Procedural Content Generation for 3D Pixel-Art Worlds

UFCFHQ-45-3 Comprehensive Creative Technologies Project Proposal
Stephen Rayment
18034264
University of the West of England

October 25, 2022

1 Description

"Procedural content generation (PCG) is the process of using an AI system to author aspects of a game that a human designer would typically be responsible for creating, from textures and natural effects to levels and quests, and even to the game rules themselves." (Smith, 2017, p.1)

This document proposes the creation of a system in the Unity Engine capable of generating and editing environments for a persistent open world. Specifically, targeting the HD2D art-style coined by Square Enix in their 2018 game Octopath Traveler (Figure 1). The proposed system would attempt to combine the Wave Function Collapse (WFC) algorithm with a number of modern PCG techniques to create a seamless open world in real time while maintaining the aesthetic that comes with a hand crafted approach. The WFC algorithm will form the basis of the system while allowing for the additional PCG algorithms to be used to augment the result to create something unique that the base WFC algorithm would not be capable of generating on its own.

Since the release of Octopath Traveler, Square Enix have continued to create games in the HD2D category. With it's increasing popularity, more companies are likely to adopt the style in the future. A tool such as the one described in this proposal could enable the developers of such games to rapidly prototype game worlds, use them as a base for further artistic polish or have content created on demand at runtime. The tool could also produce prefabs to be placed by designers or complete overlooked gaps in the game world.

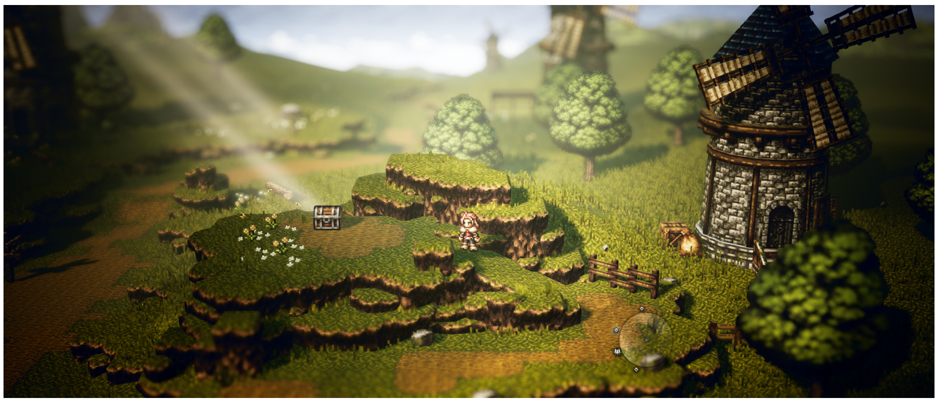

Figure 1: *Example showing the HD2D art-style and terrain composition. Enix Square, 2018*

1.1 Deliverables:

- A Unity Engine build & executable showcasing the procedural generation capabilities of the project.
- A report detailing and reviewing the final implementation.

2 Background & Research

Procedural Content Generation covers a broad category of methods and algorithms used to convert a set of input into a usable piece of content. The emphasis of this project will be on research and exploration surrounding different methods of PCG and how they might be combined in a single implementation to create the desired output.

2.1 Wave Function Collapse

Wave Function Collapse (WFC) is a texture synthesis algorithm that supports constraints (Gumin, 2016).
It works by tracking a list of each possible state that a node could evaluate to, and then removes those possibilities based on the updating constraints of the nodes surrounding it. Once the node is left with a single possibility it has "collapsed", meaning whatever value it holds is the final value. If there are no nodes left to update and the algorithm has not yet completed, then a node with the highest entropy (meaning lowest number of possibilities) is chosen and collapsed to one of its possible values. This continues the process by forcing a cascade of updates to the constraints of the surrounding nodes.

In their paper on expanding WFC for use with graph based systems (Kim et al., 2019), Kim found that WFC excelled in content control, they also found that because of its ordered approach to problem solving the algorithm could sometimes find itself in a state where a solution was impossible. Because of this some implementations of WFC require the algorithm to backtrack or restart until it is able to reach a viable solution.

