City Generator

This python script does 2 main tasks:

1. Procedurally generates a street layout according to the directional constrains placed by the user.
2. Generates buildings around the streets using user-defined groups and their responding paintable maps to create a varied cityscape.

Manual

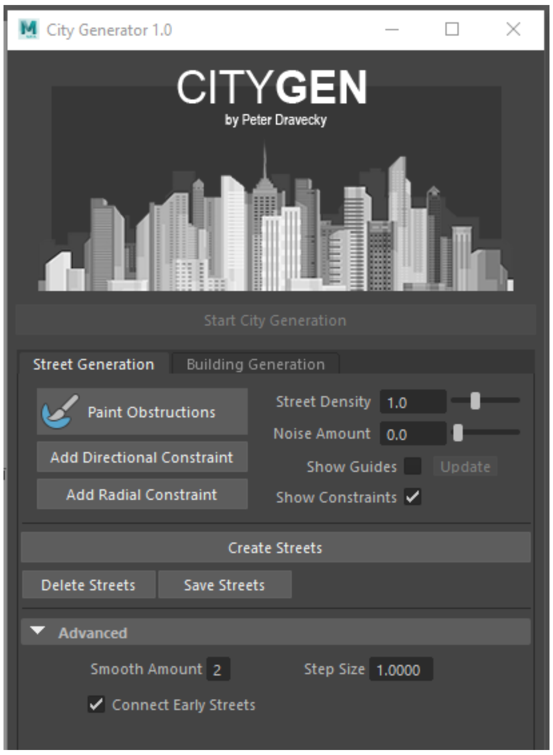
Before running the script, make sure the insides of the src folder are placed into the scripts folder of your Maya workspace.

How to use UI:

Press *"Start City Generation"* to start using the menu.

There are 2 main tabs:

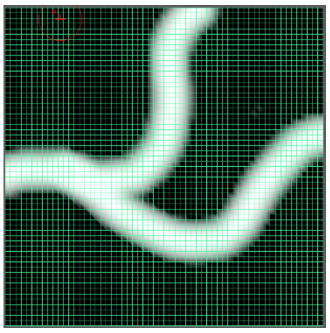
Street Generation – To create streets, set the relevant settings and press the *"Create Streets"* button. This will create the surface the city will be generated on. The user can resize the surface to the size they need.

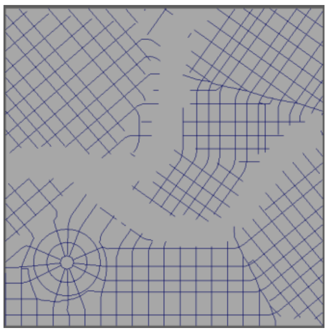


Street Density 1.0
Noise Amount 0.0
Show Guides ☐
Show Constraints ✓
Smooth Amount 2
Step Size 1.0000
✓ Connect Early Streets

Options:

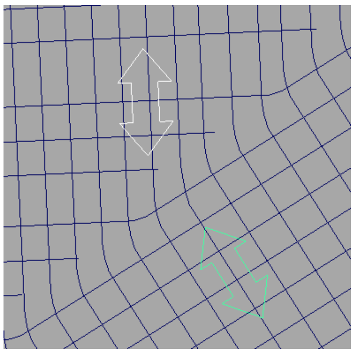
• **Paint Obstructions –** Opens a painting context to paint obstructions – areas painted in white will not have streets nor buildings generated inside of them. Double clicking the button will open Paintable Attribute Tool Window.

