created.

Synchronize changes with ArcGIS Urban

After you make changes to the scenarios or shapes in the imported scene, you must synchronize the changes in CityEngine to upload them to ArcGIS Urban. When you synchronize the scene, it is also updated with changes from the ArcGIS Urban version.

To synchronize the changes, click **ArcGIS Urban** > **Synchronize all scenarios** in the main menu.

Note:

- Adding or deleting a new scenario is also synchronized.
- CityEngine doesn't check for locks on a scenario when it synchronizes; changes to locked scenarios are also uploaded.
- Changes to the existing scenario are uploaded.
- If a shape or scenario has been modified in both the scene and ArcGIS Urban, a dialog box appears and you must choose which version you want to keep; the other version is discarded.

Enable sync between CityEngine and ArcGIS Urban

To allow synchronization between CityEngine and ArcGIS Urban, you must configure the project or plan in the ArcGIS Urban settings.

To enable sync, do the following:

1. Open the project or plan you want to sync.
2. On the project or plan detail card, do one of the following:
   - Click the **Settings** button ⚙. Under **Options**, click the **Enable sync** check box.
   - On the project or plan detail card, do the following:
     - Click **Open**.
     - Click the name of the project or plan at the top of the page.
     - Click **Configure**.
     - Under **Options**, click the **Enable sync** check box.

Note:
Only owners of the project or plan can enable sync.

Publish Urban scene layers

You can publish CityEngine models as scene layers and link them as external layers to an ArcGIS Urban plan or project scenarios. The scene layers appear in ArcGIS Urban when you switch scenarios in the **Projects** or **Plans** 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 in the **Scene Editor** or the **Viewport** pane.
   - You can publish generated models and static models.
   - Shapes are not exported.
3. Click **ArcGIS Urban** > **Publish selected models to Urban Scene Layer** in the main menu.
   The models are published to a scene layer and are linked to the scenario.

   Note:
   - If the scene layer is already 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.

Generate plausible buildings

in ArcGIS Online, you can generate plausible buildings of developed parcels similar to that in ArcGIS Urban. When you open CityEngine, the latest ArcGIS Urban rule is automatically applied to your parcels.

Parcels Inspector

When a parcel is selected, the **Inspector** displays the **Parcels** section instead of **Object Attributes**. The selected parcel zoning attributes are shown as they are in ArcGIS Urban.

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 override suffix in the attribute label. To reset an overridden attribute, click the drop-down menu in the combo box and select **Remove override.**

Change building types

To change the building type for the selected parcel, click the **Building type** combo box and select the new type from the list.

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.

ArcGIS 360 VR

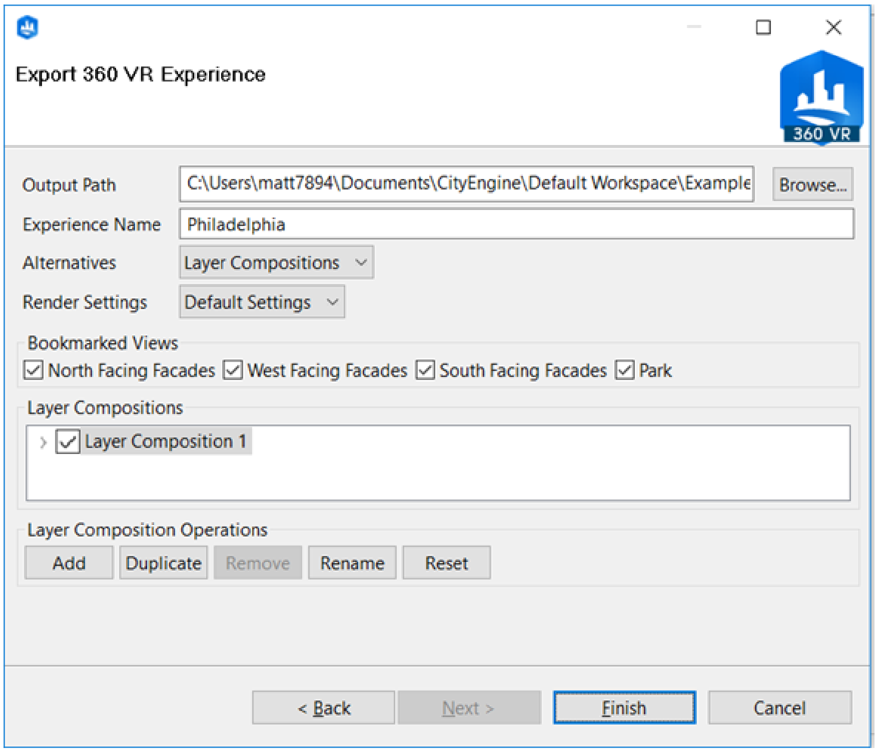
360 VR Experience Export

ArcGIS 360 VR Experience (3VR) is a file format to publish and consume panoramic images of CityEngine 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 3VR exporter is basically 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). Afterwards, the resulting 3VR file is ready to be consumed locally or via the cloud after uploading.

To view CityEngine 3VR experiences on desktop, mobile devices, and VR, see 360 VR Web Application.

Basic 3VR Export

The 3VR export converts CityEngine scene layer compositions or scenarios into 3VR alternatives and writes everything into a single 3VR file. Camera bookmarks are turned into VR viewpoints.



To create a 3VR file, do the following:

1. Open the CityEngine scene to export to VR.
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 3VR export wizard in **File** > **Export** > **CityEngine** > **Export 360 VR Experience**.
5. Adjust the export settings.

Output Path	Specify where to save the 3VR file. If you plan to upload it to to ArcGIS Online, a subdirectory of the current CityEngine project is recommended.

Experience Name	The title of the panoramic image.
Alternatives	• **None**: Offer no alternatives. The 3VR exporter will use the current layer and object visibility and bake a panoramic image for each bookmark. • **Layer Compositions**: Specify the composition of scene layers, i.e. which scene layers to include in an alternative. The default state is initialized with the layer visibility. Use the Layer Composition Operations for modification. • **Scenarios**: Set the scene scenarios as alternatives.
Render Settings	Choose the render settings to use when exporting the panoramas. • **Default Settings**: Choose the default render settings. • **{Viewport}**: Choose a viewport to export its render settings defined in the View settings.
Bookmarked Views	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 resulting 3VR file can now be viewed locally, transferred onto a device, or uploaded to the cloud, such as ArcGIS Enterprise or ArcGIS Online.

Share 3VR to ArcGIS Online or ArcGIS Enterprise

Note:
Although you can share 3VR items to ArcGIS Online and ArcGIS Enterprise, currently, they can only be viewed in ArcGIS Online.

To share a 3VR file, do the following:

1. Export the 3VR to a folder in your current CityEngine project , such as the `\model\` folder.
2. Sign in to ArcGIS Enterprise of ArcGIS Online.
3. In the **Navigator**, right-click on the 3VR file and choose **Share as...**.
4. Choose option **Upload package to my ArcGIS Online or Portal account** and enter a name of the experience.
5. Fill in the required fields for the **Item Description**.
6. Choose the access permission of the Item in the **Sharing** tab.
   360 VR Experience viewers might only work with public items.
7. Click **Analyze** to validate your 3VR Item 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 3VR file.

Preview 3VR files

To preview a 3VR file locally in your browser, do the following:

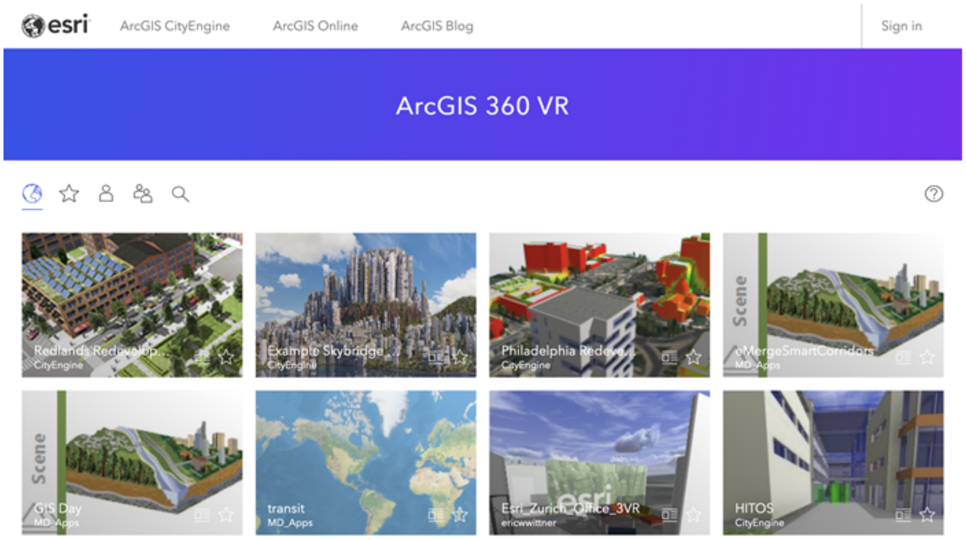
1. Export the 3VR to a folder in your current CityEngine project , such as the `\model\` folder.
2. In **Navigator**, right-click the 3VR file and click **Open in browser**.
   The 3VR item opens in your default browser.

