# Tutorial 2: Terrain and Dynamic City Layouts

# Table of Contents

# Tutorial 2: Terrain and Dynamic City Layouts

### In this tutorial:



### Download:

- Tutorial Data
- Tutorial PDF

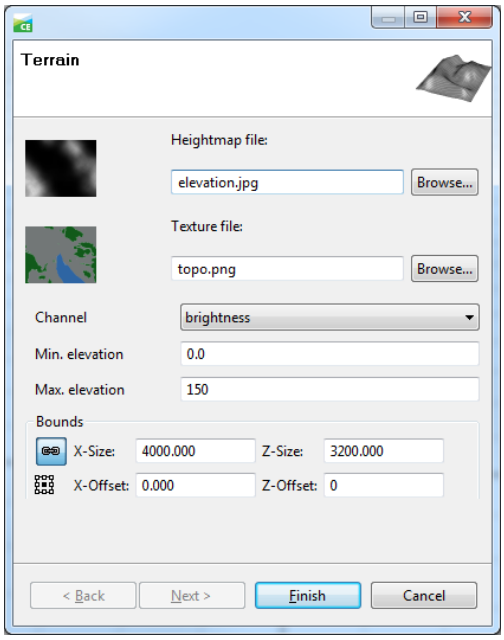
## Part 1: Terrain creation

Usually, the starting point of a CityEngine scene is the creation of a terrain. Terrains can be created from simple image files or also from DEMs (Digital Elevation Model). In the latter case, e.g. with a GEOTif file, the georeferencing information is supported.

Currently, CityEngine only supports image-based terrains (greyscale heightmaps), no 3d meshes. In case you only have a 3d mesh or any other data type, you need to convert it to a heightmap, using a tool such as 'Leveller' by Daylong Graphics (http://www.daylongraphics.com/).

Steps:

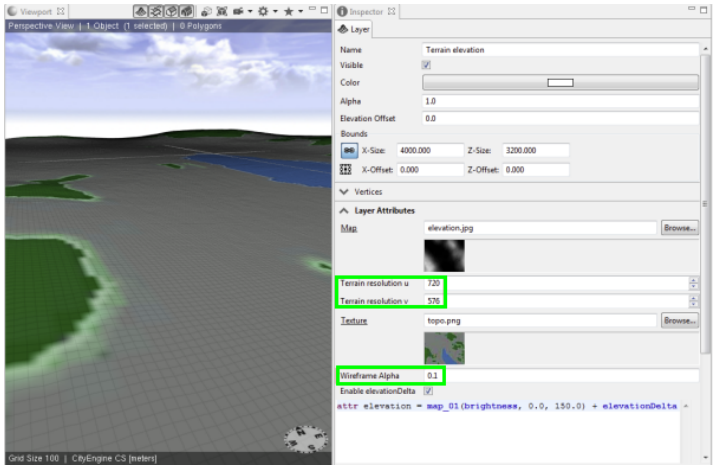
1. Open the file **Terrain_Creation_Start.cej**.
2. Open the **maps** folder, drag and drop the elevation.jpg image into the viewport. This triggers the **Terrain Import** dialog. Choose the **Raw data (no projection)** coordinate system option.
3. Change the **Texture file** to the file **topo.png** using the **Browse** button.
4. Edit the Bounds so the dialog looks as in the following picture. (Note that if you load an image without geolocation information, its pixel resolution is set as the terrain dimensions.)
5. Note the **Keep ratio** and **Alignment** (default = centered, edit it by clicking on the icon) buttons.
6. Edit the min and max elevation values (= black and white pixel value elevations).
7. Then click **Finish**




.

Once the terrain map layer is created, you can select it in the Scene Editor and see it's attributes in the Inspector window. Note that you can edit the **Terrain Resolution** (the number of subdivisions of the terrain plane) and the **Wireframe Alpha** value. The latter shows the wireframe lines on the terrain, if, in the viewport's display options, the option **Wireframe on Shaded/Textured** is activated (toggle with **7**).

8. Play with the terrain's attributes!



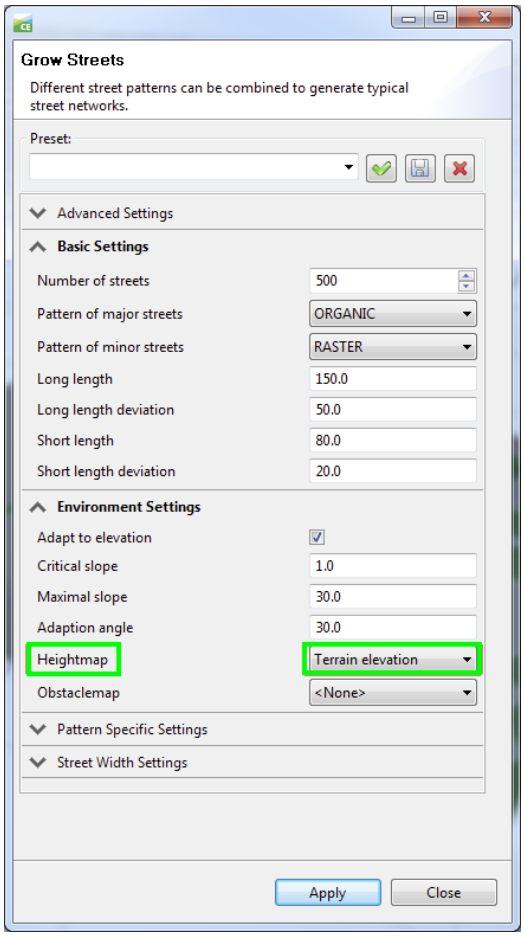
## Part 2: Growing streets: Heightmap and Obstaclemap

### Grow Streets: Alignment to the terrain

In Tutorial 2, we have learned how to grow streets on the flat grid, where there's no elevation present. We will now see that the **Grow Streets** algorithm can also directly adapt the grown streets to the actual terrain elevation.