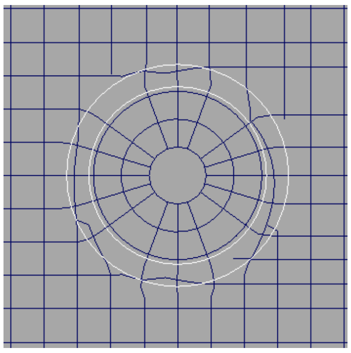


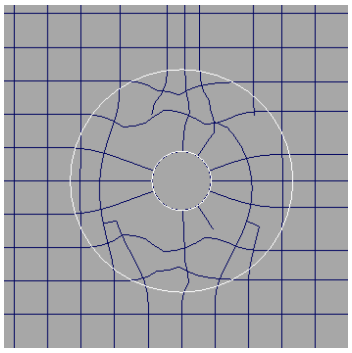


• **Add Directional Constraint** – Adds a double-sided arrow to the surface, which can be moved and rotated. Its rotation controls the direction of the street in that area.

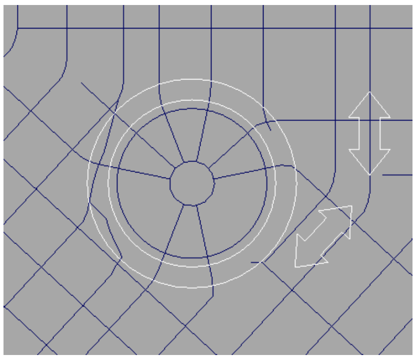
• **Add Radial Constraint** – Adds a circle, which can be moved and scaled, to the surface. The roads within the radial constraint will create a circular pattern. The inner circle controls the influence/decay of the circle on the roads and it can be scaled separately.

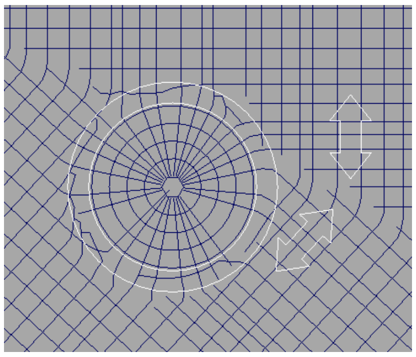

Directional Constraint


Radial Constraint with original decay

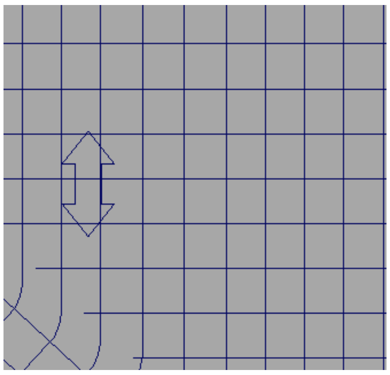

Radial Constrain with smaller decay circle

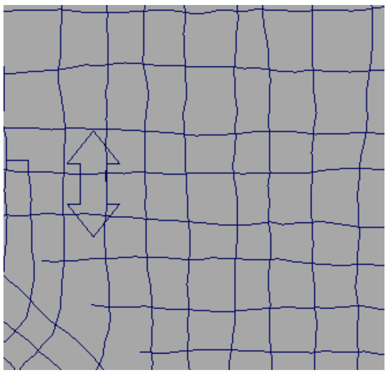
• **Street Density** – Controls how densely the streets are generated.


Street Density set to 1.0


Street Density set to 2.0

• **Noise Amount –** Controls the noisiness of the streets being generated.


Noise Amount set to 0.0


Noise Amount set to 10.0

• **Show Guides** – Shows a field of crosses which visualize the direction of streets before they get made. The Update button updates the field to the new position of the constraints.

• **Show Constraints** – Toggles the visibility of the constraints.

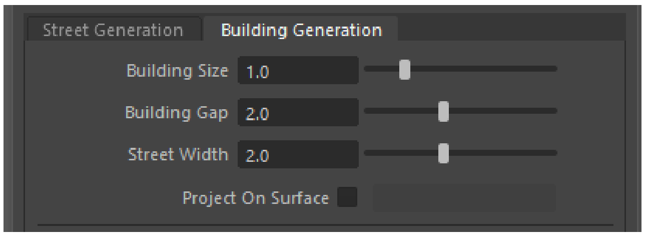
Advanced Settings – Optional experimental settings. They do not really change much but I wanted to leave the option for additional customizability.

• **Smooth Amount** – Default is 2. Makes the change from one direction to the other smoother.