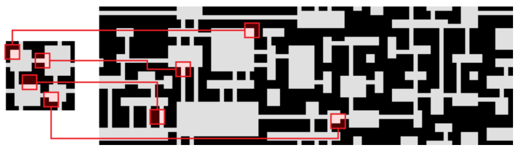

Figure 2: *Example WFC generated image with constraints generated by analysing an input image. Gumin, 2016*

2.2 Texture Synthesis

A Texture Synthesis algorithm processes a sample image and utilises that to generate a larger image based on the original. One method of utilising this is called quilting, which takes smaller samples of the input image based on constraints and then overlaps them to create the final output (Efros and Freeman, 2001).

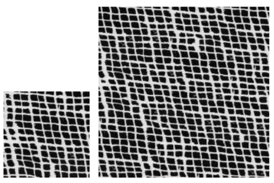

Figure 3: *Example input and output from Texture Synthesis' Quilting approach. Efros and Freeman, 2001*

2.2.1 Generative Adversarial Networks

An alternative approach to texture synthesis is to utilise a Generative Adversarial Networks (GAN). This form of deep learning utilises convolutional layers to plot data points onto a feature map. This map can then be used to generate images based on how the information has been categorised within the map (Brownlee, 2020).

2.3 Additional PCG Techniques

2.3.1 Shortest Path Algorithms

Dijkstra's algorithm (Dijkstra, 1959) and the A* pathfinding algorithm (Hart, Nilsson, and Raphael, 1968) are examples of algorithms that attempt to find the shortest path between two points. Functions like these can be used to generate long structures like rivers and paths. They can also be used to evaluate generated terrain with different sets of rules and ensure that they are traversable.

2.3.2 Cellular Automata

Cellular Automaton (CA) are a category of algorithms that act on a set of cells arranged in a grid. Each cell has a finite set of states, usually on or off, which can get changed based on the states of the cells surrounding it. CA can be used to generate natural looking 2D cave systems in games as demonstrated in an article by Michael Cook (Cook, 2013).

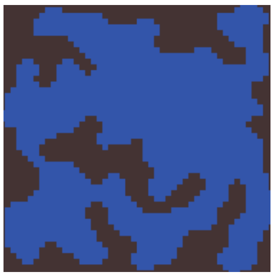

Figure 4: *Example of 2D Cave generation using Cellular Automata. Cook, 2013*

2.3.3 Random Walk

Random walk is a simple algorithm that can be used as an end of generation step to add randomness into the content. This can often help to hide or obscure unnatural looking patterns. The paper by Doran., et al (Doran and Parberry, 2010), shows an example of this by utilizing random walks to apply smoothing to terrain after it has been generated.

2.3.4 L-Systems

A Lindenmayer system, more commonly known as an L-System, is grammar based algorithm that can be used to generate natural looking plant like structures. (Lindenmayer, 1968). L-Systems could potentially be used for road or texture generation.

2.3.5 Voronoi Diagram

A Voronoi diagram is a way of partitioning a plane into convex polygons. It works by distributing points, usually randomly, across a plain and then uses those points to divide the space. This creates a visually distinctive pattern (Figure 5).

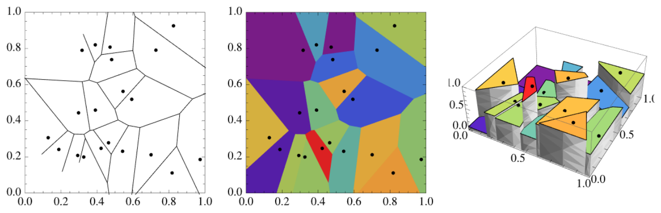

Figure 5: *Example of a generated Veronoi diagram. Martin, 2020*

3 Objectives

3.1 Project Objectives

- Combine WFC with multiple PCG techniques to create unique and novel content for HD2D environments.
- Create a generation tool for the Unity Engine capable of creating environments in the desired art-style.
- Create a mesh import tool capable of categorising and preparing meshes for use with the WFC algorithm.
- Create a texturing solution to apply materials to the generated environment in the desired art-style.
- Create tools to procedurally dress the environment with foliage and static objects.