Steps:

1. Open the **Grow Streets** dialog and open its **Environment Settings**.
2. Change the **Heightmap** dropdown to your terrain's name:




3. Now grow some streets and see how they're automatically aligned to the terrain. Nice.

## Grow Streets: Obstaclemap

You may have noticed that the **Grow Streets** functionality just grows streets randomly. This includes areas where you may want to have no streets at all. To be able to leave those areas empty and minimizing the manual cleanup process, CityEngine lets the user use so-called **Obstacle Maps**. An Obstacle Map is a black and white image, where the brightness defines the **growable** regions. Let's create such an Obstacle Map!

Steps:

1. Delete your previously generated streets again.
2. Click **Layer** > **New Map Layer** > **Obstacle**
3. Browse to the **obstacles.png** image file provided in the Tutorial's **maps** folder.
4. Define the same extent as your terrain's extent.
5. Leave the Channel on **brightness**.
6. The **Obstacle Threshold** defines the image brightness, which distinguishes between **obstacle** and **non obstacle**.

Once the **Obstacle** Layer is created, select it and check it's **Layer Attributes** tab in the Inspector. You will find the following code, which drives the attribute mapping, in this case as a Boolean value (true / false).

```
attr obstacle = brightness < 0.5
```

Steps:

1. Hide the Obstacle Layer in the Scene Editor so it does not interfere visually with the terrain.

2. Open the **Grow Streets** dialog again and assign the Obstacle Map.
3. Grow again.

   Notice how the streets try to avoid the black areas of the image.
4. To grow more streets in specific regions, select some existing graph segments before growing more streets.

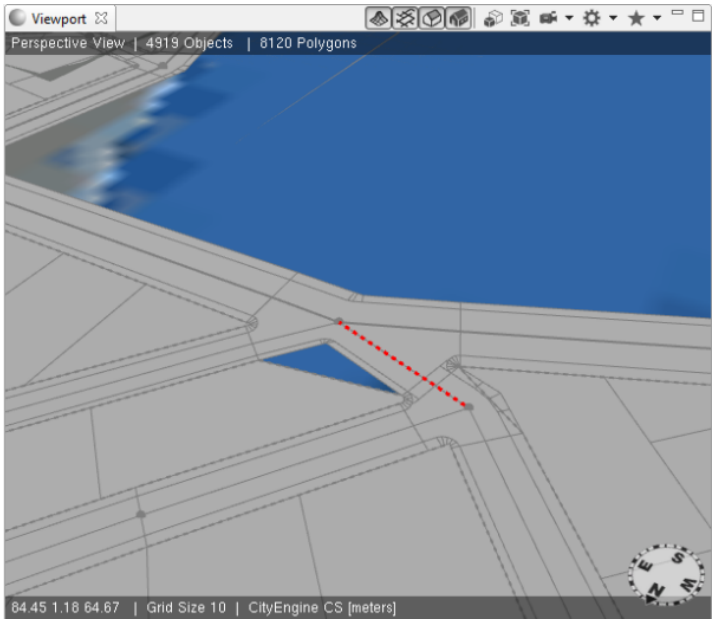
### Part 3: Streets: Cleanup Tool

Under certain circumstances, it may be the case that CityEngine's **Dynamic City Layouts** system is not able to solve a conflicting part of the street network as proper Shapes. In that case, CityEngine will highlight the conflicting graph segments red.

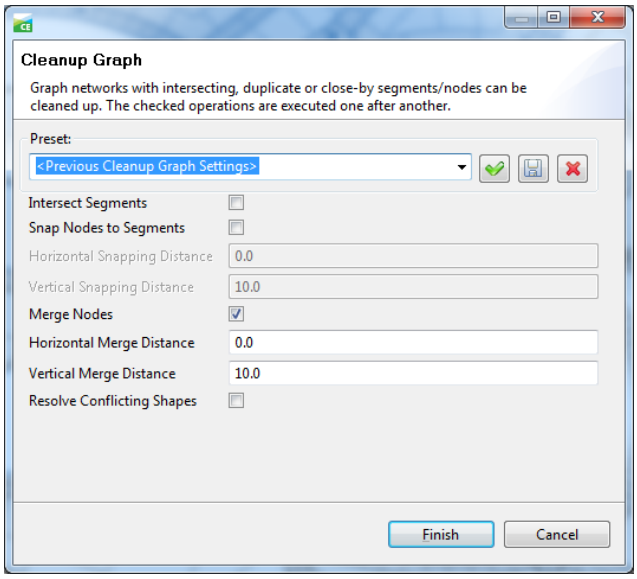
To solve this, you can of course go in and manually fix the issues by editing street widths, curve radii or other parameters, but this can be tedious work. A simple shortcut usually is to use the **Cleanup Graph Tool**.

Steps:

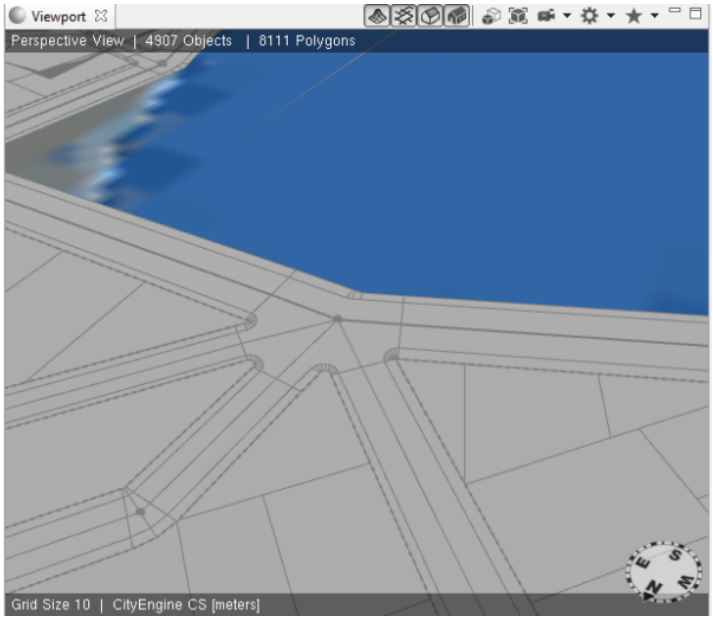
1. Find a conflicting graph segment (or produce one).




