Final Project: Procedural City

Ryan Tsai, Ali Hafez
Github Repository: https://github.com/earth418/cs479-final
Slides from Progress Update: CPSC 479 Final Project Update

Quick Notes when running the project:

1. Make sure your .blend file is in the same directory as “tree.png” and “bird.fbx”
2. Made sure the Blender official plugin “Import Image as Plane” is enabled

Tips when using the plugin:

- be VERY careful of changing the settings, especially the tree density, scale, and number of birds – the amount of time it takes will totally blow up
- The “building area ratio”, if turned up to vary >= 1.0, can allow for buildings to be connected to neighboring buildings – if you want that kind of look, go for it![1]
- DO NOT USE THE SLIDERS, JUST TYPE NUMBERS IN (EVEN FOR COLORS)
- The brick ground does not look that good

[1] I really wish I’d been able to calculate neighboring buildings and then make their colors the same, or even merge their geometries, to create diversity in building shape, but alas, no time. :(Maybe later!

Goal

For our final project, we wrote a blender script that procedurally generates terrains, buildings, clouds, and animated bird flocks. Rather than an even grid of skyscrapers like is usually seen, we wanted the terrain and city to have a more naturalistic / medieval city appearance. A variety of cloud shapes and styles can be specified in order to replicate different types of weather— this, in addition to procedurally generated bird flocks, is meant to enhance the theme of interaction and integration with nature.

Our script does not change camera, lighting, or rendering settings. For the best results, we recommend Cycles rendering with a sky-blue background and ample lighting.

Method 1: Fractal Terrain (Ali)

I have a lot of experience with fractal terrains from projects I did in the past, and so this part didn't really involve as much new research, moreso tweaking to fit the parameters of our project. For example, I wanted a rolling-hills sort of appearance, smoother without many tiny bumps, so I decreased the persistence[2] of consecutive octaves. Because I didn't want buildings to form on cliffs, I used our terrain function and finite differentiation to calculate a gradient, and used that gradient to determine where buildings should be placed (flatter terrain), as well as trees and texture colors.

Method 2: Buildings (Ali)

This gave me a lot of trouble. I originally planned to create buildings that are each the shape of a Voronoi cell, which can be used to create central courtyards, pathways, et cetera, but figuring out how to get the individual buildings as independent geometry was increasingly difficult. Zhaoyang found this python implementation of the Delaunay Triangulation, which is the dual of a Voronoi diagram, and creates convex triangulations for a given point field. I simply passed in the centers of all the Voronoi cells over the terrain, and it created triangular "buildings" that connected each one in a way that would create geometry, that I then swept upwards to give each one a height. Originally my plan was to add a "roof" structure (with varying shapes) on top of that, but I didn't have time for that – these sweeps and structures from the triangle shapes were very similar to our midterm project, though. And having each building as its own geometry is very powerful – I was able to set them all to be separate colors, to scale them up independently, and I had the ability to modify their geometry. Because we ended up with buildings sort of independent of each other in the scene, we end

[2] Alain Fournier, Don Fussell, and Loren Carpenter. 1982. Computer rendering of stochastic models. Commun. ACM 25, 6 (June 1982), 371–384. https://doi.org/10.1145/358523.358553

up with more of a village look than a city look – though I think that's totally fine, and you can also recreate a city look if you choose to, by making the land flatter!

Method 3: Cloud Generation (Ryan)
Clouds tend to have overall shapes that are more well-suited to implicit surface representations rather than closed-form expressions. Because of this, I decided to generate cloud shapes using Blender metaballs (essentially creating a soft object).[3, 4] These metaballs are converted into a mesh, which is then subsequently converted into a volumetric representation that can be rendered by density (as discussed in Ken Perlin's *Hypertexture* paper).[5] After applying some modifiers to add wispiness and modify its optical properties, the clouds are complete.

Method 4: Boids (Ryan)
Model: https://sketchfab.com/3d-models/birds-3a9bb97be78944f9bffc23fb25c2154e
Inspiration: How do Boids Work? A Flocking Simulation

In order to model birds, I took an object-oriented approach and largely followed the algorithm proposed by Reynolds.[6] Each bird is given its own position, velocity, list of potential flockmates, and think timer. While the former two variables are discussed in Reynolds' paper, the latter two were added in order to increase computational efficiency. Rather than computing a precise list of flockmates at each frame, each bird gets an approximate list of flockmates every fifth frame, and performs exact computations on the smaller list at each frame instead.