• **Step Size** – Changes the step size at which the street generation algorithm checks for the change of angle. Lower amounts make the program more precise but much slower.

• **Connect Early Streets** – By default, when a street being generated gets too close to another streets, it ends the generation and connects those streets together. This option disables the connection part.

Building Generation – To generate the buildings, change any relevant settings, paint the Building Placement for each building group (important, otherwise no buildings will be generated) and press the *"Generate"* button.



Building Size 1.0
Building Gap 2.0
Street Width 2.0
Project On Surface

Options:

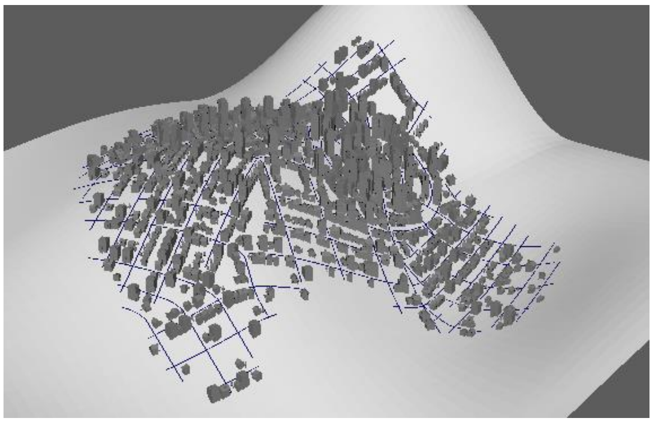
• **Building Size** – Changes the size of the building.

• **Building Gap** – Gap between the buildings being placed.

• **Street Width** – Controls how far away from the street curve the buildings should be placed.

• **Project On Surface** – Allows the generated city to be projected onto a surface/plane, allowing the city to be generated into a pre-made environment. After enabling this option, you have to input the name of the plane to be projected onto into the text field next to the option.

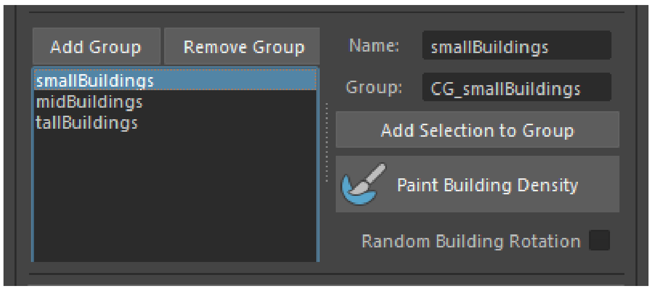
The surface must be big enough to have space for all the curves, otherwise the projection will not be executed.


City projection onto a deformed surface

• **Building Group Options**

This pane contains controls for each building group. To create a usable building group:

1. Press the *"Add Group"* button
2. Either assign an existing name of group of buildings to the *Group:* tab, or type in a new name, select the objects you want to add to the new building group and press "*Add Selection to Group".*
3. Paint the Building Density.



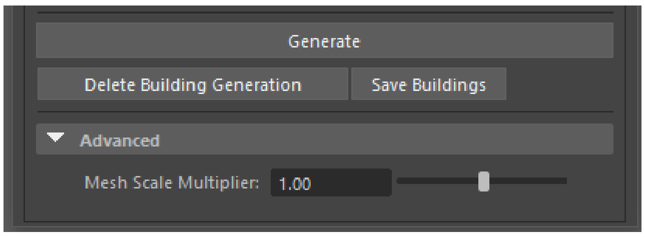
smallBuildings
midBuildings
tallBuildings
Name: smallBuildings
Group: CG_smallBuildings
Random Building Rotation

Each building group has its own settings.

• **Paint Building Density** - Opens a painting context to paint the building density on the surface. This allows to blend different blending groups together to create outskirts, city centre, the skyline, etc, like in the example below.

• **Random Building Rotation** – Makes it so that the generated buildings do not always face the street they were generated at.