2. Select the objects around the conflicting region. It does not matter if also Shapes or other object types are selected.
3. Start the Cleanup Graph Tool from the toolbar.




4. Play with the settings and read the **Cleanup Graph Tool** documentation in the manual.
5. Your network should now be fixed, as here:




## Part 4: Aligning objects

In reality, streets usually follow the surface of the terrain or buildings are placed on the terrain. Since it's rare that newly created objects in a scene are directly aligned properly – e.g. after the import of 2d GIS data – specific alignment tools are a necessity. CityEngine provides a series of commands to align certain object types to others.

The common alignment types are:

| | | |
|---|---|---|
| Graph Segments | > Terrain | Example: Project street segments onto the terrain |
| Static Shapes * | > Terrain | Example: Project footprint shapes onto the terrain |
| Terrain | > Shapes | Example: Align the terrain to shapes (street bank, ...) |

**Note:** *Shapes which were manually drawn or imported are so-called **Static Shapes**. CityEngine's own system to create Shapes – the **Dynamic City Layouts** – creates Shapes dynamically, thus they're called

Steps:

1. Select all Graph Segments in your scene.
2. Click the **Align Graph to Terrain** button on the toolbar.
3. Choose your terrain as the **Heightmap**.
4. Maybe add an offset of `0.2`.
5. Click **Finish**.

   Note that not a lot is going to change since the streets were already aligned.

   Depending on your terrain, you may need to adapt the **Maximal raise distance** value.

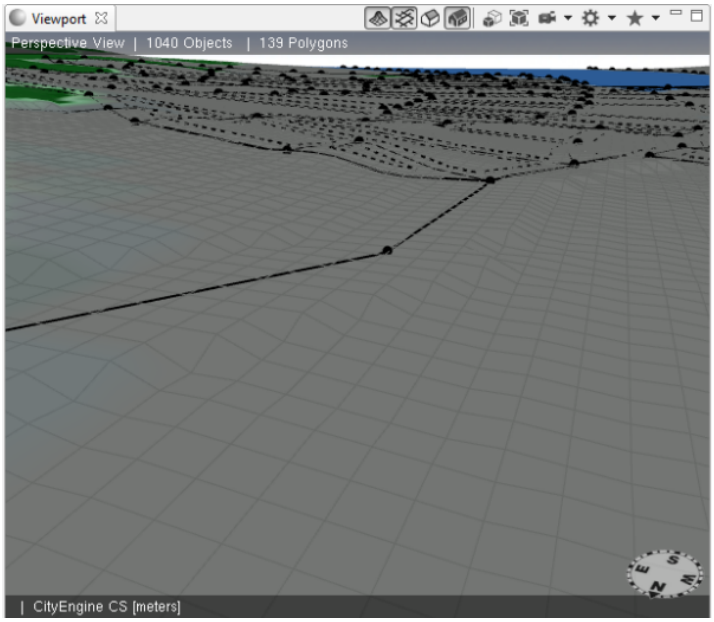
   For further details on the settings, please consult the Manual in the Help Menu **F1**.

In this scene, we do not have any Static Shapes, so let's continue with aligning the Terrain to the Shapes we already have!

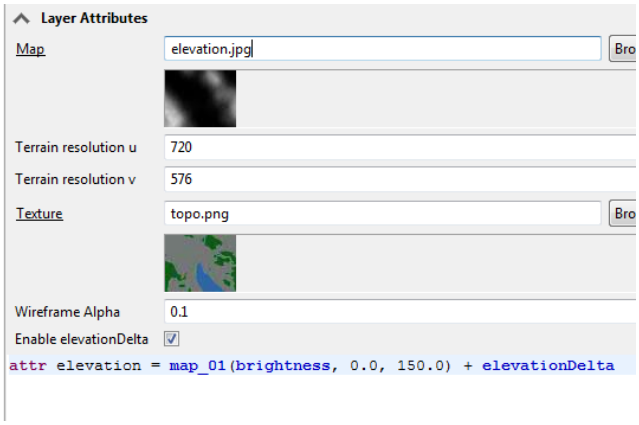
Steps:

1. Select all the shapes in your scene (it does not matter if you also have Graph Segments selected).
2. Click the **Align Terrain to Shapes** button on the toolbar.

   Notice how your terrain was adapted. The differences may not be very prominent, depending on your terrain. To get a stronger effect, edit the terrain's max height.

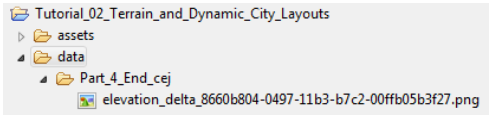



3. Open the terrain's **Layer Attributes** in the Inspector.
4. Toggle the **Enable elevationDelta** checkbox a few times.

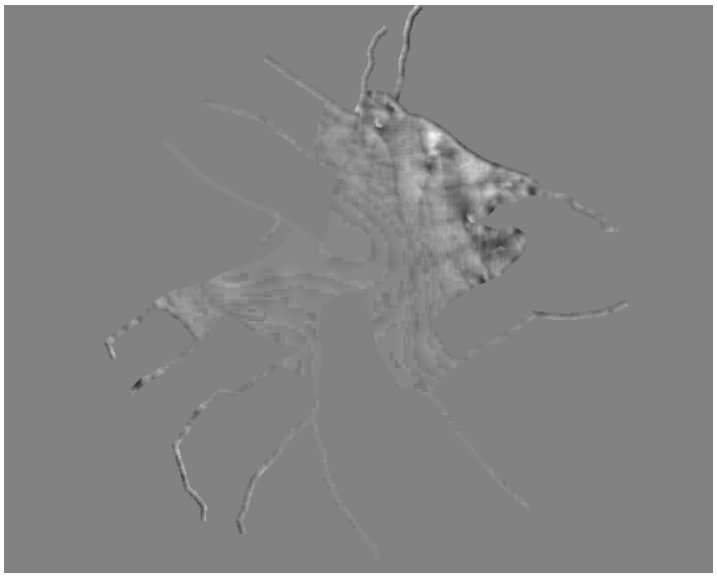



Note how the terrain switches back and forth between the **unaligned** and **aligned** state. This must mean that somehow the alignment data (the **elevationDelta**) must have been stored!