360 VR Web Application

The ArcGIS 360 VR Web Application allows you to view 360 VR Experiences (3VR) on desktop PCs, mobile devices, and virtual reality (VR) headsets. 3VR is a file format exported from CityEngine that allows the consumption of panoramic images of CityEngine scenes. See https://360vr.arcgis.com to view the web application.

3VR gallery

The 3VR gallery displays available 3VR items that you can launch as experiences.



You can filter the items with the following options:

	Featured and popular items
	Your favorite items*
	Your items*
	Items from your organization*
	Search items
* Requires sign in	

Each item thumbnail allows you to do the following:

- Click to go to the 3VR item page.
- Click to add it to your favorites.

Using the web application

Use VR, desktop, or mobile devices to view 3VR. Viewpoints and scenarios allow you to explore the 3VR further. Click the viewpoints to navigate to predefined views. Click the scenarios to see specific alternatives when multiple scenarios are present in the 3VR.

Screen experience on desktop and mobile

On a desktop, click and drag to rotate and look around. On a mobile device, rotate the device around to navigate. Click or tap the viewpoints and scenarios to explore the 3VR.

Use **Full-screen** to enter full-screen mode. To go back to the 3VR gallery, click .

Note:
The full-screen mode is not available on all platforms.

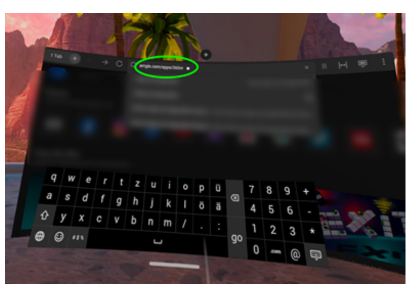
The following are desktop keyboard shortcuts:

- To switch viewpoints, press the `Left` or `Right` arrow key or `A` or `D` keys.
- To switch scenarios, press the `Up` or `Down` arrow key or `W` or `S` keys.

VR experience using a headset

Open the 3VR item in the web browser on your VR device. For example, on Oculus Quest, click the Oculus button on the right controller to open the quick menu. Select **Browser**. Enter the `360vr.arcgis.com` URL in the browser.

Select an item to enter VR mode. On desktop browsers, or when the VR mode is not automatically entered, click to enter VR mode.



Use the VR controllers to switch to a different viewpoint by pointing the controller at the viewpoint thumbnail or camera button , and clicking any controller button. You can also click the scenarios to see different alternatives. Point at the gallery button and click to go back to the 3VR gallery.

The following are controller shortcuts:

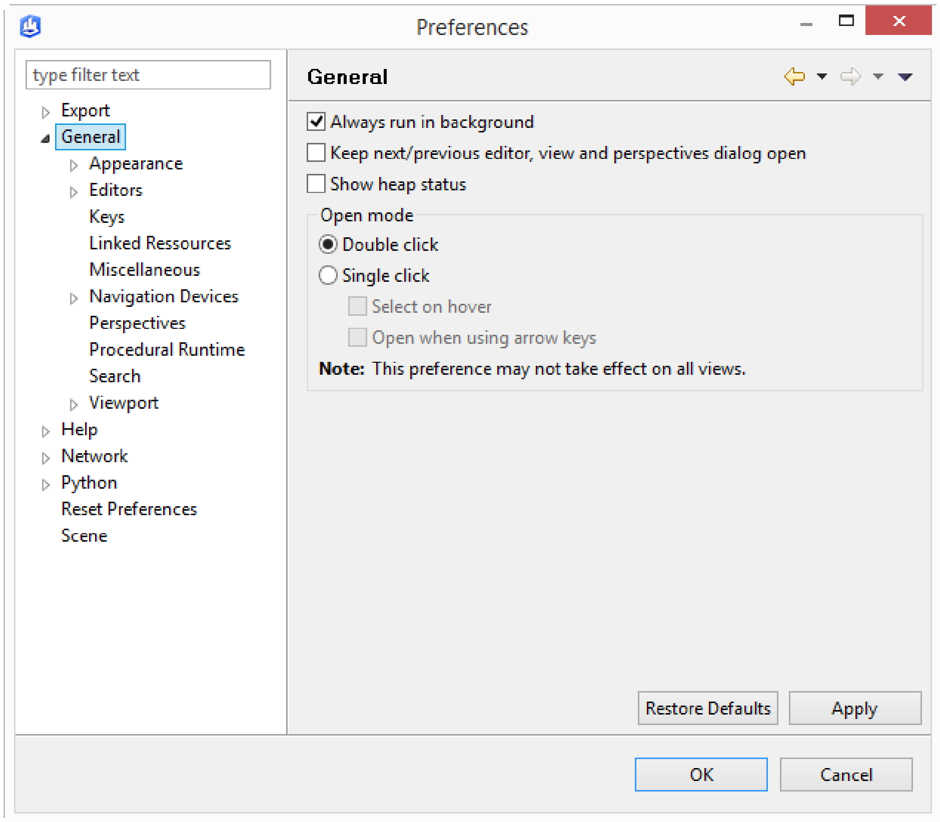
- To quickly change views, tilt the controller joystick left or right, or swipe left or right on a controller with touch pad.
- To quickly change scenarios, use up or down.
- To show or hide the menu, point the controller outside of any menus and click any controller button.

To learn more about how 3VR items are created in CityEngine, see ArcGIS 360 VR Experience.

Note:
The ArcGIS 360 VR Web Application replaces the legacy Oculus app for Oculus Go and Gear VR headsets.

Preferences and Shortcuts

General preferences

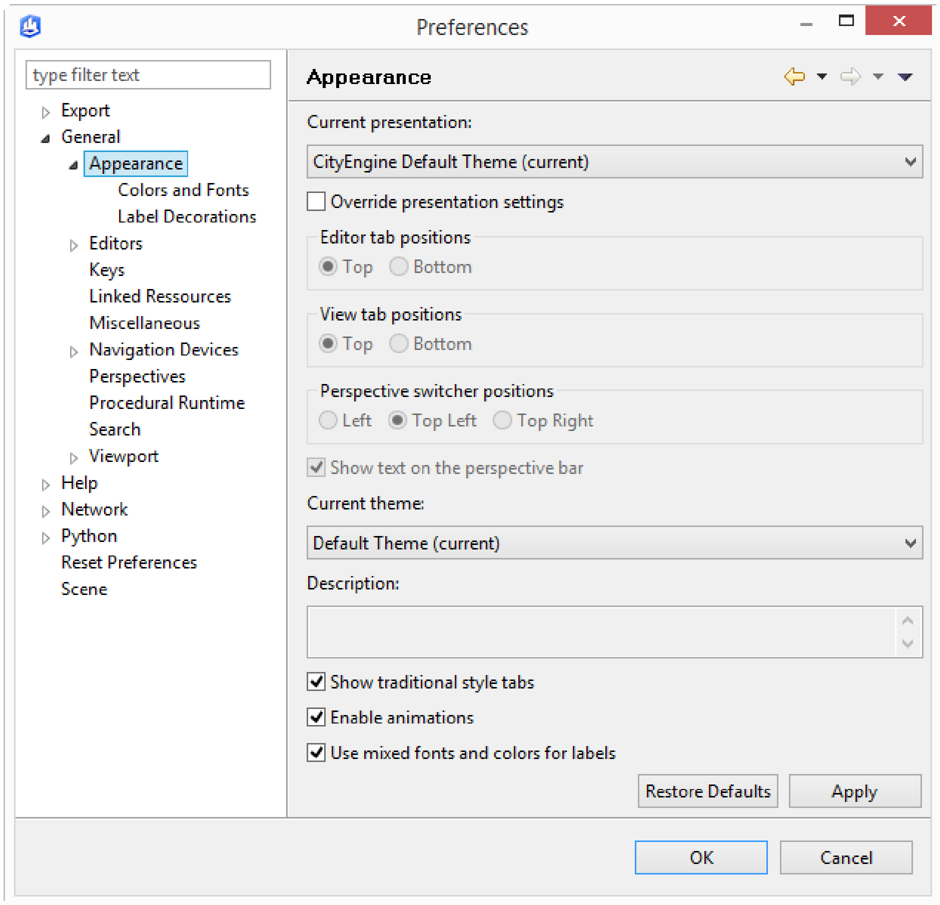


General preferences menu

The following preferences can be changed on the General preference page:

- Always run in background: Turn this option on to perform long running operations in the background without blocking you from doing other work.
- Keep next/previous part dialog open: If this option is turned on then the editor and view cycle dialogs will remain open when their activation key is let go. Normally the dialog closes as soon as the key combination is release.
- Show Heap Status: Turn this option on to display an indicator showing information about current Java heap usage. A basic heap monitor is always enabled inside CityEngine and reports Java and core heap usage.
- Open mode: You can select one of the following methods for opening resources:
- Double click - Single clicking on a resource will select it and double clicking on it will open it in an editor.
- Single click (Select on hover) - Hovering the mouse cursor over the resource will select it and clicking on 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.