• **Mesh Scale Multiplier –** When the mesh is scaled through the *"Building Size"*, it changes the offset of the building from the street and other buildings to account for the change of scale. This option scales the object without any other changes to generation.



Mesh Scale Multiplier: 1.00

Implementation

The street generation part of the program was heavily based on the paper "Interactive Procedural Street Modeling" by Greg Esch, Peter Wonka, Pascal Muller and Eugene Zhang, and a web-based Procedural City Generation Tool by ProbableTrain.

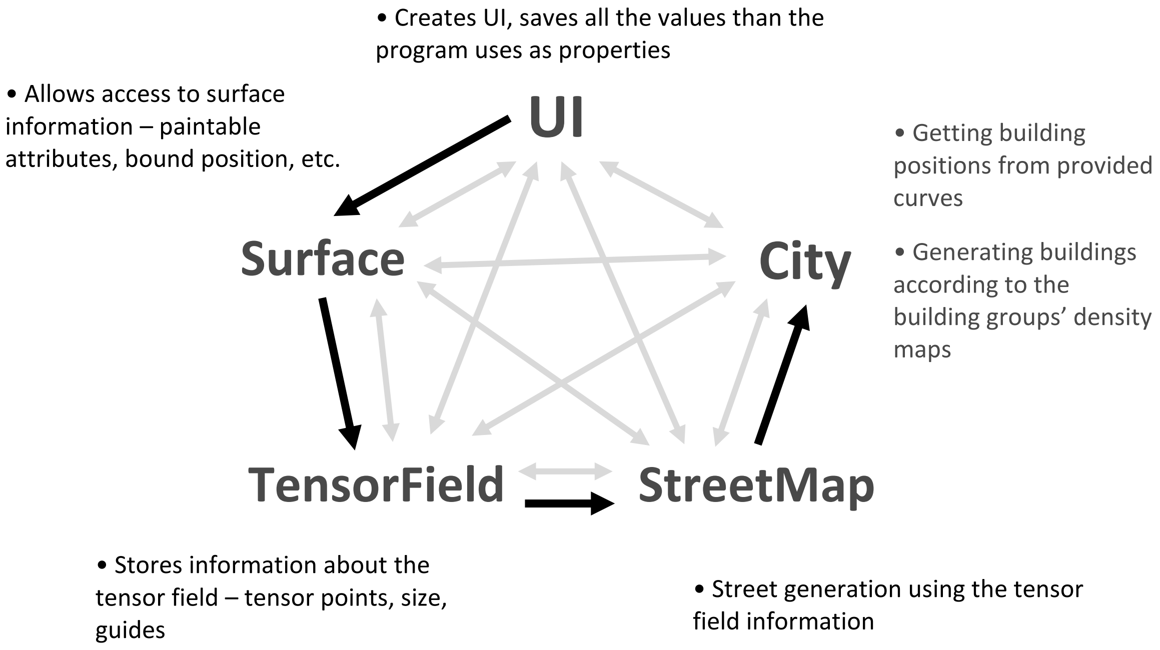
Term Definition:

• **Tensor field** - a normalized vector field, where each vector can be influenced by directional and radial constraints

• **Tensor point** – name in the script for the tensor field constraints. They influence the rotation of the vectors within the tensor field

• **A tensor** – a point inside the tensor field. It contains the vector information as two angles – "major" and "minor", which are always perpendicular to each other.

The script is centered around *5 class objects* which are initialized at the start of the program – **Surface**, **TensorField**, **StreetMap**, **City** and **UI**. These classes are passed into each other as properties, so that they can access each other's values at any point.


• Creates UI, saves all the values than the program uses as properties
• Allows access to surface information – paintable attributes, bound position, etc.
UI
• Getting building positions from provided curves
Surface
City
• Generating buildings according to the building groups' density maps
TensorField
StreetMap
• Stores information about the tensor field – tensor points, size, guides
• Street generation using the tensor field information

Tensor Field Constraints

Adding and updating the tensor points is handled by the TensorField.TensorPoint subclass, and getting the tensor information at a certain position is handled by the TensorField.Tensor subclass and its function *Tensor.setRotation().* This function goes through the list of tensor points, finds which one is the closest and sets its rotation according to the rotation of the tensor point.

