Project 1: Procedurally Modeling a Building

Brooklyn Brownstone User Guide & Breakdown

Rendering Statistics

Renderer	-	mantra
Average Render Time	-	4.5 min / frame
Image Resolution	-	1280 x 720
Number of Lights	-	2 (Sun and Skylight)
Geometry Complexity	-	204 Polygons per House

Sampling

Noise Value	-	0.01
Min/Max Rays	-	1/9
Diffuse	-	2
Reflection	-	3
Global Quality	-	1

User Guide

This tool can be used to customize the appearance of a row of Brooklyn Brownstone style homes. Customizing the appearance takes place in the top level of the Controls node. A description of each control is as follows:

- **Building Width** - This controls the overall width of each individual building. More windows will automatically be generated if the space between the edge of the building and the first window is bigger than the space between any two windows.
- **Building Depth** - Controls the overall depth of the block of buildings.
- **Size of Window / Door** - Controls both the width of the windows as well as the width of the front door. Windows will be removed if the width of the windows grows too large. On the entrance floor, the front door will always remain on the right side of the building (if facing the building).
- **Window Spacing** - Controls the amount of space between windows. If the space becomes too large, windows will be removed automatically.
- **Thickness of Modillions** - Controls the width of the modillions located underneath the roof overhang.

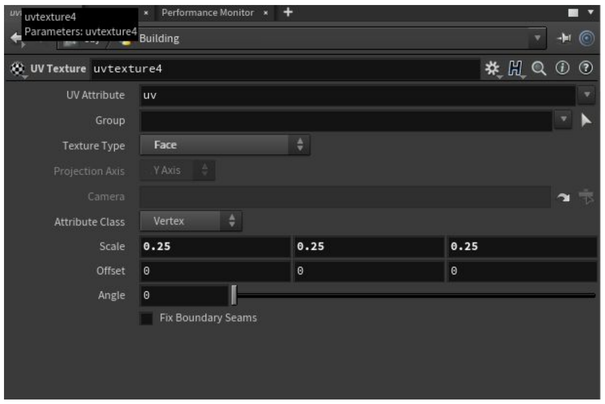
- **Modillion Spacing** - Operates similar to the window spacing. Controls the space between any two modillions. Like windows, more or less modillions will be created if the space grows too large or too small.
- **Pediment Height** - Controls the height of the pediment located over the front door.
- **Pediment Depth** - Controls how much overhang is created by the pediment located over the front door.
- **Pediment Offset** - Controls how much the pediment will overhang the sides of the door frame. An offset value of 0 results in the pediment being flush with the sides of the doorframe.
- **Pediment Thickness** - Controls how thick the pediment is. The cross piece above the front door will scale proportionally to the thickness of the pediment above it.
- **Number of Upper Floors** - Controls how many identical “Upper Floors” are generated above the entry floor. Ground floor and entry floor are required for each building. A value of 0 in this parameter results in a ground floor with staircase leading up to the entry floor.
- **Number of Houses** - Controls how many identical houses are generated. Houses will be generated to the right (if facing the building) of the previous building.

Technical Guide

- Geometry for the building was broken down into separate containers and object merged into the Building node. This allowed for cleaner networks and easier debugging when it came to piecing the buildings together.
- Placement of most windows, doors, modillions, steps, railings were done through HScript in copy nodes.
    - When it came to the pieces of the roof, bbox expressions were helpful in ensuring the roof was always placed on the top of the building.
- The ground floor and entrance floor are built the same way, with the door frame geometry being replaced with the staircase geometry on the ground floor.
    - Delete nodes were used to replace the first window iteration with the geometry for the door or staircase.

Beyond the Requirements

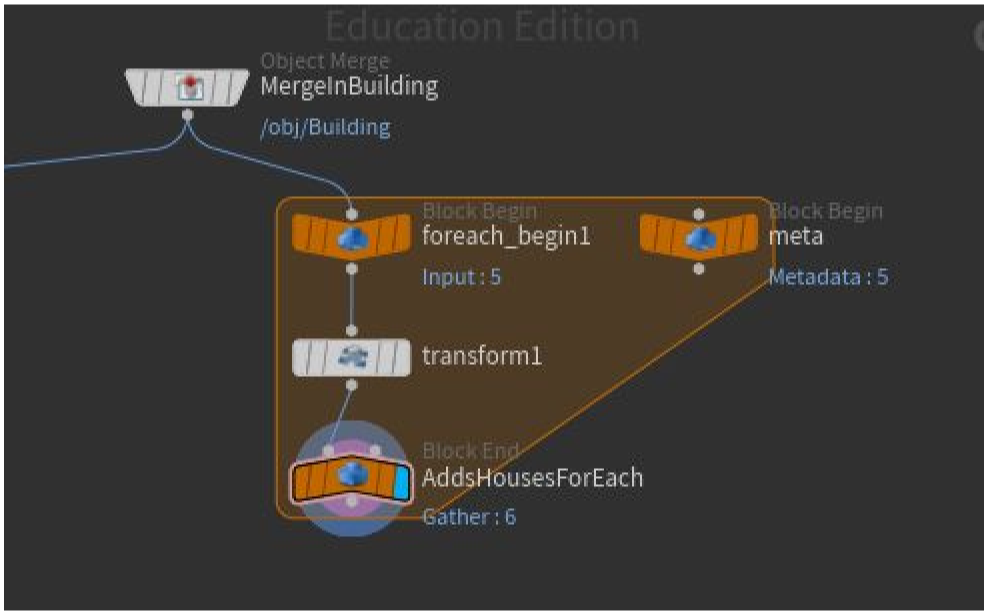
- I created tileable textures from scratch to closely match the reference images.
    - UV texturing was relatively simple due to the majority of Brownstone style buildings having the same texture throughout the facade of the building. A seamless texture could be used and applied to the faces of the geometry. Using copy nodes and simple geometric boxes kept UV's simple and clean.