Appearance preferences

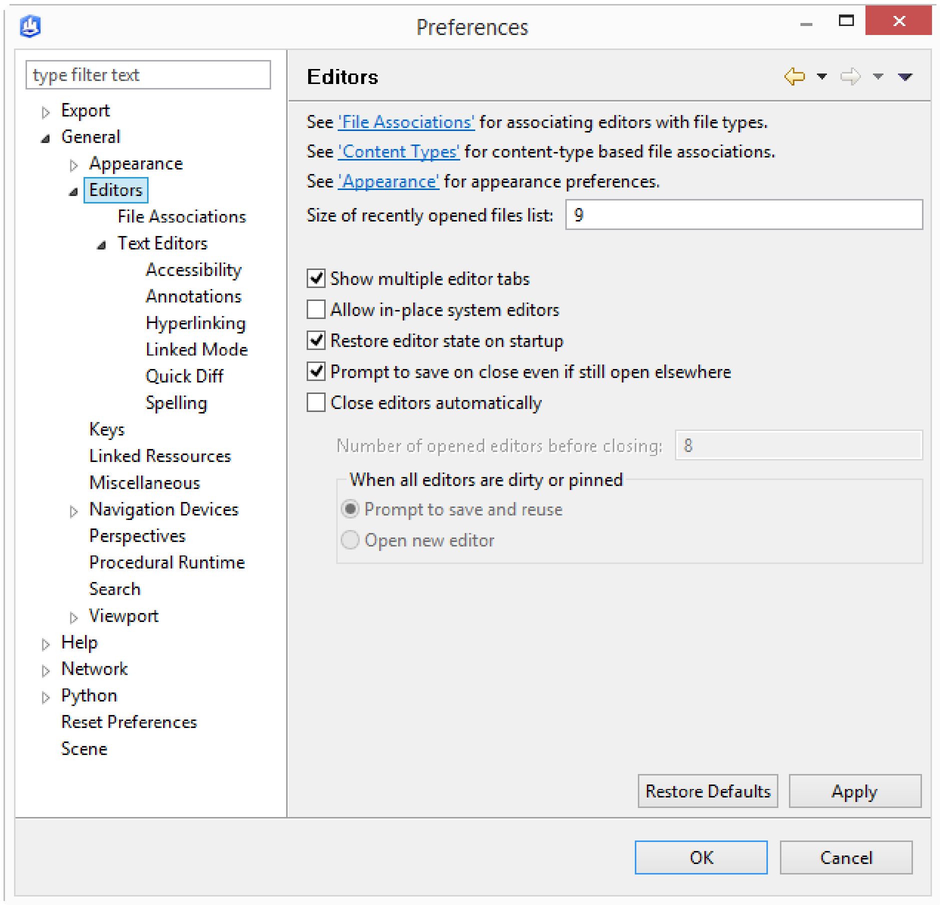


Appearance preferences menu

You can change the following preferences on the Appearance preferences page:

- Current presentation: Specify the currently active presentation (look and feel).
- Override presentation settings: Locally override the settings for the current presentation.
- Editor tab positions: Specify either top or bottom to indicate where you want tabs for stacked editors to appear.
- View tab positions: Specify either top or bottom to indicate where you want tabs for stacked views to appear.
- Perspective switcher positions: Specify the location of the perspective switcher bar (unused in the current CityEngine release).
- Current theme: specifies the currently active theme (color and font set).
- Show traditional style tabs: Specify whether traditional (square) tabs should be used in place of the curved tabs.
- Enable animations: Enable/disable the feature where views animate to their location when closed or opened.

Editors preferences

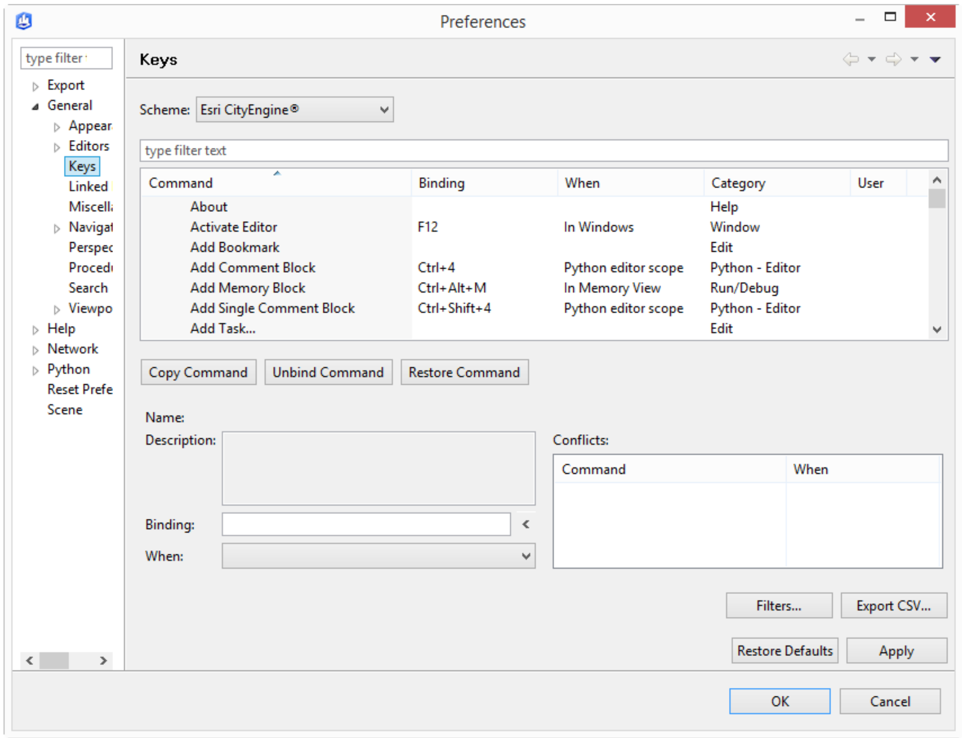


Editors preferences menu

You can change the following preferences on the Editor's preference 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: Specifies whether you wish to show multiple editor tabs. If off, editor workbooks have one large tab and all non-visible editors are accessible only from the chevron.
- Show Heap Status: Turn this option on to display an indicator showing information about current Java heap usage. A basic heap monitor is always enabled inside CityEngine and reports Java and core heap usage.
- Allow in place system editors: Specifies if OLE in-place editing will be used on the Windows platform.
- Restore editor state on startup: Specifies if editors should be restored on the next launch of CityEngine.
- Close editors automatically: Specifies whether or not to re-use editors in CityEngine. If 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 should be opened or if a new editor should be opened 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



Keys preferences menu

The function of the keyboard can be extensively customized CityEngine using the keys preference page. Within CityEngine, key strokes and key sequences are assigned to invoke particular commands.

Key Strokes, sequences, and bindings

A 'key stroke' is the pressing of a key on the keyboard, while optionally holding down one or more of these modifier keys: `CTRL`, `ALT`, or `SHIFT`. For example, holding down `CTRL` then pressing A produces the key stroke `CTRL+A`. The pressing of the modifier keys themselves do not constitute key strokes.

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`. To enter this key sequence, one presses the key stroke `CTRL+X` followed by the key stroke `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 preference page.

Contexts

Key bindings can vary based on the current context of CityEngine.

Sometimes the active part might be a CGA shape grammar editor, for instance, where 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`, or `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 as well. If the active window does not choose a particular context, CityEngine will set the active context to `In Windows`.

CityEngine includes a number of different contexts. Some examples are:

- 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 which it extends. For example, it is not recommended to move an Editing Text key binding to the In Dialogs and Windows context. This may have unexpected results.

It is possible for some key bindings to work in dialogs. 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". It is possible to change these key bindings. For example, it is possible to have `CTRL+X` as cut in dialogs, but `CTRL+W` as cut in windows.

Platform and locale

Key bindings also vary by platform and locale. On Chinese locales (zh), `ALT+/` is assigned to `Content Assist`, instead of the usual `CTRL+SPACE`.

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. To make things easier, all key customization is done on the keys preference page.

In this example we want to bind `CTRL+5` to the About command. By default, the keys preference page will show 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` like you would if you were executing the command.

When you type `CTRL+5` you have created a binding for About. The right-most column 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 combo box to change the key binding context (for example, to move this binding to "Editing Text").

If you wanted 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. If you want to delete a binding, you can either use the `Remove Binding` button or simply give focus to the Binding text box and hit `Backspace`.

Conflict resolution

There are only a finite number of simple, common key strokes available to assign to a multitude of commands. We have seen that 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 keybinding and creates a conflict, the conflicting bindings will be displayed in the conflicts list. This can be used to navigate between conflicting key bindings so that they can be changed.

