TerraForm

Procedural Generation of Earth-like Ecosystems

By Spencer Villars and Cezar Mocan

Abstract: The goal of our project is to create a planetary simulator to explore different levels of procedural generation. Our simulator will be able to produce varied types of terrain, plants, and creatures on the fly, with each creation being unique to a simulation. The goal of this simulator will be to maximize quality and plausibility of creations, while also maintaining a suitable level of creativity and variety in generated objects. If time permits, we would also like to incorporate a playable, game-like aspect in our simulator, wherein the player can interact with our creations.

Details on Simulation: As a simulator, our project has the goal of maximizing quality and plausibility of generated worlds. This applies both to the quality of our generated structures -- how realistic is the terrain, flora, and fauna—as well as how these structures are distributed in and interact with the environment in general.

In regards to the creations themselves, we will strive to maximize quality and realism by limiting our scope to mimicking earth-like flora, fauna, and terrain. Though we will generate entirely novel environments and ecosystems, these environments will be generally earth-like in their appearances and behavior. We will not attempt to imagine entirely novel *classes* of creatures, such as those in mythology or science-fiction. Rather, we will strive to create novel examples of earth-like flora, fauna, and terrain.

In addition to having our terrain, flora, and fauna be earth-like, we would like the environment as a whole to be as realistic as possible. What this means is that there will be no forests on high mountain-peaks, nor will there be four-legged animals frequenting our oceans. Rather, we will see realistically distributed forests, grass-covered lowlands, and barren mountain-peaks. In addition, we will see animals frequenting a single environment, with different animals in high altitudes than in low altitudes.

In terms of interaction, our simulator will consist of a 1st person environment constructed in Unity3D. Controls will be standard WASD- and mouse- based, with free movement in all 3 dimensions. If time permits, we will incorporate interaction with the environment -- either as an active participant (i.e. a person interacting with animals) or as an invisible omnipotent actor (i.e. a God looking down from above).

Technical Challenges:

1. **Terrain**—The generation of a sphere surface which represents a plausible planet terrain is the first problem which needs to be solved. We foresee the following technical challenges:

    a. Finding the balance between a high level of detail and performance. The final product will likely involve on-the-fly terrain generation, as pre-generating all terrain is suboptimal, meaning that we need to test different terrain noise generators with multiple degrees of detail.

    b. Having enough variation, as well as plausibility in the generated terrain. Mapping different height ranges to geographical features (sea / ocean / lake, hill, mountain, etc.) can be done easily, but the randomly generated mesh altitudes should represent a possible flat land / water / hills / mountains configuration. Therefore, experimentation with different random noise generators will likely be necessary.

2. **Flora**—The in-game world is going to have a variety of plants populate the generated terrain, ranging from mushrooms or flowers to different types of trees. We are planning to have the flora be entirely procedurally generated, meaning that we do not want to use any assets. Another goal is for each generated instance to be unique. At this point, we are more interested in the quality of content generation, rather than a wide range of types of plants with questionable accuracy. We have identified the following challenges:

    a. Given the accuracy over variety preference, we will very likely choose to deal with the plant creation based on plant categories (e.g. a generator for trees, one for mushrooms, one for flowers, etc.) rather than a one-size-fits-all approach[1]. For example, [Runions et. al., 2007] propose a method for procedural generation of trees with a high level of possible customization regarding the output shape. Adapting their algorithm—or a similar one—to Unity3D's data structures (e.g. Tree, Branch, LeafGroup classes) will represent the first step in flora generation. Once trees are generated properly, we will use the built infrastructure to expand towards other types of plants.

    b. Performance—Both extensive on-the-fly generation and 3D rendering are resource-intensive processes, so we need to strike a balance between quantity, detail level and performance.

3. **Fauna**—Creating the animal population of our simulation is equivalent to procedural character generation. In most computer games, in-game creatures are designed in 3D modelling software, such as Blender or Maya, and then imported as assets in the game

[1] [Lehman et. al., 2016] use genetic algorithms for evolution and neural networks for selection in an attempt to create a wide range of 3D objects from . While their results are impressive, we need a higher level of detail—and most likely improved time efficiency—for the in-game world we are planning to generate.

development framework (Unity, etc.), with slight variations in the textures applied to the meshes, or in the physical properties such as height or color. For large-scale game design, this approach is very harsh on the development process: designers need to focus on producing a wide variety of secondary characters to populate the in-game world, and repetition of the same structures is bound to happen. This is the reason why procedural character generation is such a useful method (if implemented well): the diversity of the in-game fauna can reach new levels, with little leverage on the designers' part. Games such as *Spore*[2] or *No Man's Sky*[3] employ procedural generation techniques in order to place a diverse body of living creatures into their virtual universes. The literature on this subject is vast, as is the number of approaches. After a preliminary research on the topic, we are expecting the following challenges:

a. Understanding and implementing non-procedurally generated characters in Unity3D. The three main components of a character are its skeleton (or rig), mesh and mesh textures. The process of attaching a mesh to a skeleton is called *skinning* a character, and seems like a quite intricate process. Once a character's appearance is complete, the next step consists of animating it—walking, running, etc. For the purpose of this project, we are going to limit the scope of character animation to a minimum—walking.