3.2 Research Objectives

- Further research into WFC and how it can be integrated with more complex constraint systems.
- Further research into path drawing algorithms and solutions.
- Research into mesh generation, combination and deformation.
- Further research into Triplanar Projection for texturing and adding additional plains.
- Research into procedural distribution models for foliage and prefab placement.

3.3 Learning Objectives

- Learn 3-5 PCG techniques that can work in tandem or iteratively to create varied environments.
- Learn how to optimize runtime mesh creation and editing.
- Learn how to allocate and blend textures based on area context.
- Learn procedural distribution techniques for natural looking foliage.

4 Methods & Techniques

- ***Project Management.***
  A number of tools and techniques will be used to manage the project. These include Trello and an Agile workflow, Git and GitHub for version control, and then Unity Engine and the Rider IDE for development.

- ***Tileset Importer.***
  The first step of the project is to create a test set of tiles and an automated method to import them for use with the WFC algorithm. This setup can initially be done by hand however the complexity of doing so grows exponentially with the size of the tileset. As the tileset will be evolving with the project, this task will save time by completing it first.

- ***Alpha Generation.***
  Once the test tileset is ready, the next task will be to get a basic version of the WFC algorithm functional in Unity. This will be a chance to to test the tileset and take note of any constraints that might have been overlooked during tileset creation and the import process.

- ***Augmenting WFC Algorithm.***
  Once basic generation is functional the focus will shift to augmenting the WFC Algorithm to work with additional constraints. This will start with a manual inclusion of constraints with adjustments to the algorithm to work with previously generated or partially complete terrain.

  – The WFC algorithm is implemented on a per case basis and tailored to its intended use (Sandhu, Chen, and McCoy, 2019). Once implemented it should be expected to be changed multiple times to implement additional constraints. Therefore the implementation will adhere to the S.O.L.I.D design principles (Martin, 2000), mainly the Open/Closed principle. This simply means that the initial implementation will not change however it will be open to extension through other means.

- ***Triplanar Mapping.***
  Adjusting the uv's for each tile can take up allot of time. Triplanar Mapping (TM) allows for projecting a texture onto the surface of a model based on the direction of each of its faces. This may not be the end solution however the concepts from TM can be expanded upon and will greatly increase the visual appeal of

the project at an early stage in development.

- ***Additional PCG Techniques.***
  Once the system is capable of generating basic textured environments additional PCG techniques will be employed to create more specific environments. For the purpose of customisation these are likely to be split into ordered generation steps that layer and augment the previous steps. These steps will either act as a pre-generation stage where a final set of constraints are submitted to the WFC generator, or each step will cause the WFC generator to iterate create parts of the terrain based on context.

- ***Tooling.***
  At this point it is important to create tooling that allows for data collection that can be used to debug and review the different iterations. Doing this at this stage ensures that a substantial backlog of information is accessible in the end stages of the project.

5 Risks & Issues

Risk	*Mitigation*	*Contingency*
Tileset Importer becomes incompatible with an updated tileset.	Create tilesets within the limitations of the importer.	Set the constrains of each tile manually using the Unity Editor.
WFC algorithm is too complex or incompatible with the desired constraints.	Start with simple implementation and then scale up the complexity.	Use additional PCG techniques to augment the result.
Path and River generation is too chaotic and does not look natural.	Carry out additional research into path drawing algorithms.	Use prefabs or partially completed chunks with completed rivers/paths and have WFC generate the rest.
Editor tooling takes up too much time and halts progress.	Keep tooling simple, focus on practicality and put low priority on aesthetics.	Replace tooling with simplified scripts that can be run through the editor context menu.

5.1 Specialist Resources

No specialist resources are required for this proposal.