5. Save your scene.
6. Refresh the project's **data** folder. (e.g. right-click on the folder, then use **refresh** from the context menu).
   In the **data** folder, you will now find a new folder which carries the same name as the scene file. Note that this folder and the **deltaMap** does not exist unless you save your scene after the first terrain alignment.




The (16bit) delta map which stores the changes (Grey = no change; Black = lower; White = higher). Note that in the image may look quite different in your example.



## Part 5: Dynamic city layouts

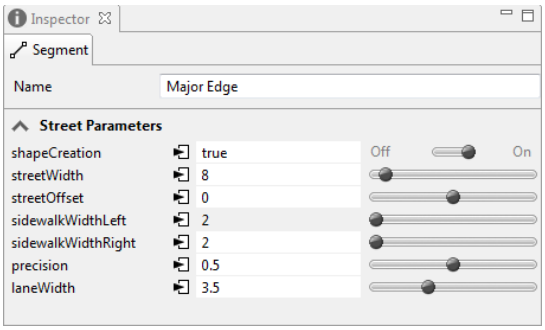
### Object types

**Dynamic City Layouts** is a set of tools and functionalities which allow the user to quickly create city layouts. The result is a series of shapes, ready for use with CGA rules to create 3d models. The main elements are Streets and Blocks. Blocks are created if the street center lines physically enclose an area (and the network is cleanly connected). Blocks are represented with a dashed line, which you can select.

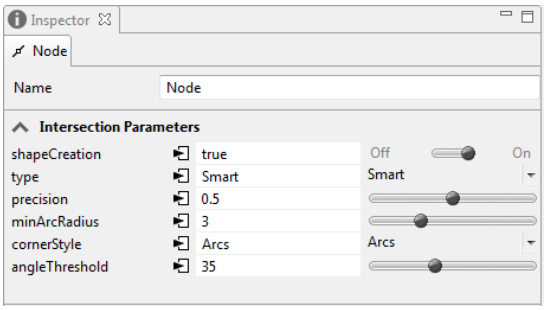
Both Street Segments and Blocks have individual parameters, which drive the creation of their children, the **Dynamic Shapes**.

When you play a little with the streets we have grown so far, you will find out that there are 3 object types selectable whose parameters drive and influence the Shape Creation.

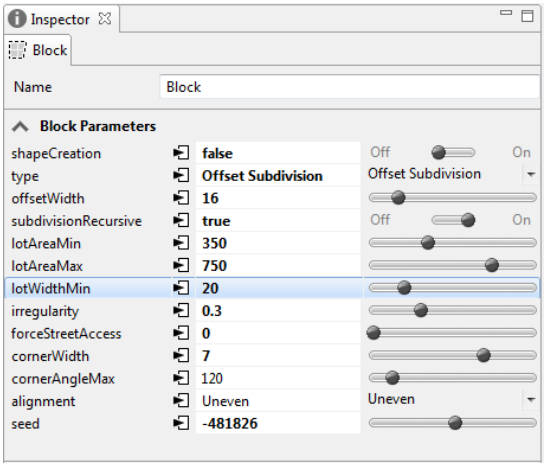
*Graph segments:*




*Graph nodes (which are the 'ends' of a Graph Segment):*




*Blocks:*

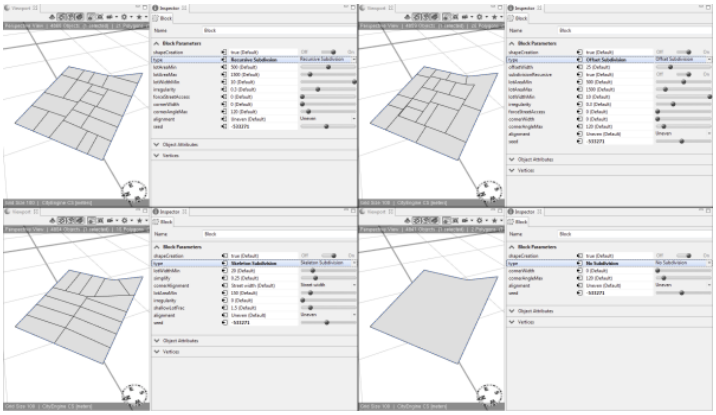



Common to all of those 3 object types is the **shapeCreation** parameter, which lets you drive whether you want those specific object types to produce shapes, or not. E.g. you may just want to get Street Shapes, but no Subdivision in the Blocks. Again: **Dynamic Shapes** are the result of the Shape Creation of Graph Segments, Graph Nodes and Blocks.

That said:

- Play with all the parameters you find on all the object types for a few minutes!
- Create some roundabouts. Edit their radii.
- Create some cul-de-sacs.

Let's now shortly focus on the **Block Subdivision**. Each **Subdivision Algorithm** has its own parameters and creates its specific pattern.



An important note: The Shape Creation is a purely interactive task, using the **Dynamic City Layouts**. The Shape Creation or Block Subdivision thus is a task which has nothing to do with CityEngine's rule system (CGA). The Shapes must first exist (be created), before a CGA rule can be assigned to the Shapes for the final Generation of the 3d Models.

