Procedural Content Generation | An Island of Your Dreams

SAUL GEBHARDT and OTTO MÄTTAS, Utrecht University, The Netherlands

1 INTRODUCTION

Procedural terrain generation is used to create land forms for applications such as computer games and flight simulators. While most of the existing work has concentrated on algorithms that generate terrain without input from the user, the team explores a more controllable system that uses intelligent agents to generate terrain elevation height-maps according to designer-defined constraints. This allows the designer to create procedural terrain that has specific properties.

The team is designing a system that would procedurally generate an island with features one would expect from a basic island. Generated elements would include a coastline, beaches, mountains and rivers as the terrain features. The team is also aiming to create (parts of) different biomes with high variety and effectively seasons to make the terrain more appealing. Finally, there will be focus on the static elements of our scene - surrounding water and the sky. The lists are not exhaustive but only indicatory as the actual implementation might somewhat differ in the end.

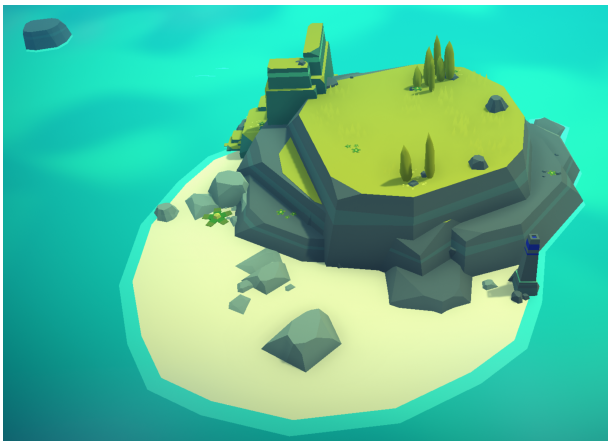

Fig. 1. Procedurally generated islands in the game ISLANDERS.

With this project, the team is focusing on the following key principles:

- Novelty: generated content contains an element of randomness and unpredictability.
- Structure: generated content is not merely random noise, but contains larger structures.
- Interest: generated content has a combination of randomness and structure that humans find engaging.
- Speed: content can be quickly generated.
- Controllable: content can be generated according to a set of natural designer-centric parameters.

Authors' address: Saul Gebhardt, s.a.gebhardt@students.uu.nl; Otto Mättas, o.mattas@students.uu.nl, Utrecht University, P.O. Box 80125, Utrecht, Utrecht, The Netherlands, 3508 TC.

2 METHODS

The team has decided to generate an island using a multi-agent approach, with the terrain being generated in stages. [Doran and Parberry 2010] The goals of this project include making the island with random input for all algorithms.

The visual art style will be low-poly. The benefits are two-fold. First of all, the clean aesthetics of the chosen art style appeal to the team. Secondly, the art style allows to use low-density height-map compared to highly detailed art styles, effectively lowering the technical barrier towards generation itself.

The implementation will be realised and visualised using a cross-platform Unity game engine. Unity serves great as a tool for visualising terrains and features, for it comes with a technical framework to cover processes like mesh creation, lighting of the scene and rendering to name a few. Also, it is fairly straight-forward to add a user-controllable camera to navigate in the final implementation to actually explore the generated terrain.

To expand on the different software packages used in addition to Unity, the team is currently flexibly leaving the options open. There are certain assets that could be leveraged through Unity but the need to do so will become blatantly clear only during implementation.

2.1 Coastline

Firstly, a coastline agent (**C**) will be developed. The agent will generate the outline of the island, effectively making space for a landmass which might be surrounded by water in the final implementation. Previously developed agents have always also immediately filled in the landmass between those coastlines, but this is something the team tries to shy away from. The coastline shall be generated according to the previously cited work [Doran and Parberry 2010, p. 5-6]

Fig. 2. Procedurally generated coastline.

2.2 Ground

Secondly, a ground agent (**G**) will be developed. The agent will not only generate ground within the landmass but also variety in it. For example, mountains and valleys. A height-map will be generated using Perlin noise. [Archer 2011] To retain the aesthetics of the low-poly art style, the team has chosen to forego more complex ground structures like overhanging cliffs.