Street Generation

Generation of the layout is handled by the StreetMap class. Generation of all streets is executed through the *StreetMap.createMap()* procedure. Streets are sorted into 3 categories:

- major – horizontal streets
- minor – vertical streets
- radial – circular streets around radial tensor points

First the program generates starting points on the bounds of the surface for minor and major streets using the function *StreetMap.getStartPointMap()* for each type.

Generation of each street is executed inside a subclass of StreetMap called Street.

For **major** and **minor** streets, the procedure *Street.createDirStreet()* starts at a point and gets the tensor angle at that point. Than it moves by a step in the direction of the angle. It repeats this until it either gets to the other end of the bounds or it gets too close to the previously made street. If it gets too far away from the previously made street, it recursively executes itself again at the position inbetween the two streets. Each street is a list of point arrays that get turned into curves at the end of the procedure.

The **radial** streets are separately for each radial tensor point in *Street.createRadStreet()*. They also go step by step, but now they rotate around the center point of the tensor. They finish when they complete the circle.

Building Generation

Building generation is handled by the *City.generateCity()* function.

First, it creates 2 offset curves to the each side of the streets, along which the buildings will be generated. The procedure *City.getBuildPositions()* then finds all the positions along the provided curves. It does so by calling *City.getStreetSegments(), which* starts at one end of the curve and going forward, checking for intersections with other curves. Once it finds it, it saves the street segment and moves forward. This splits each street into segments of different lengths. These lengths are then passed to the *City.getCoordsFromSegments()* function, where they are divided by the space, each building needs, and the remainder is split between the building coordinates to create an even building distribution on each street segment. In the end it the procedure *City.fixCollision()* compares every single new point with previously made points and the obstruction map and deletes the points which are either too close or obstructed. This whole process is done street by street and the end result is a dictionary containing each street as a key and the building positions as an array of points belonging to each key.

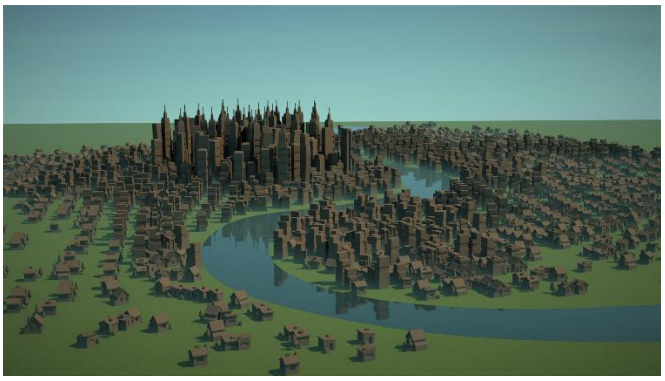
The construction dictionary is then used by the *Street.generateBuildings()* function. For each positions it checks the density map for each building group. From those values it compiles a list of buildings it can use for the specific position, from which it randomly picks one and places it.

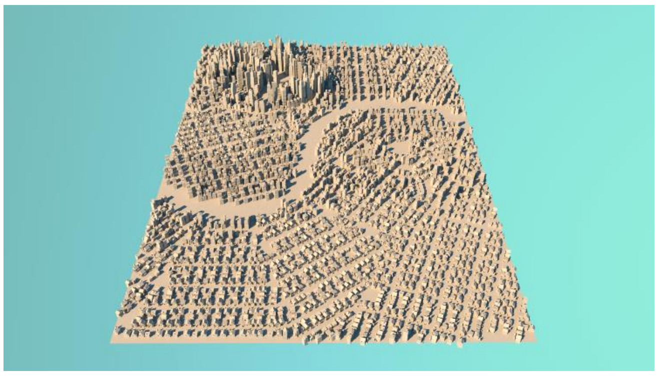
All the position comparing and calculations are done in 2D on the X and Z axis, with the Y coordinate of the curve being added to the building coordinates at the end, if projection is required.