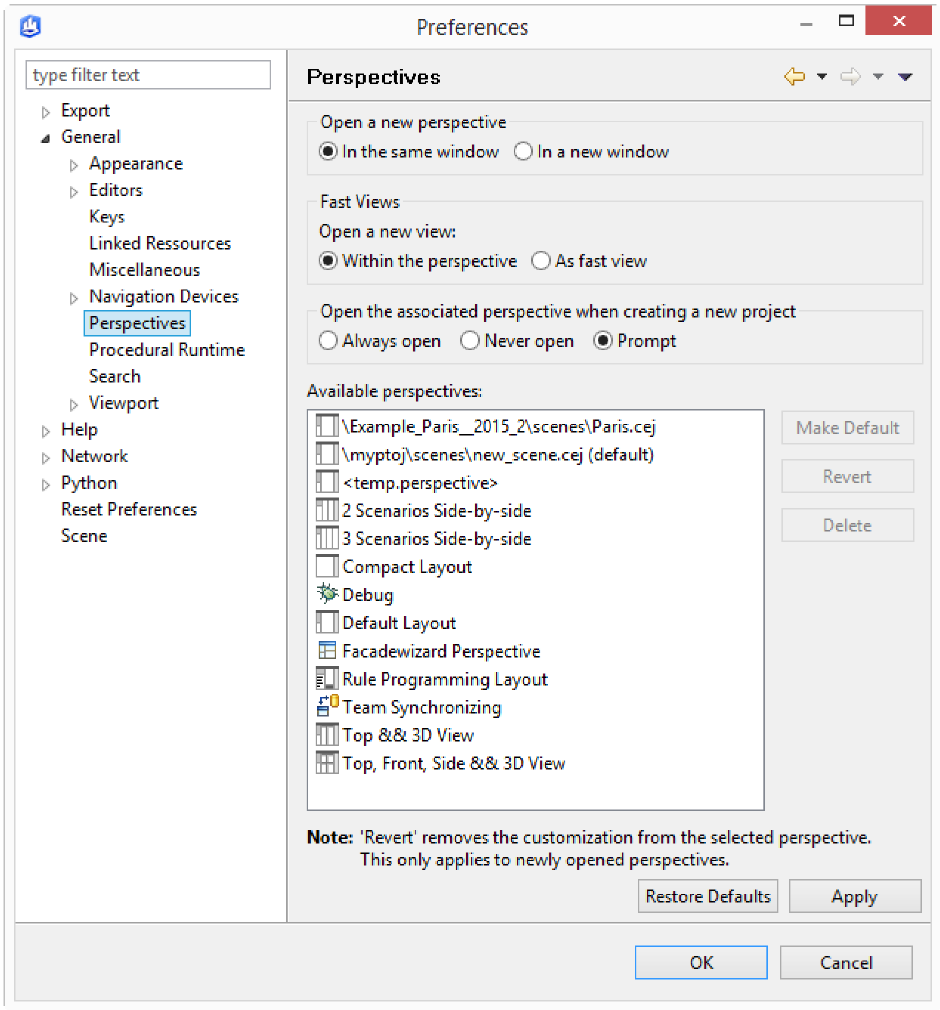
These types of conflicts can be resolved by explicitly assigning the key sequence to one of the commands, or remove it from the other.

Miscellaneous preferences

The following preferences can be changed on the Miscellaneous preference page:

- Exit CityEngine after [minutes]: CityEngine features an auto-exit functionality. CityEngine can automatically exit after a given number of minutes of inactivity. Before exiting, all open files will be saved. Enter '0' minutes in order to disable automatic exit. Automatic exit can be useful for installations where people usually do not close applications and thus do not return licenses held by these application to the license server. Closing the application returns the licenses held to the license server as a side-effect and allow in turn other users to start the application.
- Double click behaviour: Can be set to None, Inspect Model or Frame Model.
- Enable immediate mode: In immediate mode, procedural models are generated automatically if the underlying initial shape or its attributes change.

Perspectives preferences



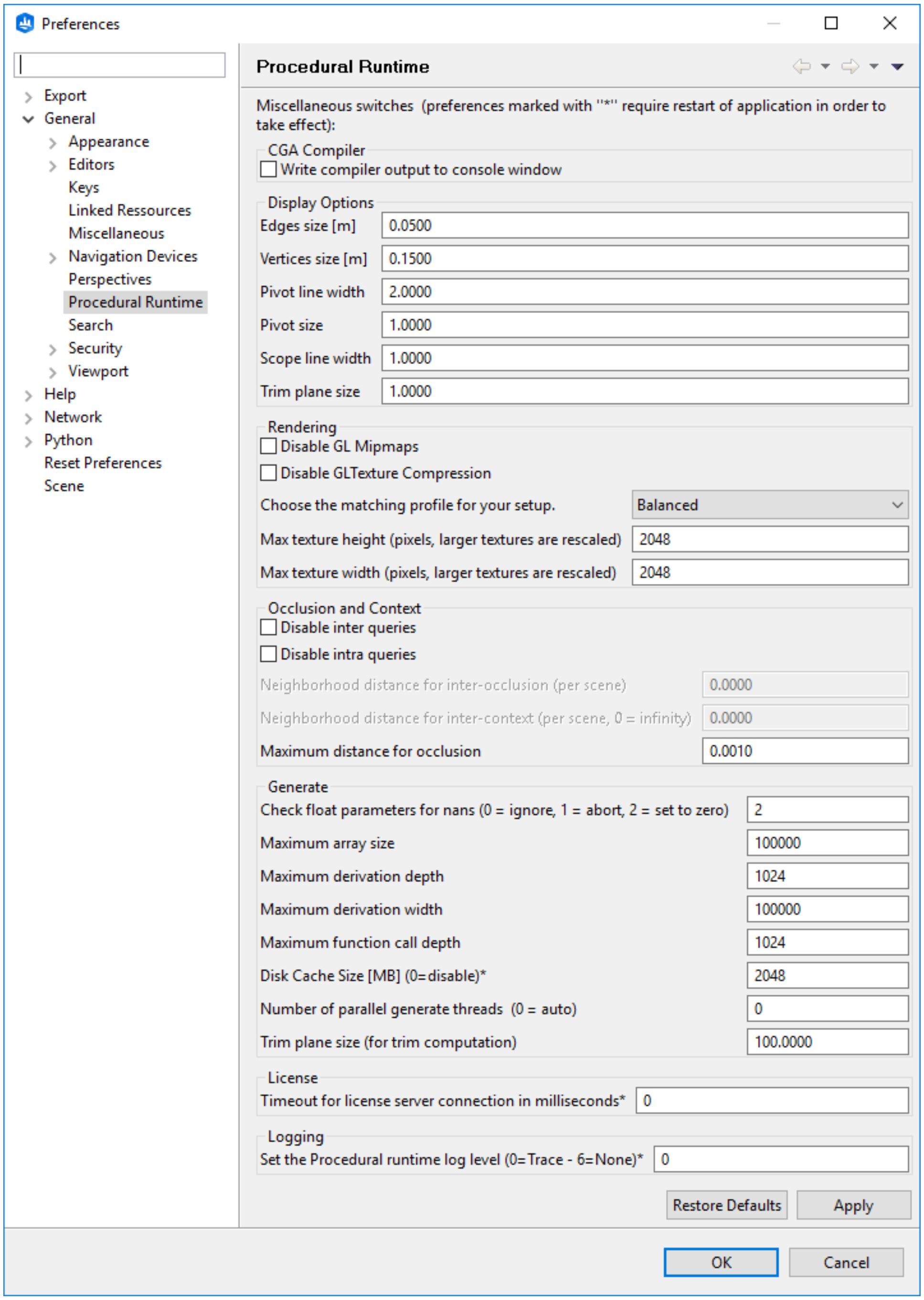
Perspectives preferences menu

On the Perspectives preference page, you can manage the various perspectives defined in CityEngine.

- Open a new perspective: Use this option to set what happens when you open a new perspective. Do you want the perspective opened within the current Workbench window or opened in a new window?
- Open a new view: Use this option to specify what happens when a new view is opened. It is either opened to its default position within the current perspective or it is opened as a fast view and docked to the side of the current perspective.
- New project options: Use this option to specify the perspective behavior when a new project is created. You can set it to switch the current perspective to be the one associated with the project type and open the perspective in the same Workbench window as the current one, switch the perspective and open it in a new Workbench window, or not to switch perspectives at all.
- Make Default: Sets the selected perspective as the default perspective.
- Reset: Resets the definition of the selected perspective to the default configuration. This option is only applicable to built-in perspectives that have been overwritten using `Window→Layout→Save Perspective As...`.

- Delete: Deletes the selected perspective. This option is only applicable to user-defined perspectives (built-in perspectives can not be deleted).

Procedural runtime preferences



Procedural runtime preferences

The Procedural Runtime preferences page controls various options with respect to rule derivation (model generation), display, rendering, occlusion and miscellaneous engine arguments.