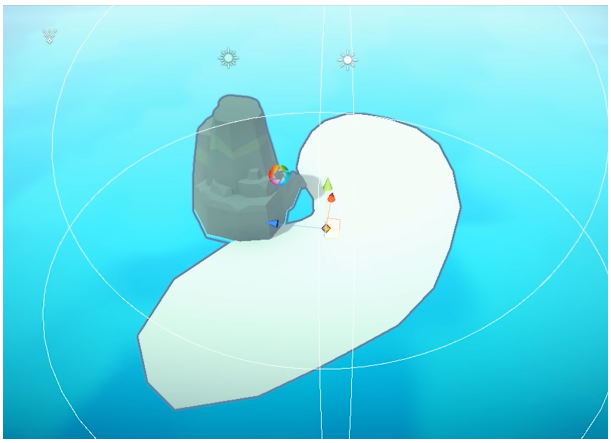

Fig. 3. A procedurally generated mountain.

2.3 Beaches

Thirdly, a beach agent (**B**) will be developed. The agent is responsible for generating beaches or flat areas on the coastline in places where beaches might naturally occur. A well-function algorithm already exists out there for generating beach agents [Doran and Parberry 2010, p.7-8], and will be modified accordingly, to allow the beach agent to generate sand beaches or rock beaches.

Fig. 4. A procedurally generated beach.

2.4 Rivers, Lakes and Oceans

Fourthly, a water agent (**W**) will be developed. The agent will generate rivers and lakes on the landmass generated by the ground agent. Also, the water agent will be responsible for handling the water flow and the ocean itself. For this, the team will look towards the following paper by Belhadj and Audibert, although some of the methods may be simplified.[Belhadj and Audibert 2005]

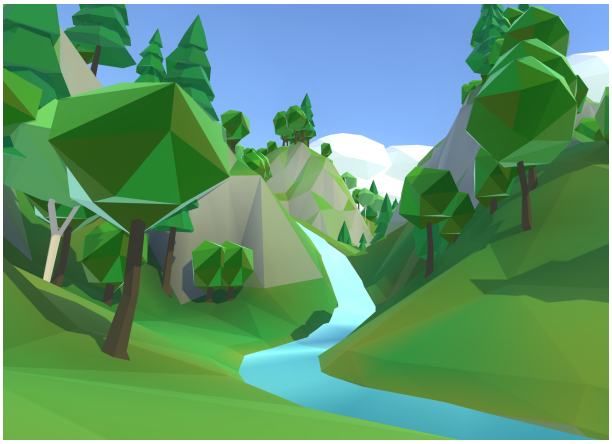

Fig. 5. A procedurally generated river.

2.5 Biome

Fifthly, a biome agent (**B**) will be developed. The agent will generate vegetation and wildlife. As soil and climate are also part of a biome, the agent should be also able to handle these. As for the scope of this project, the team will focus on the vegetation and wildlife.

There are five major types of biomes: aquatic, grassland, forest, desert, and tundra, though some of these biomes can be further divided into more specific categories, such as freshwater, marine, savanna, tropical rainforest, temperate rainforest, and taiga. The team will try to develop the agent in a way that it can be iteratively updated with certain features at a later time. This has been done pretty extensively before [Hammes 2001], which is why the team shall look at this paper extensively for inspiration.

Another aspect that may be considered for this agent is the ability to generate structures such as rocks, cliffs and reefs in the ocean.

Fig. 6. Procedurally generated grass, trees and flowers.

2.6 Urban Environments

Sixthly, an urban agent (**U**) will be developed. The agent will generate urban areas with buildings and signs of human activity. To fit the scope of the project, this agent will only handle very simple objects.

There are two ways to generate these urban environments. The first way to do it is to generate the structures themselves procedurally. This would mean that the buildings are built with some constraints in mind (maximum depth, height, how many windows, etc.). This would be very intense computationally.

The second, more basic way of generating would be to have a list of some prefabricated structures, and then place them randomly over the island. This way, there would be less randomness involved in the system, but it would be much easier to generate and ensure that there are at least some urban structures that can be generated on the island.

Given the time constraints, the team currently aims for the second method of generating urban environments. There are several reasons for this. First of all, it is much easier to implement. Second of all, it is much quicker to run. And thirdly, prefabricating certain objects can be superior to procedurally creating everything.