Reusability

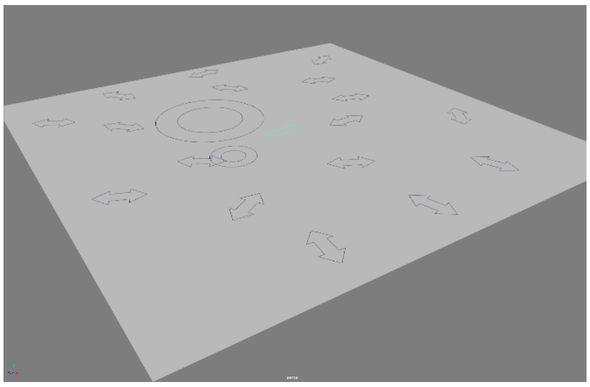
The script recognizes objects used in previous sessions – it can use tensor points from previous sessions for city generation and assigns the building density maps of the surface correctly.

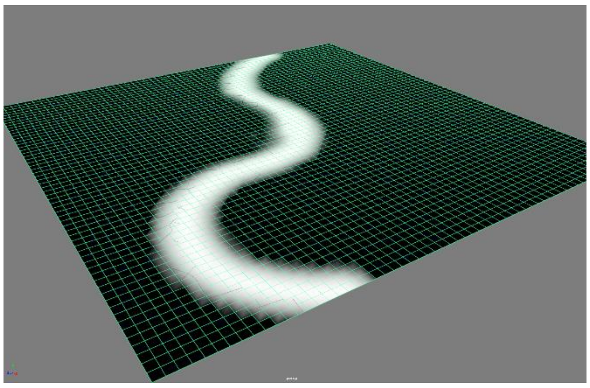
As the building generation works with any curves, the user can go in and delete, move or add current curves and add new ones – as long as they are placed into the program's street group in the outliner, the building generator will use them.

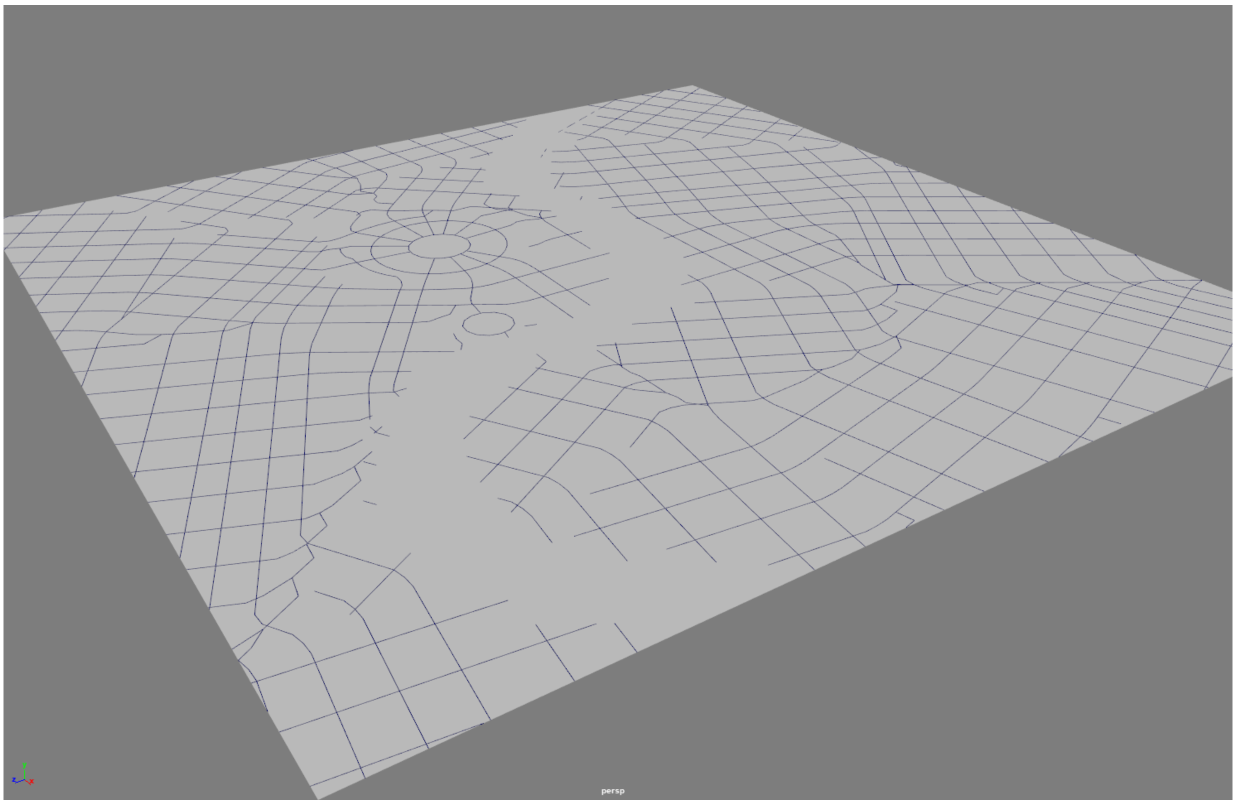




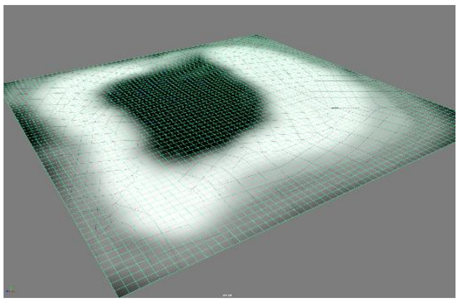
Workflow Example:

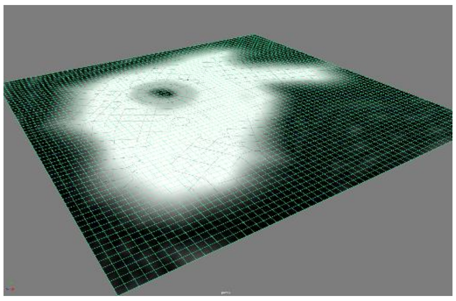

User-placed tensor points

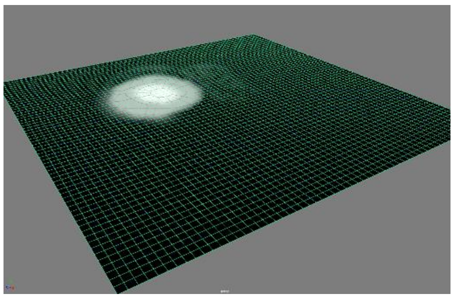

Painted obstruction map

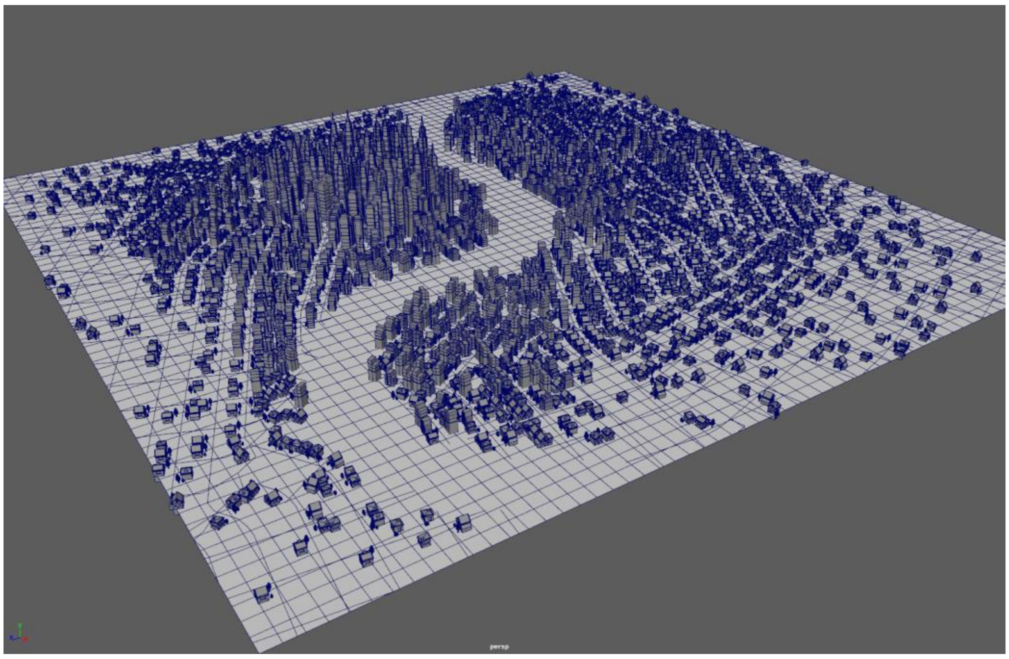


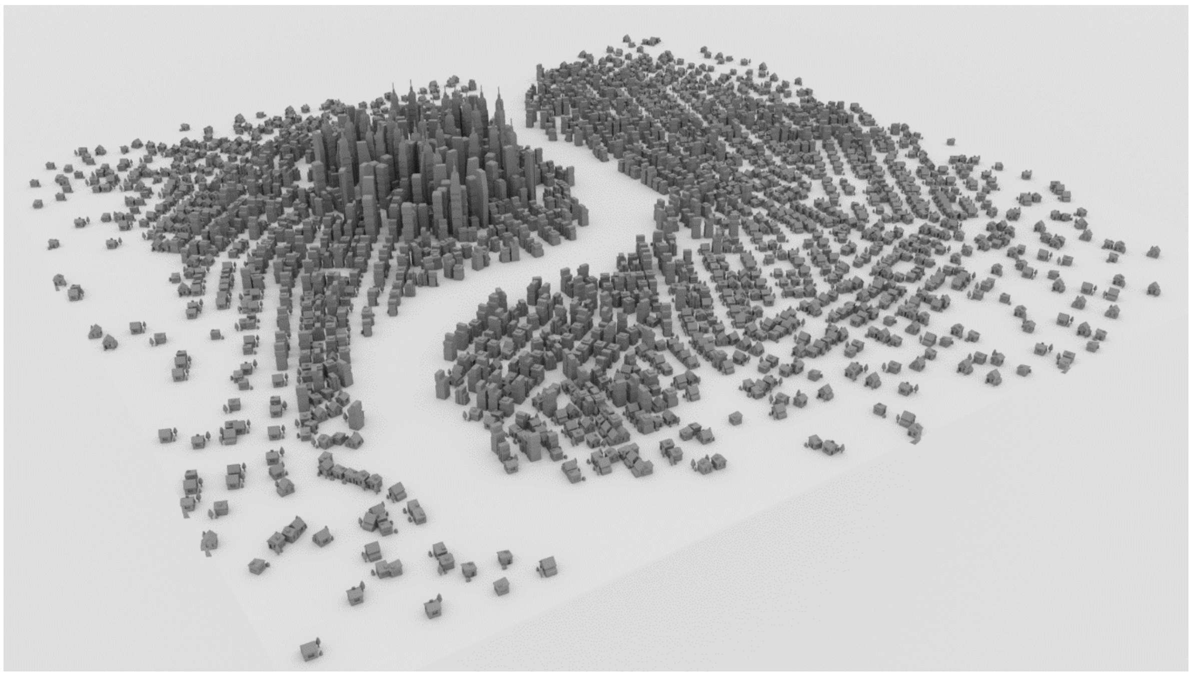
Generated street layout


Urban buildings density map


Mid-sized buildings density map


Highrise buildings density map


Generated buildings


Rendered scene