To calculate the behavior of each bird, I implemented two methods: weighted average and priority arbitration. These methods calculate the acceleration of each bird and update the velocity accordingly, both producing different but visually feasible results. Acceleration is split into three categories: collision avoidance (which is further split into bird, obstacle, and border avoidance), velocity matching, and flock centering / cohesion.

---

[3] James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. ACM Trans. Graph. 1, 3 (July 1982), 235–256. https://doi.org/10.1145/357306.357310.
[4] Wyvill, G., McPheeters, C. & Wyvill, B. Data structure for soft objects. *The Visual Computer* **2**, 227–234 (1986). https://doi.org/10.1007/BF01900346.
[5] K. Perlin and E. M. Hoffert. 1989. Hypertexture. SIGGRAPH Comput. Graph. 23, 3 (July 1989), 253–262. https://doi.org/10.1145/74334.74359.
[6] Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. 21, 4 (July 1987), 25–34. https://doi.org/10.1145/37402.37406

Other Methods (not the focus, but still used)

- Solid textures (checkerboard/brick texture on the buildings)[7]
    - Uses the z and x/y coordinates to vary bricks over height and width, and z coordinate to just vary the checkerboard over the height
- Impostors (Trees)[8]
    - Just generated at every grid cell and then offset by a seeded random float, and then a tree is placed there

Conclusion

We are very satisfied with the results of our project! The trees, multicolored buildings, terrain, clouds, and trees combine to form very satisfying images. We do still have a few areas that could be improved (and may come back to in the future).

For example, our procedural clouds are static and could be improved by procedural animation. Furthermore, our procedural birds are unable to do long-term planning and there is no collision avoidance with the terrain/buildings. While we did implement spherical obstacle and border avoidance behaviors, the latter behavior only works with horizontal planes, and calculating a direction to go in to avoid the closest terrain to a boid is computationally expensive. In addition, the user does not have as much control over the flock paths as we would have liked— adding the option to include high-level behavior patterns / global goals would be an interesting idea for a follow-up work.

It would be interesting to add a few more building shapes. If we could generate square buildings, or irregular shapes with the Voronoi cells, that would be super cool, and we could get an interesting combination of buildings. Also, using more complex solid textures to add windows would be interesting, as well as adding roofs and other geometry to buildings. We also could have made special geometry for the paths between buildings, since those are easily identifiable, and made those into alleys, roads, or staircases. In addition, it would have been cool to *cut* through cities with wide boulevards, and definitely would have been possible to do! There are definitely more parameters we could have included, too. Lastly, of course, many parts of the project are slow and we would definitely want to improve efficiency given more time.

This project allowed us to really dive deeper into procedural modeling and creatively apply the concepts we learned in class. It also exposed us to many Blender

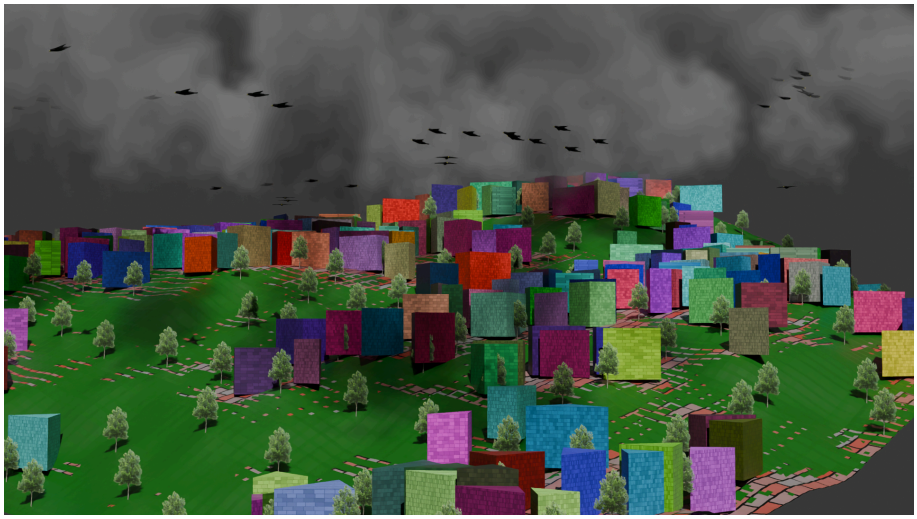
---

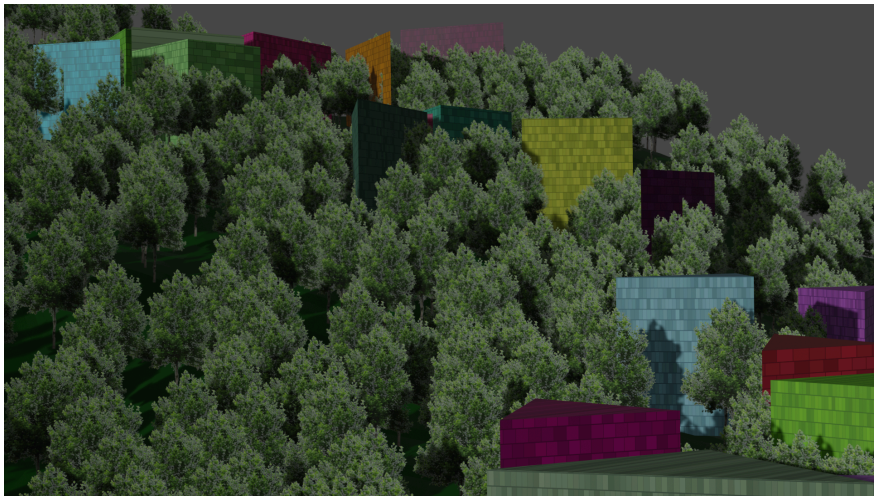
[7] Darwyn R. Peachey. 1985. Solid texturing of complex surfaces. SIGGRAPH Comput. Graph. 19, 3 (Jul. 1985), 279–286. https://doi.org/10.1145/325165.325246.

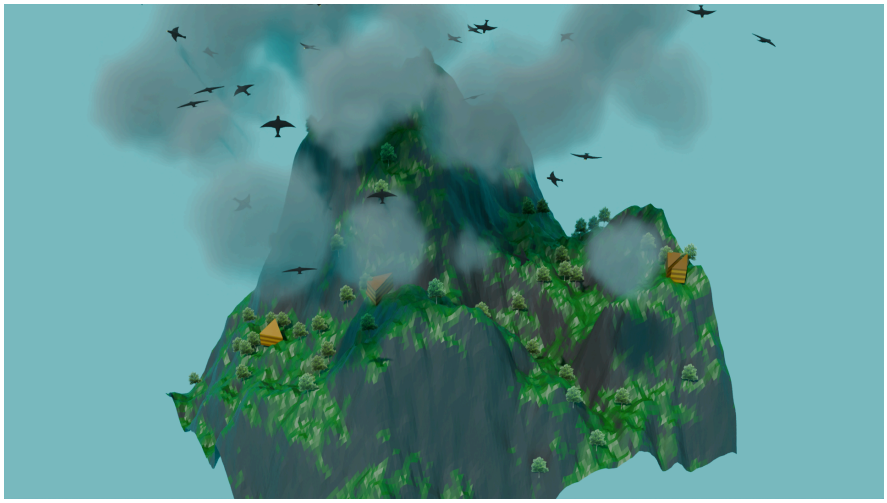
[8] EG 2007 Course on Populating Virtual Environments with Crowds. Daniel Thalmann, Carol O'Sullivan, Barbara Yersin, Jonathan Maïm and Rachel McDonnell. In 28th Annual Conference of the European Association for Computer Graphics, Eurographics 2007 - Tutorials, Prague, Czech Republic, September 3-7, 2007 (Karol Myszkowski, Vlastimil Havran, eds.). Eurographics Association. 2007.

tools that we had never seen or used before (like object constraints). In this project alone, we explored solid textures, fractals, noise basis functions, metaballs, volumetric representations, procedural agent models, and impostors. Both of us feel much more comfortable with these ideas and are excited to continue exploring them in the future!

Rendered Images







Basic Video (to show bird animation):

0001-0240.mkv