Sam Levatich

Procedurally Generated Landscapes For Sandbox Play
Advised by: Holly Rushmeier

There exist a variety of procedural methods for generating terrain[1], and the choice of algorithm is primarily dictated by the ultimate goal or use for the terrain. Examples of these goals include mirroring reality as closely as possibly, minimizing cost for maximum realism, simulating natural phenomenon, and others. As one of the places where most digital terrain design happens, game development stands to benefit a lot from a smart, modular procedure for generating different sorts of terrain. But the single most important, and hardest to measure, quality of a game is a simple question: is it fun? This is of course not something that can be reduced to a formula, but creating a sandbox world that is fun to play around in through high interactive potential can serve as an ultimate goal for my procedure, which I will design using a combination of my computational training and personal study of game design.

Previous Work:

Minecraft, one of the most wildly popular games of recent memory, is built on an entirely procedurally generated voxel world, unique for each user. I would argue that in Minecraft interactive potential of the terrain map is much less important than aesthetic pleasure, and this relates to the mechanics. Because all the terrain is breakable, and reformable, it is most important that the game presents an interesting looking world, in addition to populating the landscape with numerous caves that lead into networked tunnels of mines, also procedurally generated. The low graphics fidelity is both a stylistic choice and a practical one: it allows the game to easily be stored and take up an amount of space longer than a player could ever reasonably hope to traverse.

---

[1] See *A Survey of Methods for Terrain Modeling*, by Smelik, Kraker and Groenewegen

No Man's Sky, a more recent high profile game for high end consoles, is a game where there are 5 quintillion planets, each waiting to be generated on the fly when a player discovers them. Because an entire planet is generated from one seed, the game is an interesting exploration of ecosystem relationships: the foliage, rocky terrain, and more can all be implicitly related because they come from the same value. Similarly to Minecraft, aesthetic pleasures in these ecosystems is more important than fun, but it's even stronger in this game because the main joy of the game, as least from my perspective, is finding new planets which through their rich ecosystems create planets which feel like genuine discoveries.

Deliverables:

This will be a 3D platformer made in the Unity game engine with a 3rd person perspective viewport rotatable by the user where a bounded terrain is generated anew each play session by an algorithm tuned for interactive potential. The input, at least initially, to the algorithm will be a color map: a player will be able to draw in a bounded grid colors indicating different types/elevations of terrain which will then be interpreted and expanded upon by the algorithm. Player input during the game will control the character, whose basic options for interacting with the world will include moving in 360 of motion on the XZ plane and being able to jump upward (though this mechanic set will likely be expanded: see the next section). This means the deliverable is just the game, which includes entirely procedurally generated terrain as well as the character. The game won't have a goal state, as it is mainly designed to be an effective sandbox, but procedurally placed 'reward' elements (think coins, stars, etc.) could serve to be an optional goal for players.

Project Timeline and Ideas:

The obvious first step is to get the game world set up, which will involve figuring out a workflow in Unity, setting up a basic terrain without the user drawing input function (I plan to just hardcode a heigh map to start), and inputting character and camera control (there are free simple characters available for use within Unity). Once this setup stage is done I envision the project proceeding as testing independent ideas that I hypothesize will lead to fun emergent gameplay. One advantage of this approach is that I can quickly scrap an idea if it isn't fun without having to change significant pieces of code and I can pursue them in any order if I find a thread or two additions that would go well together. I will likely come up with more ideas as I progress the project, but here are my initial sub projects:

- **Play around with basic height map parameters**: how much noise should I add to make the game world fun to traverse, what scale should it be relative to player size (majesty and awe in game worlds create a stronger sense of immersion).

- **Platforms:** given a max jump height and max speed we can quantify the difficulty of making a jump. One way to measure fun in a game is asking how long the gamer is in a state of flow: where the game is not so easy as to be boring nor so hard as to be frustrating. By setting up a cluster of flat topped rock formations (using similar clustering algorithms for placing trees in realistic landscapes) tuned so the vertical distance is always jumpable, I can create small challenges for the player to set themselves to

- **Rewards:** in the platform example above, while the activity can be inherently enjoyable games often employ additional incentives to prod players to explore the fun and interesting parts of the game world. As I tune the procedure to create loci of interactivity I can then have the game place small collectibles at the goal points of those sub challenges (for example at the top of the highest platform rock in a cluster).

- **Overhangs + Caves:** Creating terrain that can occupy multiple points in vertical space for a given (x,z) pair is one of the more computationally challenging opportunities in this project, but both of these types of structures provide a lot more emergent gameplay opportunities because of the way they complicate vertical space, which is in many ways the most interesting axis of motion due to the jump mechanic.

- **Adding mechanics** (grappling hook / gliding / swimming / etc.). Depending on which pieces of terrain I add to the map running and jumping might not maximize the potential of certain elements of the space and extending the characters set of possible actions is one way to get more fun out of a given terrain

I hope to have that initial setup done by the first week, two features from the list above added by break, and then spend the time between break and reading period adding at least three other features (some bullet points contain multiple features) and packing the project and working on appeal while collecting playtester data.