CGA Compiler

- If "Write compiler output to console window" is enabled, the CGA compiler's activities are logged to a console.

Generate

- 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 hirarchy), 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 hom much disk space is used to cache decoded textures between starts of the city engine. Using the disk cache might reduce memory consumption during runtime because textures do not need to be decoded (and kept in memory) to find their metadata (scuh as aspect ratio etc.).
- Number of parallel generate threads: Use this 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

- Disable inter/intra queries: Disabling intra-shape tree or inter-shape tree (neighbors) queries might 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 need to 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 need to be derived for inter-context queries. If set to 0, the distance is ignored, i.e. 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

- 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 (affects only generated models)

- 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 here..

- 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 which are wider/higher than this value are rescaled to this value in order to save memory.
- Matching profile: Using this popup, you can control render performance versus memory consumption. For most use cases, "Balanced" is a good choice.
- Force OpenGL Double Buffering: On Windows (starting with Vista), double buffering is disabled because the operating system already takes care of smooth rendering. Use this option to force double buffering.

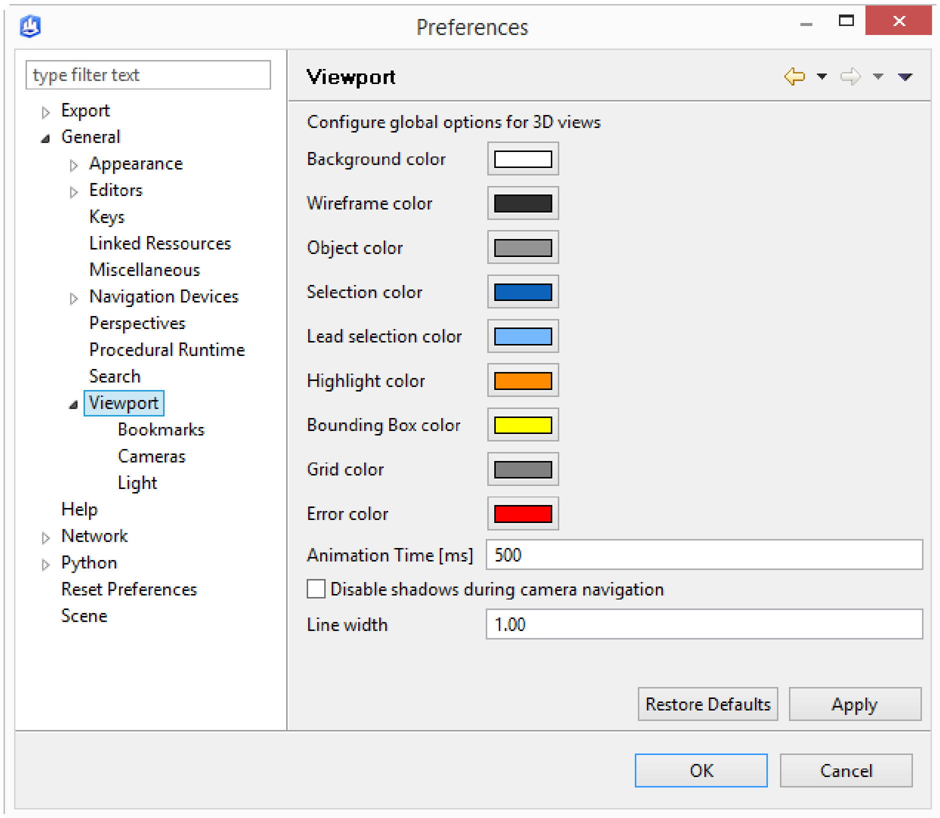
License

- Timeout for license server connection in milliseconds.

Logging

- Set the Procedural runtime log level: If set to a level less than 6, Procedural Runtime logs errors, warnings etc. to a console.

Viewport preferences

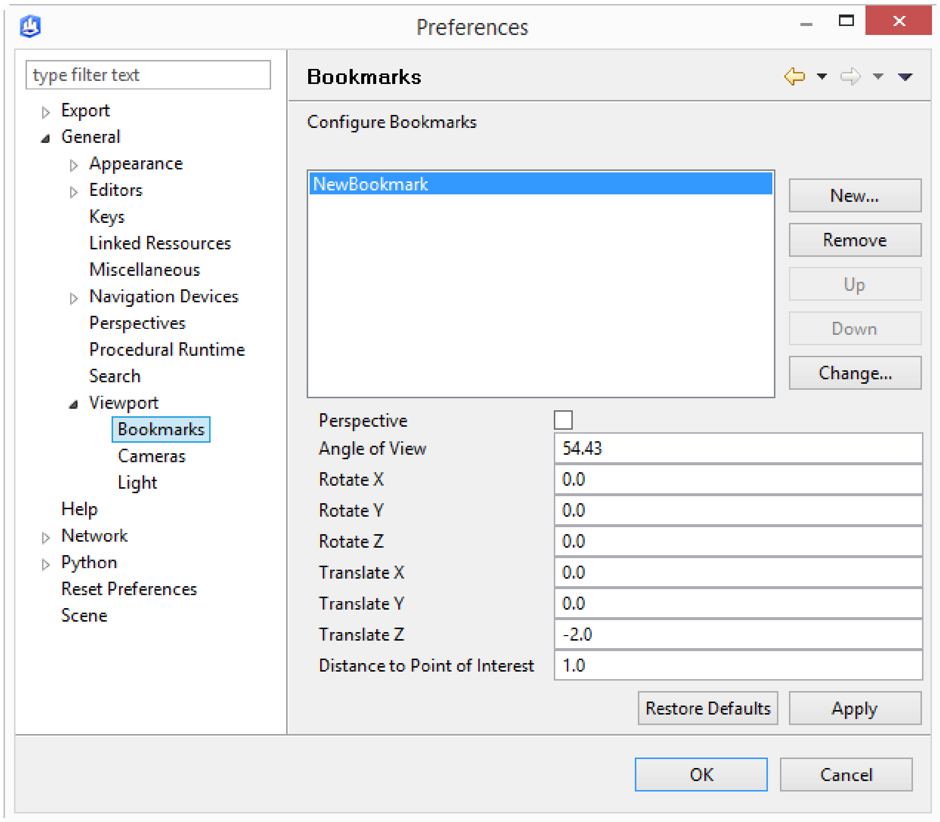


Viewport preferences menu

Change appearance and behavior of 3D viewports.

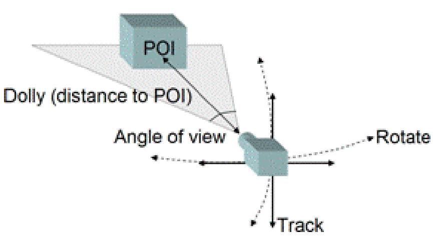
Colors	Set various colors for scene elements
Animation time	Set time for camera animations (e.g. when triggering a camera bookmark)
Checkbox	Disable camera during camera navigation. Speed up camera navigation on heavier scenes.
Line width	Set width for line rendering

Bookmarks preferences



Bookmarks preferences menu

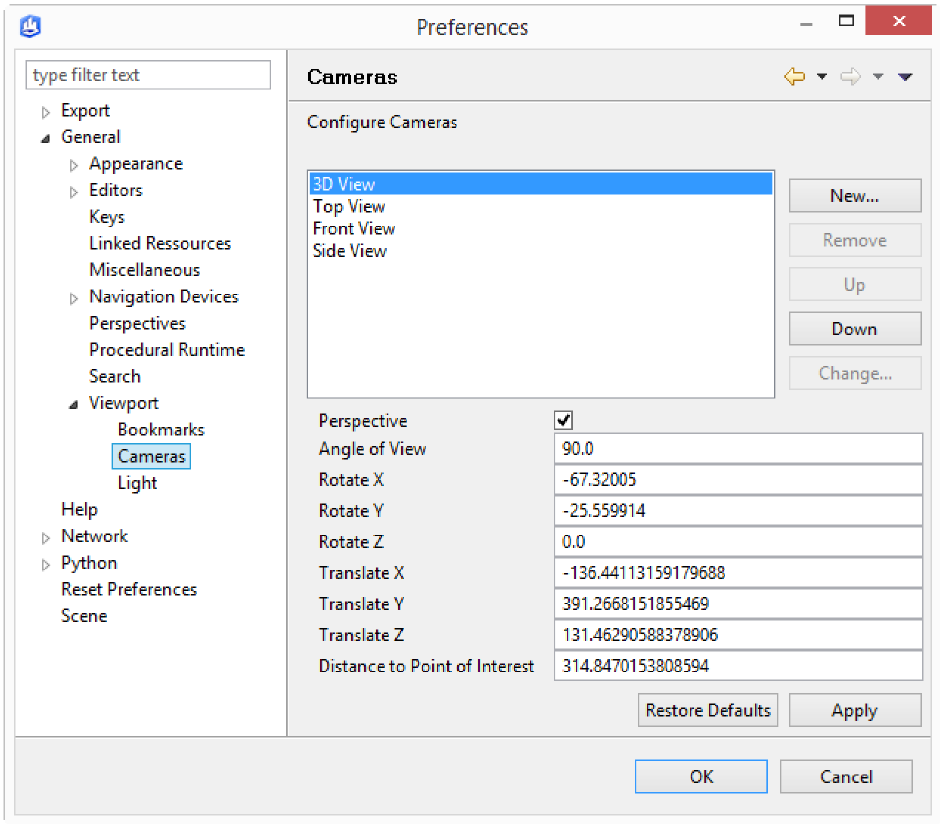
A bookmark has the following options:



Bookmark schematic

- **Perspective** — If selected, the camera will give a perspectively 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. For your convenience, 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 enter 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 POI. The POI is also the center of rotation and thus the distance to the POI defines the radius of the camera rotation.