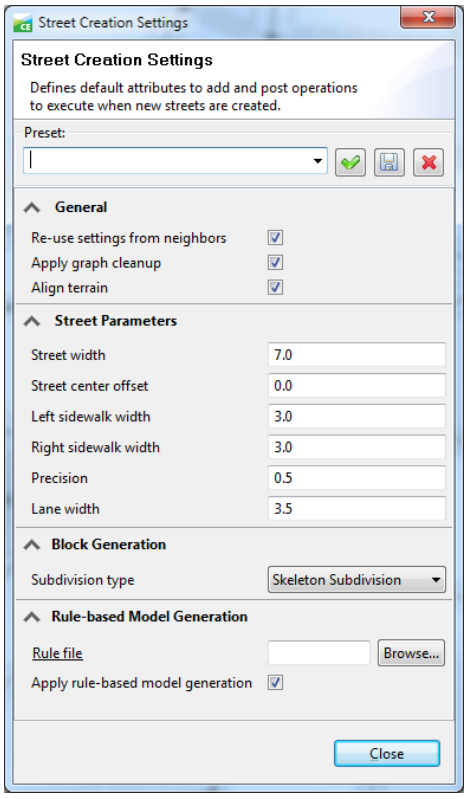
Nevertheless, there is the option to subdivide a full Block Shape (or any other Parcel Shape) using CGA.

## Streets: Creation

Next to the option of importing Graph Segments, CityEngine provides two tools which are dedicated to the creation of Street Graphs, located in the toolbar: The **Freehand Street Creation Tool** and the **The Polygonal Street Creation Tool**.

Steps:

1. Use both tools to draw some streets.
2. Go to **Graph** > **Street Creation Settings...**
   This dialog lets you define your **Street Brush**.

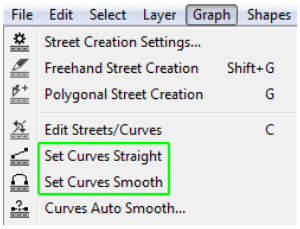



3. Play with the Street Creation Settings and draw some more streets.

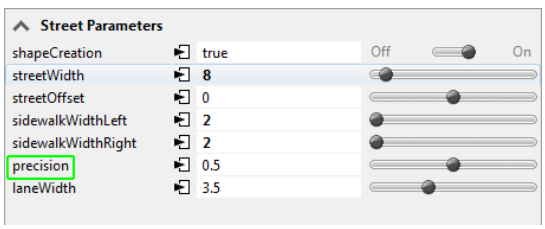
You certainly have noticed that streets can be created with straight or curved segments:

Steps:

1. Select some Street Segments
2. Play with the commands **Graph** > **Set Curves Straight and Graph** > **Set Curves Smooth**



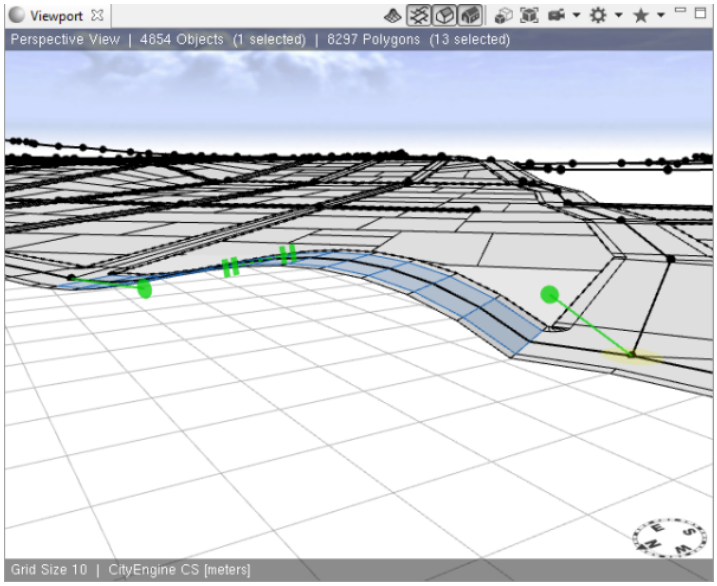

Note that when having individual Graph Segments selected, you can also edit the **precision** parameter to drive the precision of the Subdivision (number of segments) for curved streets. Keep in mind that curved streets produce many polygons, so keep them at a minimum!



## Streets: Editing

Steps:

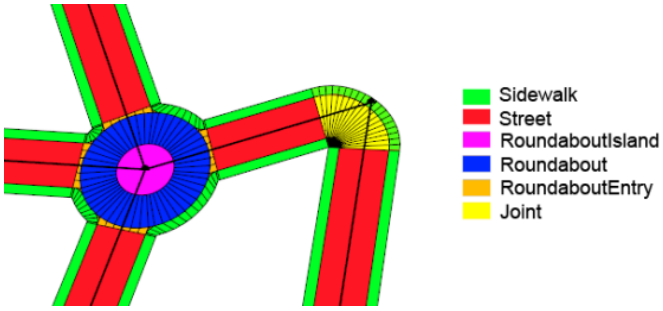
1. Enter the **Edit Streets/Curves Tool** via the toolbar or the c keyboard shortcut.
2. Edit the widths of some streets and sidewalks.
3. Edit some tangents. Note that depending on the viewing angle, only the horizontal or vertical tangents can be edited.




## Streets: Model Generation

Steps:

1. Explore the StartRules of the streets you have created in the Inspector. You will find 5 types (you need to change the **type** parameter of some Street Nodes to find them all):


2. Assign the provided **simpleStreets.cga** rule to all streets and generate the models.

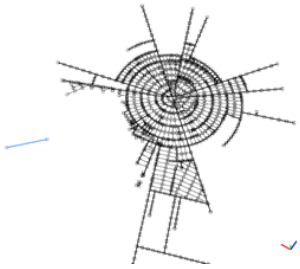
3. Continue editing the streets while the models are being regenerated on the fly.