The team has found an extensive paper by Jess Martin, titled *Algorithmic Beauty of Buildings Methods for Procedural Building Generation*, which describes a lot of very extensive methods of generating buildings and urban environments.[Martin 2005] This paper shall be used for creating the urban agent,

Fig. 7. Prefabricated and procedurally placed buildings.

3 PARAMETERS

With the parameters, the user can influence the generation of the island. One way could be to control how many trees are on an island. This way, the user can control if the island contains a lot of life, or the island is part of a more arid climate. Another thing that could be controlled is saying that there is a maximum height that can be generated. This way, the user can control if an island is allowed to have any mountains at all.

The coastline agent will generate a coastline for the island, but with some parameters, the user can control how much the coastline curves. This would allow the user to theoretically generate a wide variety of coastlines. From coastlines that form a perfect square to coastlines that can remind you of the fjords in Norway with many nooks and crannies.

The ground agent fills up the areas in the coastlines with land. By using height-maps, mountains can be generated on the islands. The user can use parameters to control the maximum height of the island. There are many different types of islands, including low islands which have formed by reefs instead of tectonic processes and do not give any reason to have a mountain on it. If the user would like to generate such an island, this can be controlled by using said parameters. Also, the ground agent would allow us to work with valleys and possibly even caves.

The beach agent generates beaches on the coastlines, on top of the generated ground. The beaches that are generated can be either sand beaches or rock beaches. The user can choose to only generate one of the two beaches. The user can also exhibit some control over the amount of beaches that are generated.

The biome agent can be controlled by excluding certain biomes from generating, or forcing to generate a certain biome from a specific subset of all the biomes. In the scope of this project, the team is aiming to realise creating at least grass and trees, possibly even animals.

If it fits the scope of the project, the team is also interested in improving the urban agent. This opens up a lot of new opportunities for control. For example, one thing that could be controlled is the technological advancements made by the society. This could mean that houses either look more futuristic (for example, the houses being domes), or islands could house more primitive people. Then the people would be living in small encampments of tents, without real houses existing just yet.

Finally, it will also be possible for the user to choose the amount of islands they wish to generate in a given area (for example a 5km radius). While the islands may have a similar biome, they can look completely different in their terrain and may server as a home for different animals and plants.

However, it is important to keep in mind that all of these parameters are subject to change. If the team decides to implement an agent differently or decides to change the agent's behaviour, the parameters of the agent may change a bit too. It also may be possible that the user will get more control over the terrain generation, if more parameters are to be added.

If no parameters are entered, then every agent would be provided a random seed number, and will start to generate the island accordingly.

4 DISCUSSION

The project's implementation requirements are not fully defined at this stage, leaving a lot of freedom in making decisions. This is beneficial for exploratory actions while also a caveat. Namely, the team is inexperienced in the field of procedural content generation which means there is little foresight. As a result, the team is allowing for the design to change during the implementation phase. This might prove necessary in order to show a decent outcome, even if it's more simplistic than first imagined.

The team has given thought to creating an infinite world where the user is not limited to exploring only one island. In principle, the generation go on forever, where new islands are spawned as the user traverses the ocean.

The team is also interested in procedural content creation in the evolutionary paradigm. For example, it might be interesting to experiment with biomes and urban environments which are generated following the principles from genetic programming.

REFERENCES

Travis Archer. 2011. Procedurally generating terrain. In *44th annual midwest instruction and computing symposium, Duluth*. 378–393.

Fares Belhadj and Pierre Audibert. 2005. Modeling landscapes with ridges and rivers: bottom up approach. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. 447–450.

Jonathon Doran and Ian Parberry. 2010. Controlled Procedural Terrain Generation Using Software Agents. *Computational Intelligence and AI in Games, IEEE Transactions on* 2 (07 2010), 111 – 119. https://doi.org/10.1109/TCIAIG.2010.2049020

Johan Hammes. 2001. Modeling of ecosystems as a data source for real-time terrain rendering. In *Digital Earth Moving*. Springer, 98–111.

Jess Martin. 2005. Algorithmic beauty of buildings methods for procedural building generation. *Computer Science Honors Theses* (2005), 4.