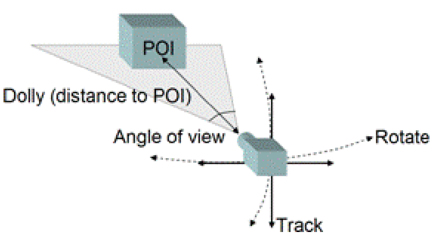
Cameras preferences



Cameras preferences menu

Configure cameras

Camera configurations have the following options:

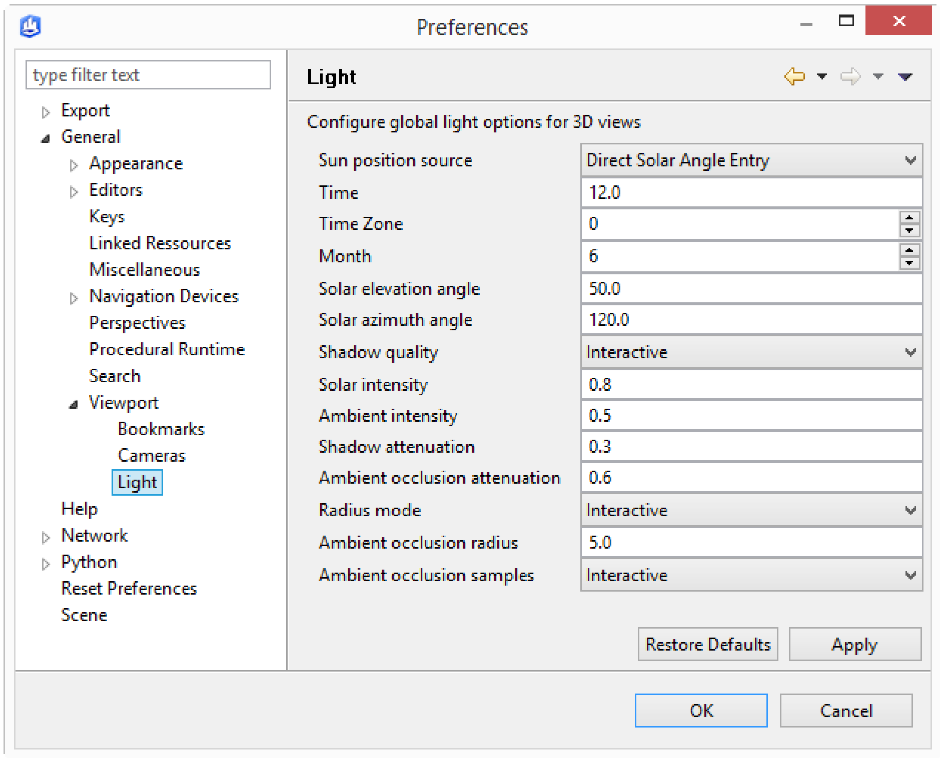


Camera schematic

- **Perspective** — If selected, the camera gives a perspectively 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.

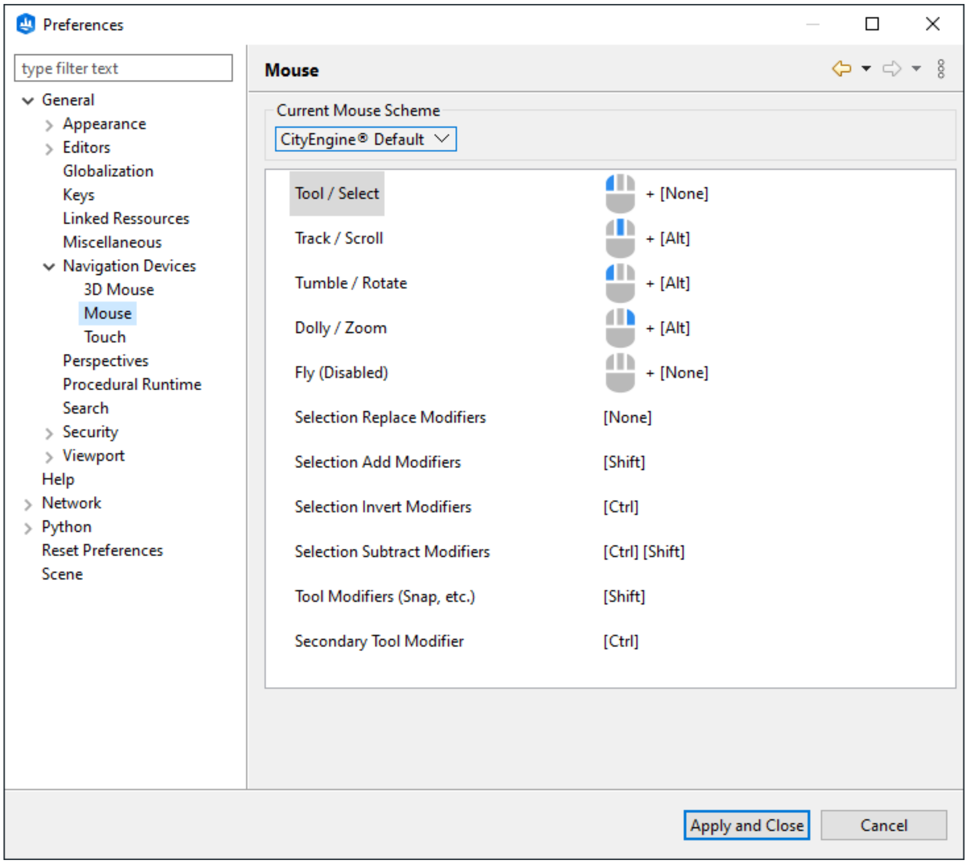
Light preferences

Below are the configure options for light source for the 3D **Viewport**.



Light preferences menu

Mouse preferences

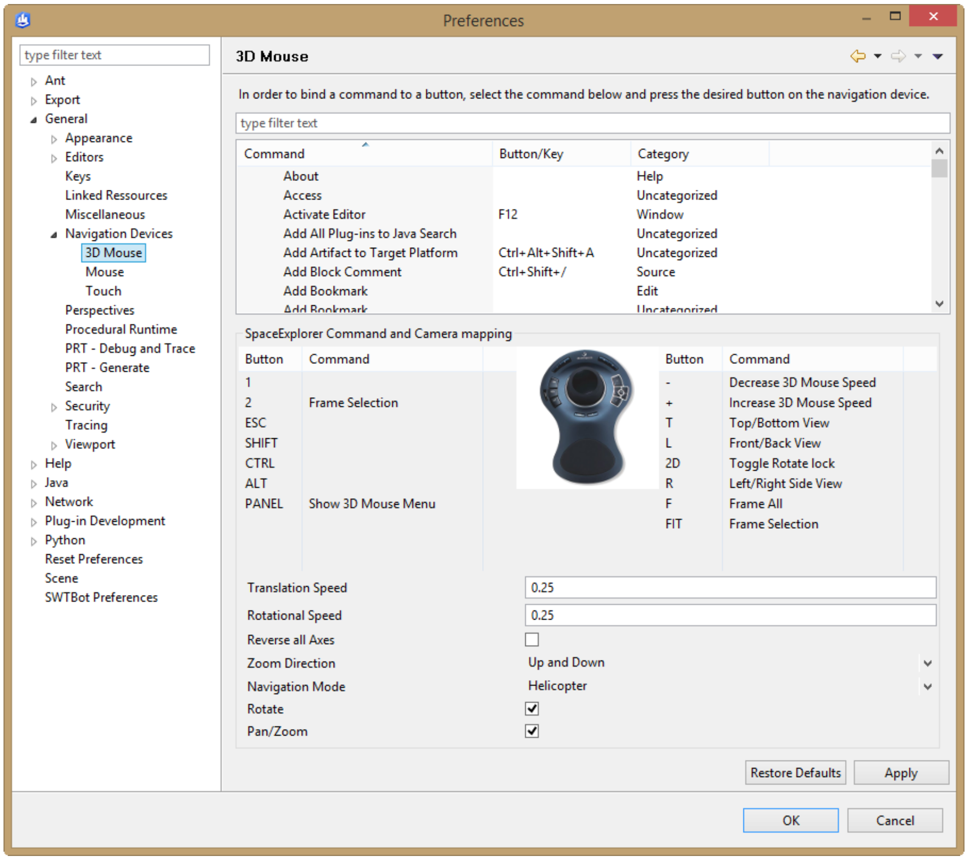


Mouse preferences menu

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 according to your needs.