## Part 6: Street Pattern Examples

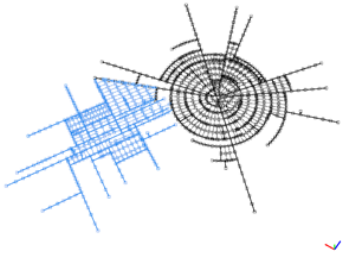
### Workflow examples as impressions

Steps:

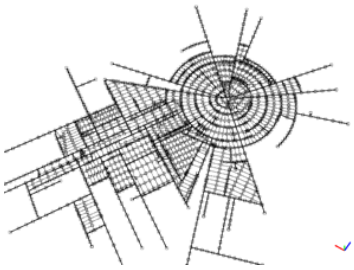
1. Create a separated single street:



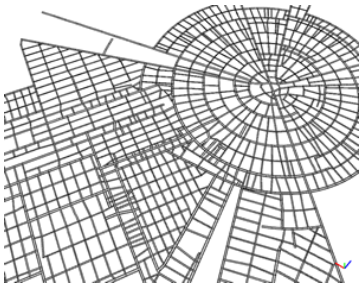
2. First growth phase with raster pattern:



3. Second growth phase with raster pattern:



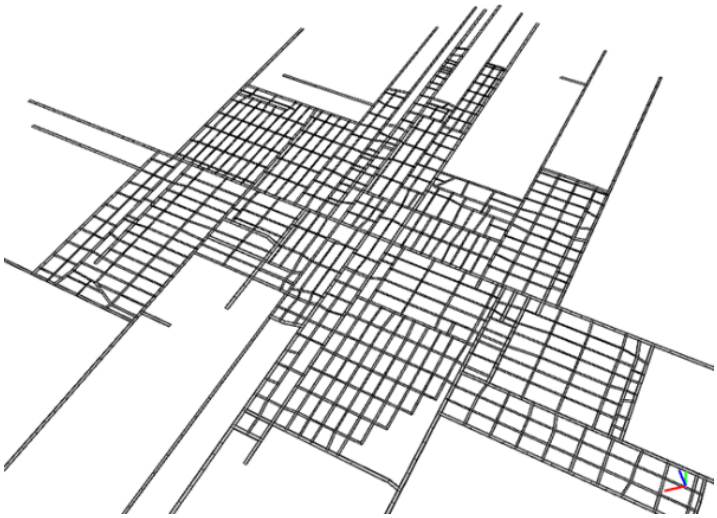
4. The two networks are connected:

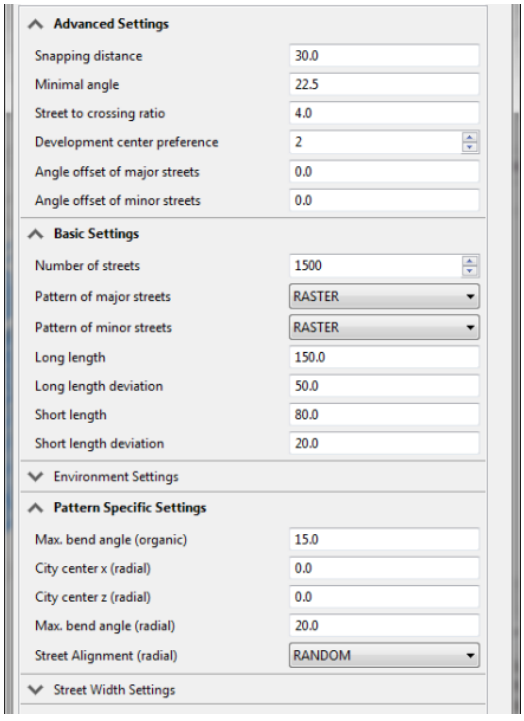


### Parameter Sets

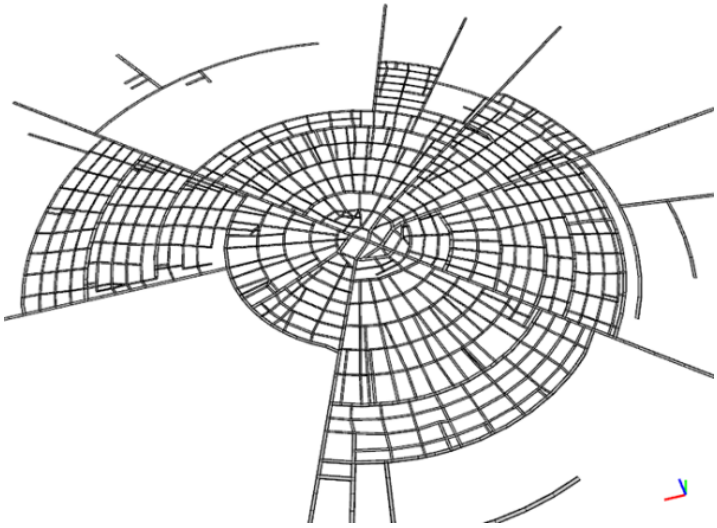
Steps:

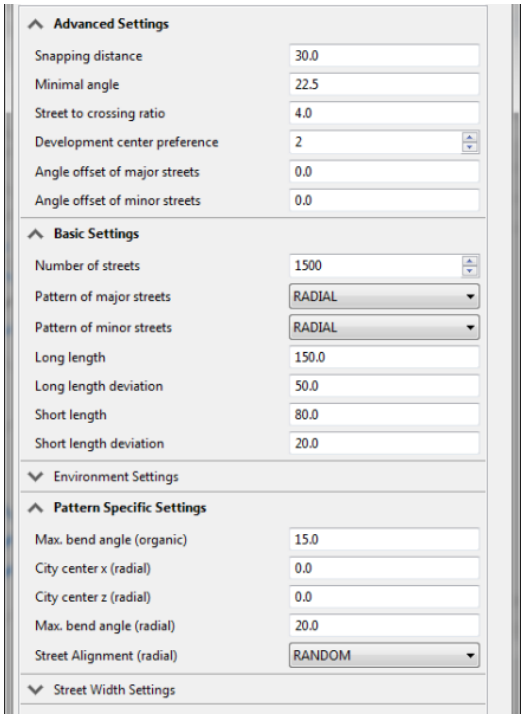
1. Raster Pattern:



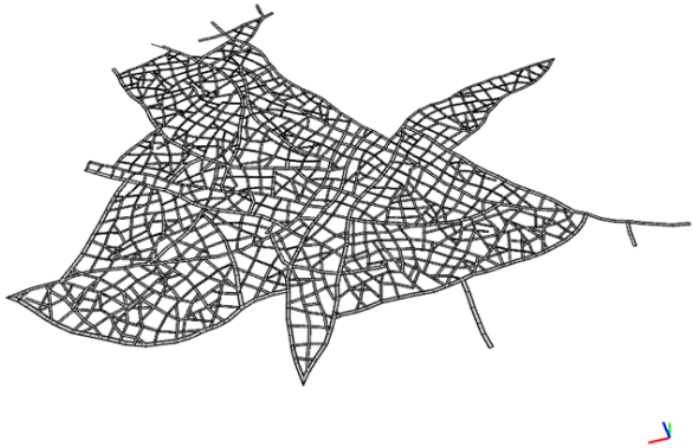


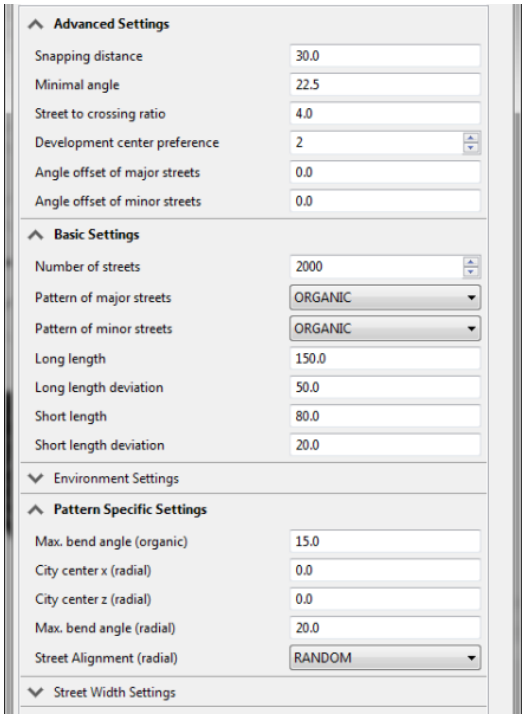

2. Radial Pattern:



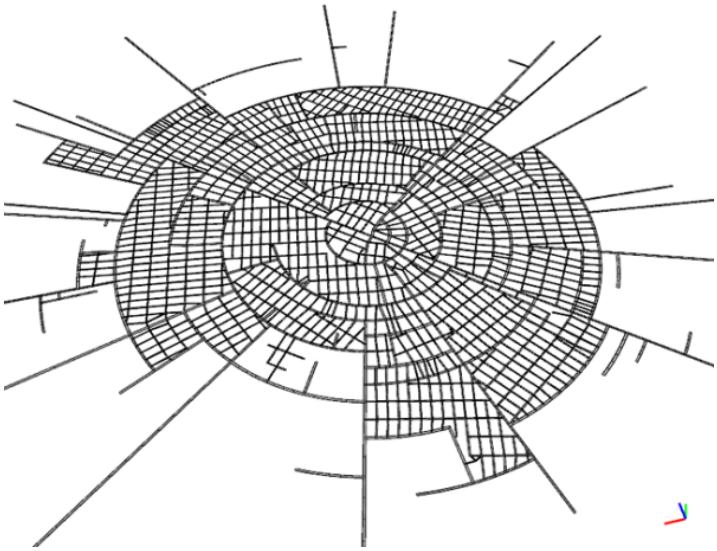


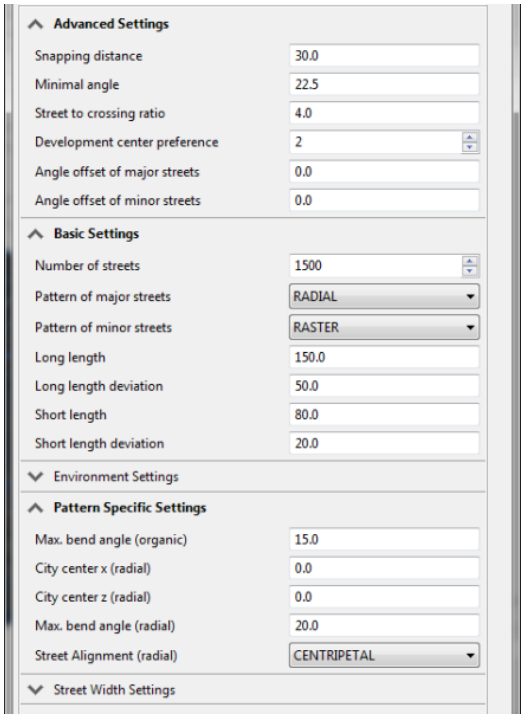

3. Organic Pattern:



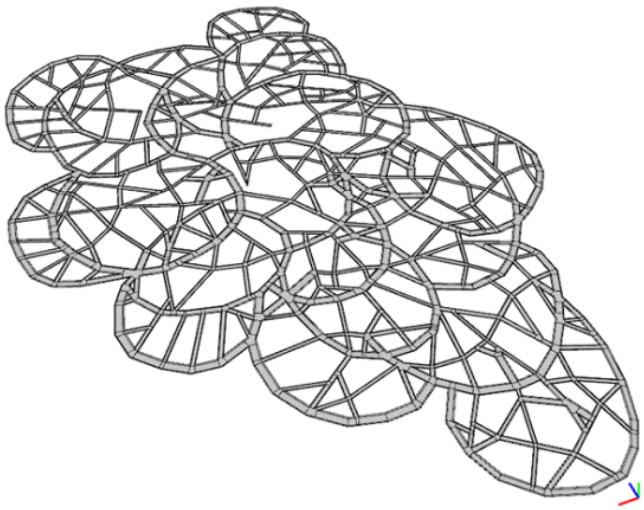


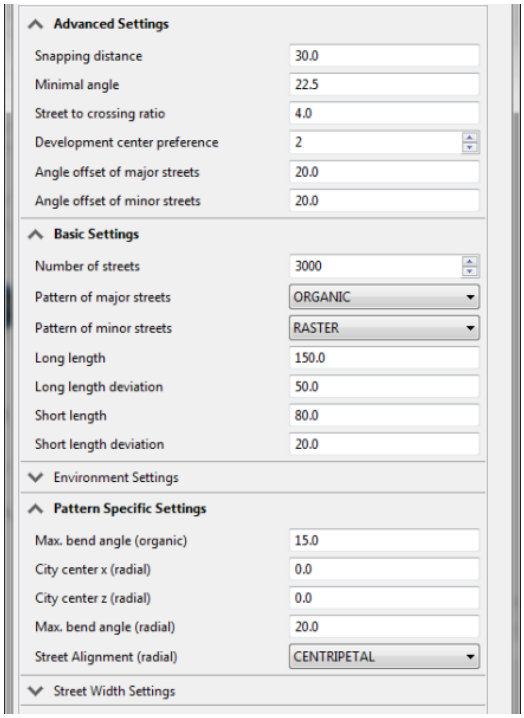