b. Once we achieve good results with a static character, we are planning to programmatically make changes to its mesh, in order to modify its appearance, while keeping the bone structure. One option for implementing such changes are genetic algorithms.

c. Upon being able to generate random unique characters with identical skeletal structures, we will proceed to making small changes in the existing rig, by scaling certain bones or the skeleton as a whole. Programmatic changes in mesh and micro-level bone structure (as opposed to the skeleton as a whole) can already produce a wide variety of different outputs, so we will set this as the goal for this aspect of our project.

d. If time allows, making more significant programmatic changes in the skeleton (adding new bones, creating new structures within a skeleton) would increase the diversity even more. Publications on this subject explain different methods[4][5][6] of using genetic algorithms for evolving a skeletal structure to novel types of creatures.

---

[2] www.spore.com

[3] www.**no-mans-sky**.com/

[4] [Sims, 1994]

[5] [Guo et. al., 2014]

[6] [Hudson, 2013, *unpublished*]

Technical Stack:
C# and the Unity3D Engine. Potential tests done in Python or other scripting languages.

Division of Labor:
We are planning to work on (almost entirely) disjoint parts of the simulator: Spencer will tackle the world-creation (terrain, environment) and flora, while Cezar will deal with the generation of fauna. The respective challenges of these areas are discussed in a previous section of this proposal.
There will be an integration part—bringing Cezar's creatures into Spencer's world, which will require collective work, but that represents a tiny fraction of what we are planning to do.

Timeline and Deliverables:

September 19th—October 2nd
Spencer:
- Setting up a walk-able randomly-generated terrain with heights and textures
- Implementing dynamic loading of terrain and dynamic LOD
- Improving quality of generated terrain textures
- Multithreaded terrain generation
- (If enough time): Implementing extra features such as rivers and biomes

Cezar:
- Setting up the Unity environment for working with characters
- Importing a character in Unity with a predefined mesh and skeleton.
- Manually skinning the character, in order to understand the process.
- Animating the character's walking.
- Literature review of procedural mesh modification and generation.

October 3rd—October 16th
Spencer:
- Add gravity and collision to terrain generation
- Implementing flora-based ground textures
- Implementing grass and ground-flora sprites for terrain
- Add stylistic, randomly generated color palettes for terrain and flora textures and sprites
- Begin researching procedural forest and flora dispersal / generation

Cezar:
- Literature review of procedural mesh modification and generation.
- Experimenting with different algorithms for mesh modification and generation
- Creating infrastructure for generating characters with the same skeleton and different meshes.

October 17th—October 30th
Spencer:

- Research L-system tree and bush generation
- Research cloud, atmosphere,and sun procedural generation
- Add statically generated trees and bushes into world
- Begin integrating Cezar's creations into world

Cezar:

- First round of integration with Spencer's generated world. By this point, we should have the ability to automatically generate (slightly) different characters.
- Tackle gravity and collision detection of generated creatures on an uneven terrain.
- Character scaling—entire skeleton & mesh

October 31st—November 13th

Spencer:

- Implement L-system tree and bush generation
- Implement procedurally generated / distributed forests
- Implement biomes -- different trees and plants for different environments
- Begin plant / animal interaction

Cezar:

- Complete technical exploration of micro-level skeletal modifications (longer spine, shorter limbs, etc.) and integrate into existing generator → Ability to generate characters with variations in skeleton and mesh.

November 14th—November 27th

Spencer:

- Finish tree/bush procedural generation
- Implement creature wander behavior and idle behavior
- Implement gravity and collision detection into creatures
- Begin locality of creatures & fauna
- Buffer time for performance improvements: sprite & texture optimization, mesh optimization, AI optimization
- Work on integrating Cezar's systems

Cezar:

- Explore ways of generating different textures for characters—randomized colors & materials.
- At this point, our character generator should be able to generate characters with different skeletons (small variations on the same base structure), mesh shapes and textures.
- Second integration with Spencer's world.
- Resolve any technical debt or performance issues.

November 28th—December 11th

Spencer:

- Polish and Performance optimization time
- Work on locality of animals -- different animals for different regions of terrain

- Work on animal-environment interaction
- Work on animal-animal interaction
- Work on player-environment interaction

Cezar:

- If time allows, begin exploring larger scale modifications in skeleton—adding bones, genetic algorithm for generating entirely new skeletons. Literature review of bone structure generation.
- Begin implementing larger-scale skeleton modifications.
- Programmatic skinning the new skeletons with meshes—I'm expecting this to take quite a lot of time, since up to this point, all the work was done on the same skeleton model.

December 12th—End of finals

Spencer:

- Finish player-environment interaction
- Finish integrating Cezar's systems -- specifically focus on animations and behaviors
- Polish time for creatures and fauna + world interaction
- Performance optimization
- Code cleanup
- Write-up

Cezar:

- 3rd integration of Cezar's creatures into Spencer's world.
- Final polish of random creature generator.
- Fixing performance issues.
- Code cleanup.
- Finalize write-up.