Note:
On Linux platforms, it is recommended to 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.

3D mouse preferences



3D mouse preferences menu

3D Mouse support is not available in all CityEngine versions.

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 CityEngine's builtin 3D Mouse preferences.

The 3D Mouse preference page lets you configure CityEngine related settings of your Logitech/3DConnexion 3D Mouse. The top of the preference page presents you all 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, just select the command from the list and press the button on the device the command should be assigned to. In order to simplify command selection, the list of commands may be filtered by typing the desired filter into the text field above the list of commands.

You can always restore the default command bindings for your specific device by clicking on the "Restore Defaults" button.

Settings

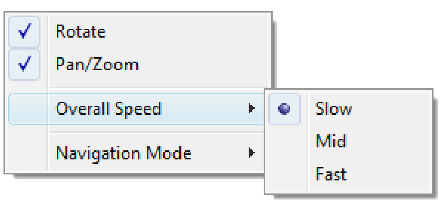
- **Overal Speed:** The overall speed (or sensitivity) of the 3D Mouse. Change this if you think your device responds to quick or slow.
- **Reverse all Axes:** Some people prefer to have the movements in helicopter or camera mode aligned with the mouse. You may reverse the axes with this option.

- **Zoom direction:** By default, zoom is mapped to moving the cap forwards and backwards. Some people prefer zoom on the up/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.

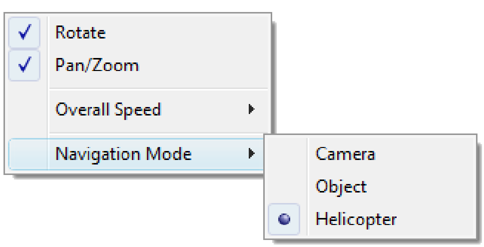
Navigation modes

- Camera mode navigation is characterized by the user having the impression that they are moving around in the scene they are observing. This moves the user around and 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 main characteristic of object mode navigation is that the user has the impression they are holding the object in his hand. The center of rotation is the current point of interest, e.g. set to the center of the selection by framing.
- As the name suggests, this 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



Overall speed settings

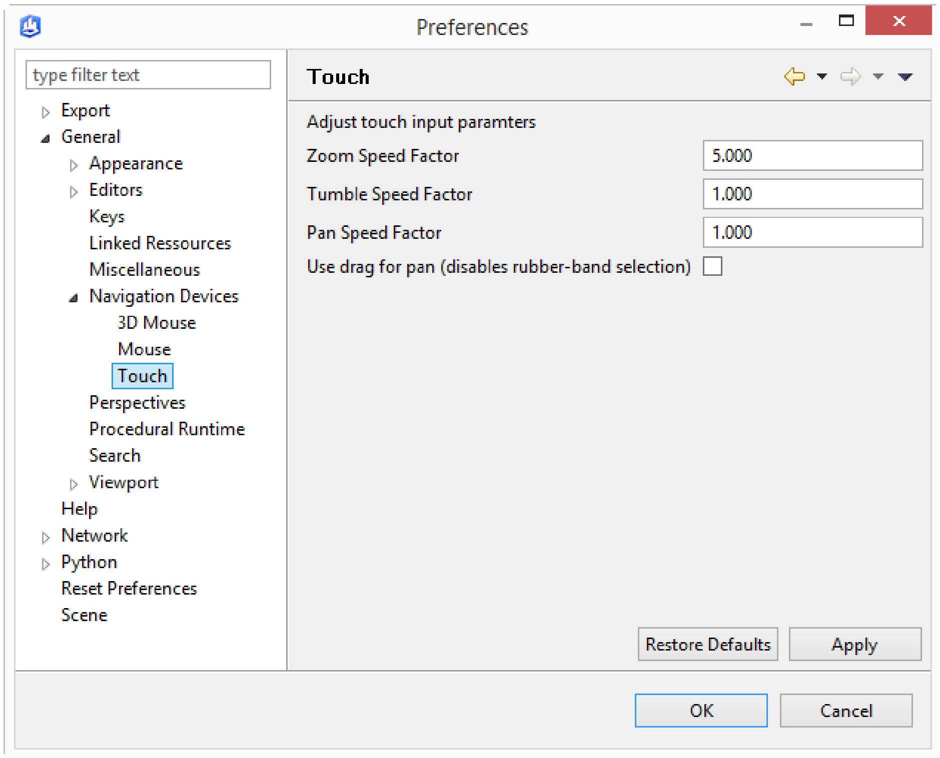


Navigation mode settings

By default, button "1" on the 3D Mouse is assigned to the 3D Mouse menu. This menu allows you to lock translation and/or rotational movements as well as change the speed and navigation mode quickly.

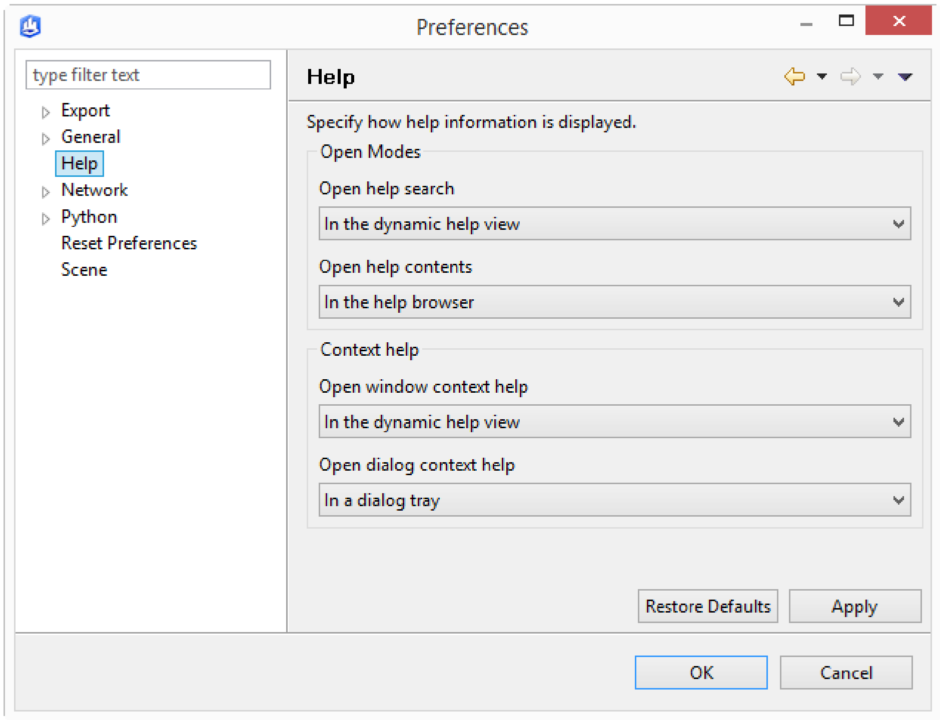
Touch preferences

The touch preferences allow you to fine-tune the touch navigation in CityEngine.



Touch preferences menu

Help preferences



Help preferences

On the Help preferences page, you can indicate how to display help information.

- Use external browsers: If embedded web browser is supported on your system, 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 "Web Browser" preference page to select 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 an infopop.
- 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 an infopop.
- 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.

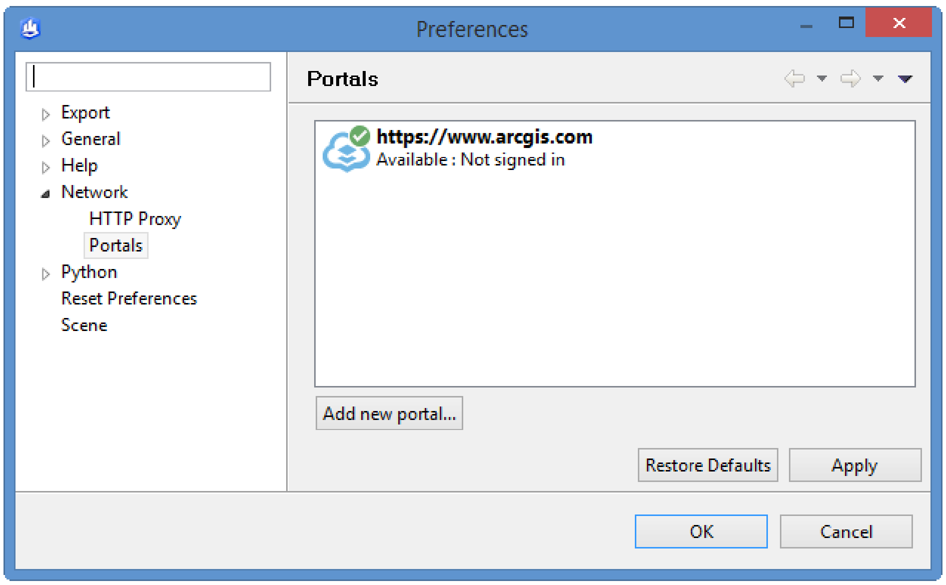
Network preferences

Change network settings for Portal URL and Proxy.