4. Radial major streets with raster pattern on minors:



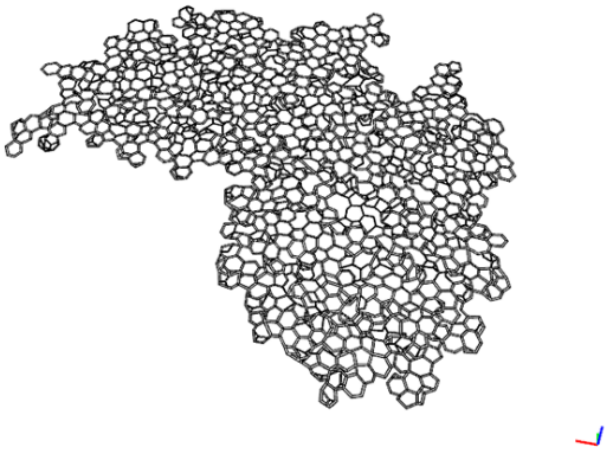


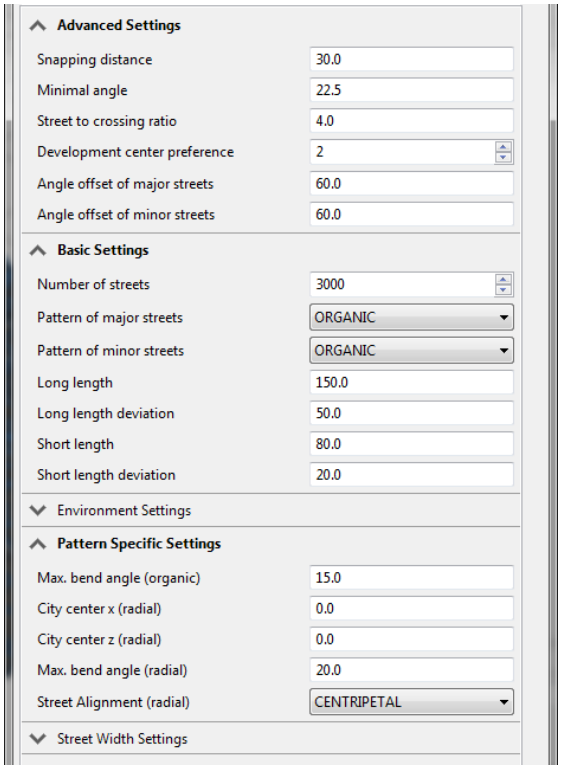

5. Organic circle pattern:



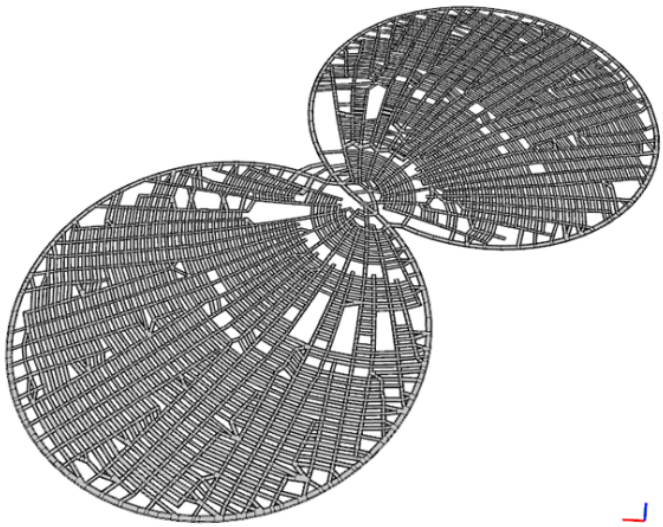


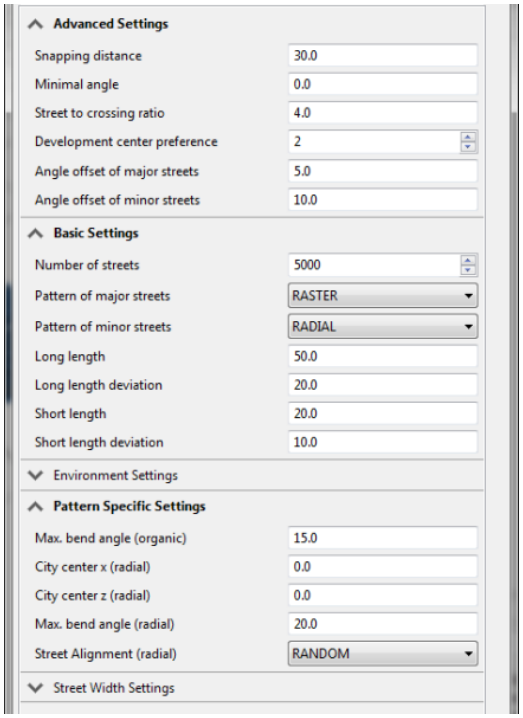

6. Honeycomb style:






7. The Glasses:






8. Organic distribution of rasters: