Building a procedural city

Gustavo Patow
Geometry and Graphics Group (GGG)
Universitat de Girona (UdG)

Street Layout: Grid

- Let’s start by creating a Grid that’ll be the outline of our city
Let’s adjust our **Grid** to
- 60 x 60 units
- 11 x 11 divisions
Street Layout: Blocks

- To build the blocks, let’s create a simple Polyextrude...

And set:
- Inset = -0.54
- Only check the “Output front” option
- Disable the other “Output” options
Street Layout: Blocks

- Let’s create some space for sidewalks with another **Polyextrude** node.
- It’s only job is to reduce the block area to leave space for the sidewalk.

Street Layout: Sidewalks
Street Layout: Sidewalks

Lots: Partition

- We are going to divide each block into lots
- Before that, we need to identify each block with a unique ID
- That’s what the Partition node is for!
- Set the rule “lot_$PR”
This node is useful for iteratively applying operations.

In this case, we want to do something for "each group".

And operate over the groups labeled lot_*

Inside the **Foreach** node, we find its inputs and an **each** node.

This node represents each element the loop is operating onto.

It’s already set, so do not touch it! 😊
And let’s create a Grid that will subdivide each block into lots.

Basically, we will generate a set of boxes that will be intersected with the block to generate lots.

**Lots: Layout**

- Center x: centroid("../each1", D_X)
- Center y: centroid("../each1", D_Y)
- Center z: centroid("../each1", D_Z)
- Rows: if(rand(centroid("../each1", D_Z)*centroid("../each1", D_X))>0.3,3,2)
- Columns: if(rand(centroid("../each1", D_Z)+centroid("../each1", D_X))>0.4,3,2)

**Lots: Grid parameters**
Now, we will reduce each grid element a little bit with a **Polyextrude** to be sure each one will generate a lot.
OK, not very visible...

Now, let's convert the grid boxes into a 3D structure.
And now, the actual intersection!

For that, we will use a **Cookie** node with
- a “user defined” operation
- “Keep Inside A” selected (the first input, the block)
Lots: Layout

- Moving one level up...
Lots: labeling

- Our buildings created with buildingEngine require each lot to be labeled “lot”, so let’s add a Group node to do it.

Copy-stamp loop

- A Copy-stamp loop is a common idiom in Houdini.
- It needs a delete and a copy nodes (rename).
Copy-stamp loop

- In the copy, set the number of copies to nprims(‘../group1’) 
- And create a stamp variable called currLot with an expression $CY$
- Do not forget to activate “Stamp Inputs”

Copy-stamp loop

- In the delete, set the group to ‘stamp(“../finalcp”, “currLot”,0)’ 
- And disable “Enable” in the Number tab
Adding the buildings

- Finally, we can add the building we have just constructed!
- (or a texture-based version of it...)

![Diagram of building creation](image)

- Height: $6 + 4 \times (\text{rand}() \times \text{centroid}(\text{../selectLot,D_X}) + \text{centroid}(\text{../selectLot,D_Z}) \times \text{bbox}(\text{../group1,D_SIZE})) \times 2 - 1$
- FirstFloorHeight: 1
- FloorHeight: 1
- DoorWidth: 0.5
- WindowWidth: 0.5
Viewing the whole picture

More city content

- Sidewalks
- Trees
- Parks
- …
Sidewalks

- A simple Polyextrude
- With a global translate of -0.3 (to be below the buildings)

But do not forget to check “output front”, “output back” and “output side”
Sidewalks

Let’s create an Object_merge to gather our urban elements

Viewing the whole picture

- Let’s create an Object_merge to gather our urban elements
Sidewalk Trees

- Let’s create another **Polyextrude** (only “front output” enabled) with a small negative inset

- But 4 are not enough points along this curve
- Let’s **Resample** the curve and put a point every 1 unit length
Sidewalk Trees

- For the trees, a simple **L-System** will be enough...

Sidewalk Trees: Copy

- We can **Copy** the trees to every point along the curves! (rename to “trees”)

Problems?

- The trees are on the sides!

Sidewalk Trees: Point

- We only need to add the tangent (0,0,1) with a Point node!
- But the Copy node needs the normal!
Viewing the whole picture

- Add the trees to the **Object_merge**

Creating a central park

- First, let’s visualize the block IDs...
- And select where our park will be
Creating a central park

- Create a **Delete** node erasing the unwanted (selected) blocks
  - do not forget to reconnect the nodes

Creating a central park

- We need another delete, but this time deleting the “Non-selected” blocks...
Creating a central park

- To have a single surface, let's **Bound** the blocks and **Delete** the bounding faces we do not want to keep.

Central park: Trees

- A simple **Scatter** node will create the (200) points we need.
Central park: Trees

- An **L-System** will generate a tree

Central park: Trees

- That we will **Copy** to every point!
- Rename the **Copy** node to “park”
Final View

- Adding everything to the `Object_merge`
  (with a couple of extra things)
Conclusions

- We have built a full procedural City
- Using the **building** Engine module and Houdini

- **skyline** Engine automates most of this process...
- ... and generalizes it with tons of new options!