ArcGIS Online or Portal

By default CityEngine connects to ArcGIS Online at www.arcgis.com. Set an URL to change the target portal for sign in and for sharing CityEngine Web Scenes and Rule Packages.

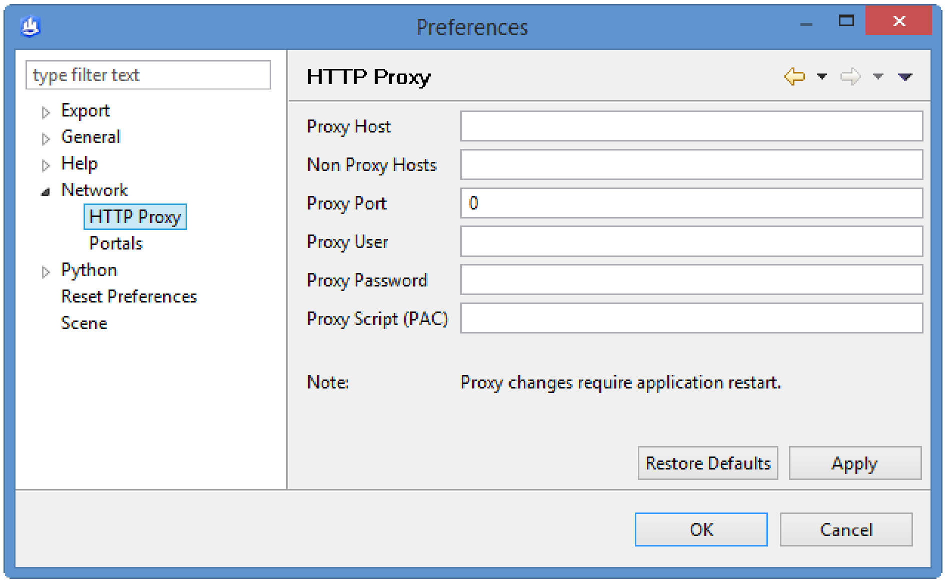
- Portal for ArcGIS on-premise installation:
  `https://webadaptor.domain.com/arcgis`
- Setting the URL to an ArcGIS Online organization:
  `http://jsapi.maps.arcgis.com`



Portals preferences

HTTP Proxy

Define proxy settings for CityEngine network connections.



HTTP proxy preferences

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 your CityEngine scene. It only redefines the internally stored reference system used for future data imports.

Show handles

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 might be globally assignedto other actions. In this case, you have the choice to either change the Operating System's defaults (where available) orto change CityEngine's key bindings (see Keyboard Preferences)

You can always recall a list of currently active CityEngine shortcuts by pressing CTRL+SHIFT+L.

Command	Binding	Category	When
Activate Editor	F12	In Windows	Window
Add Comment Block	Ctrl+4	Python editor scope	Python - Editor
Add Memory Block	Ctrl+Alt+M	In Memory View	Run/Debug
Add Single Comment Block	Ctrl+Shift+4	Python editor scope	Python - Editor
Backward History	Alt+Left	In Windows	Navigate
Bookmark 0	Numpad_0	In 3D Viewport	Viewport Control
Bookmark 1	Numpad_1	In 3D Viewport	Viewport Control
Bookmark 2	Numpad_2	In 3D Viewport	Viewport Control
Bookmark 3	Numpad_3	In 3D Viewport	Viewport Control
Bookmark 4	Numpad_4	In 3D Viewport	Viewport Control
Bookmark 5	Numpad_5	In 3D Viewport	Viewport Control
Bookmark 6	Numpad_6	In 3D Viewport	Viewport Control
Bookmark 7	Numpad_7	In 3D Viewport	Viewport Control
Bookmark	8 Numpad_8	In 3D Viewport	Viewport Control
Bookmark 9	Numpad_9	In 3D Viewport	Viewport Control
Bounding Boxes	B	In 3D Viewport	Viewport Control
Cancel	Esc	In Dialogs and Windows	Engine Control
Close	Ctrl+W	In Windows	File
Close	Ctrl+F4	In Windows	File
Close All	Ctrl+Shift+F4	In Windows	File
Close All	Ctrl+Shift+W	In Windows	File
Close Rendering	Ctrl+W	In Memory View	Run/Debug
Collapse	Ctrl+Numpad_Subtract	Editing Text	Text Editing
Collapse	-	In 3D Viewport	Graph Commands
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

Command	Binding	Category	When
Content Assist	Ctrl+Space	In Dialogs and Windows	Edit
Context Information	Ctrl+Shift+Space	In Dialogs and Windows	Edit
Copy	Ctrl+C	In Dialogs and Windows	Edit
Copy	Ctrl+Insert	In Dialogs and Windows	Edit
Copy Lines	Ctrl+Alt+Down	Editing Text	Text Editing
Cut	Shift+Delete	In Dialogs and Windows	Edit
Cut	Ctrl+X	In Dialogs 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	+	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
Extract Local...	Alt+Shift+T, L	Python editor scope	Python - Refactor
Extract Method...	Alt+Shift+T, E	Python editor scope	pepticcategory
Extract Method...	Alt+Shift+M	Python editor scope	pepticcategory
Find and Replace	Ctrl+F	In Windows	Edit
Find Next	Ctrl+K	Editing Text	Edit

Command	Binding	Category	When
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/Back View	Z	In 3D Viewport	Viewport Control
Generate Constructor using Fields...	Alt+Shift+T, C	Python editor scope	pepticcategory
Generate Models	Ctrl+G	In Dialogs and Windows	Shape Commands
Generate Properties...	Alt+Shift+T, P	Python editor scope	pepticcategory
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
Join Lines	Ctrl+Alt+J	Editing Text	Text Editing
Last Edit Location	Ctrl+Q	In Windows	Navigate
Left/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	.	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

Command	Binding	Category	When
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+.	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+.	In Table Memory Rendering	Run/Debug
Next Sub-Tab	Alt+PageDown	In Dialogs 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	Ctrl+H	In Windows	Search
Override/Implement Methods...	Alt+Shift+T, O	Python editor scope	pepticcategory
Paste	Ctrl+V	In Dialogs and Windows	Edit
Paste	Shift+Insert	In Dialogs and Windows	Edit
Previous	Ctrl+,	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+,	In Table Memory Rendering	Run/Debug
Previous Sub-Tab	Alt+PageUp	In Dialogs 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+-	Python editor scope	Python - Editor

Command	Binding	Category	When
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+/	Python editor scope	Python - Editor
Python Un Collapse	Ctrl+=	Python editor scope	Python - Editor
Python Un Collapse All	Ctrl+0	Python editor scope	Python - Editor
Python Uncomment	Ctrl+Shift+3	Python editor scope	Python - Editor
Python Uncomment	Ctrl+\	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 Dialogs and Windows	Edit
Refresh	F5	In Windows	File
Regenerate all Models	Ctrl+F5	In Dialogs 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

Command	Binding	Category	When
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
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 Dialogs 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

Command	Binding	Category	When
Set Tool (Tool: Local Edits)	O	In 3D Viewport	Tool Commands
Set Tool (Tool: Measure distance)	M,D	In 3D Viewport	Tool Commands
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 Dialogs 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 Dialogs 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 Dialogs and Windows	Window
Show/Hide Graph Networks	F10	In 3D Viewport	Uncategorized
Show/Hide Map Layers	F9	In 3D Viewport	Uncategorized
Show/Hide Models	F12	In 3D Viewport	Uncategorized
Show/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 Lower Case	Ctrl+Shift+Y	Editing Text	Text Editing
To Upper Case	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	,	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
Toggle Scene Light	L	In 3D Viewport	Viewport Control
Toggle Wireframe on Shaded/ Textured	7	In 3D Viewport	Viewport Control

Command	Binding	Category	When
Top/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+-	In Windows	Unavailable Category
Undefined Command	Ctrl+1	In Dialogs and Windows	Unavailable Category
Undefined Command	Ctrl+Shift+F8	In Windows	Unavailable Category
Undo	Ctrl+Z	In Dialogs and Windows	Edit
Update Seed	Ctrl+Shift+G	In Dialogs 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+/	Editing	Text Edit
Zoom / Dolly In	Ctrl+=	In 3D Viewport	Navigation Commands
Zoom / Dolly In	Ctrl+Numpad_Add	In 3D Viewport	Navigation Commands
Zoom / Dolly Out	Ctrl+-	In 3D Viewport	Navigation Commands
Zoom / Dolly Out	Ctrl+Numpad_Subtract	In 3D Viewport	Navigation Commands