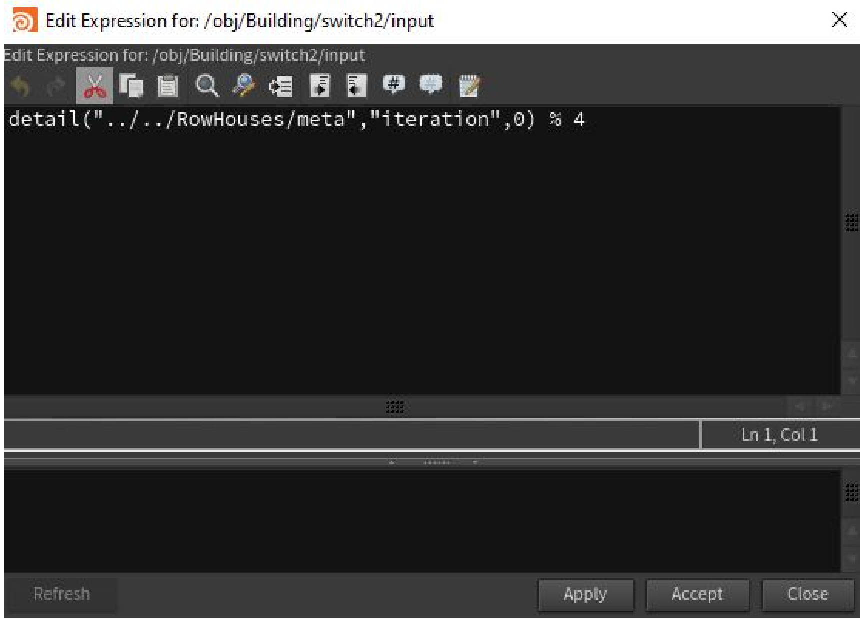
UV Texture uvtexture4

```
UV Attribute     uv
Group
Texture Type     Face
Projection Axis  Y Axis
Camera
Attribute Class  Vertex
Scale            0.25    0.25    0.25
Offset           0       0       0
Angle            0
[ ] Fix Boundary Seams
```

- In order to create the changing colors and geometry I used a foreach loop inside of my node that replicates the houses.
    - The loop contains the meta node that keeps track of how many iterations of the building we have been through. It is important because that information can be then accessed by other nodes.

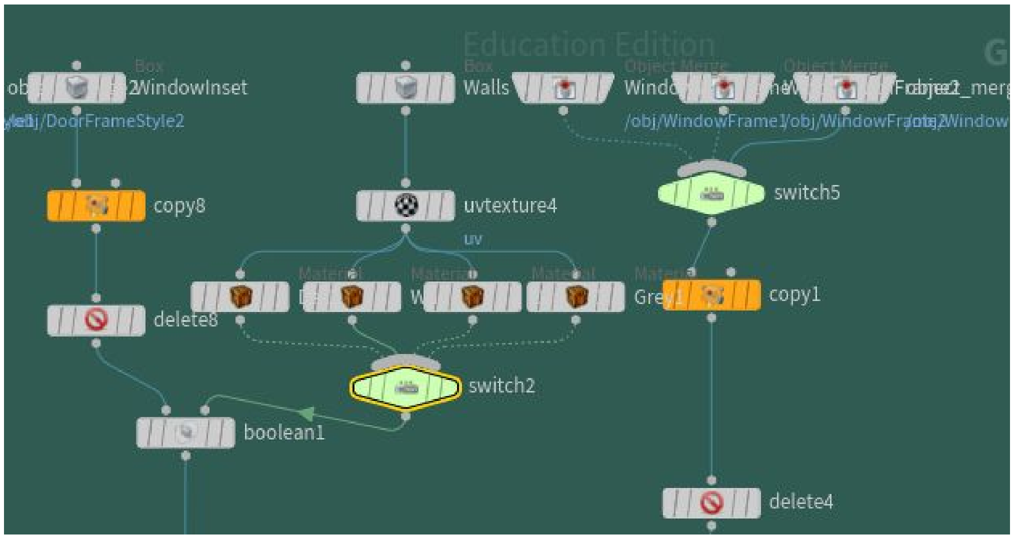


- Within my building node where I construct the building, I have my Wall geometry hooked up to 4 separate wall materials. They are fed into a Switch node that evaluates an expression to decide which of the 4 materials the wall will use.
    - Using modulus is helpful because we can create a loop. For example, the first iteration is evaluated as 0 % 4 which gives us a value of 0, therefore our default material is used. The next iteration gives us 1 % 4 which is 1 so the 2nd material is used. When we get to 4 % 4 we get a value of 0 and our pattern repeats.


Edit Expression for: /obj/Building/switch2/input
Edit Expression for: /obj/Building/switch2/input
Ln 1, Col 1
Refresh
Apply
Accept
Close

```
detail("../../RowHouses/meta","iteration",0) % 4
```

- This technique can also be applied to geometry. I used the same setup to change the doorway styles as well as the window frame styles.


Education Edition
2WindowInset
/obj/DoorFrameStyle2
Walls
/obj/WindowFrame1
/obj/WindowFrame2
switch5
copy8
uvtexture4
uv
copy1
delete8
switch2
boolean1
delete4