6 Monthly Project Plan

Month	*Task*	*Days*
November	Create test tileset.	1
	Create importer script to automate the importing and setup of tilesets.	3
	Implement the Wave Function Collapse Algorithm.	3
	Adjust WFC algorithm to include additional constraints.	7
	Create shader based on triplanar mapping.	2
		16
December	Write up the research report.	3
	Research Submission - 13/12/22	1
	Make adjustments and improvements to tileset.	2
	Implement Path and River generation	4
	Implement Prefab & Foliage distribution.	5
		15
January	Implement runtime editor tooling.	5
	Implement partial completion and regeneration of edited areas.	5
	Create varied biomes based on constrain presets.	4
	Create PCG system for creating new biome presets.	4
	Presentation preparation.	3
	Assessed Presentation - w/c 23/01/2023	1
		15
February	Make adjustments and improvements to tileset.	2
	Implement blending rules for generated areas with different biomes.	6
	Implement mesh generation and combination for optimization.	3
	Implement mesh height deformation for more natural looking surfaces.	4
		15
March	Implement procedural texture generation	7
	Implement texture blending based on chunk and biome map.	4
	Polish visual aspects for presentation	3
		14
April	Profile and gather data for the final report.	3
	Write first draft of the final Report.	3
	Edit and finalize Report.	7
	Final Submission - 25/04/23	1
		13
May	**Viva - date to be announced**	1

7 Bibliography

Cook, Michael (2013). Generate Random Cave Levels Using Cellular Automata.
Available from: https://gamedevelopment.tutsplus.com/tutorials/generate-random-cave-levels-using-cellular-automata–gamedev-9664 [Accessed 25 October 2022].

Dijkstra, Edsger Wybe (1959). A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271). Available from: http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf [Accessed 25 October 2022].

Doran, Jonathon and Ian Parberry (2010). Controlled procedural terrain generation using software agents. In: IEEE Transactions on Computational Intelligence and AI in Games 2.2, pp. 111–119.
[Accessed 25 October 2022].

Efros, Alexei A. and William T. Freeman (2001). Image Quilting for Texture Synthesis and Transfer. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 341–346.

Available from: https://doi.org/10.1145/383259.383296
[Accessed 25 October 2022].

Enix Square (2018). Octopath Traveler. [Video Game]. Square Enix.

Gumin, Maxim (2016). Tilemap generation from a single example with the help of ideas from Quantum Mechanics.
Available From: https://github.com/mxgmn/WaveFunctionCollapse
[Accessed 25 October 2022].

Hart, Peter E, Nils J Nilsson, and Bertram Raphael (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4.2, pp. 100–107.
Available from:
[Accessed 25 October 2022].

Kim, Hwanhee et al. (2019). Automatic generation of game content using a graph-based wave function collapse algorithm. IEEE Conference on Games (CoG). IEEE, pp. 1–4.
Available from: https://ieee-cog.org/2019/papers/paper_187.pdf
[Accessed 25 October 2022].

Lindenmayer, Aristid (Jan. 1968). Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. Journal of Theoretical Biology 18.3, pp. 300–315.
Available from: https://www.sciencedirect.com/science/article/abs/pii/0022519368900799
[Accessed 25 October 2022].

Martin, Robert C (2000). Design principles and design patterns. Object Mentor 1.34, p. 597.
Available from: http://staff.cs.utu.fi/staff/jouni.smed/doos_06/material/DesignPrinciplesAndPatterns.pdf
[Accessed 25 October 2022].

Weisstein, Eric W. Voronoi Diagram. MathWorld–A Wolfram Web Resource.
Available from: https://mathworld.wolfram.com/VoronoiDiagram.html
[Accessed 25 October 2022].

Sandhu, Arunpreet, Zeyuan Chen, and Joshua McCoy (2019). Enhancing wave function collapse with design-level constraints. Proceedings of the 14th International Conference on the Foundations of Digital Games, pp. 1–9.
Available from: https://dl.acm.org/doi/pdf/10.1145/3337722.3337752
[Accessed 25 October 2022].

Smith, Gillian (2017). Procedural content generation: An overview. Level Design Processes and Experiences, p. 1.
Available from: http://www.gameaipro.com/GameAIPro2/GameAIPro2_Chapter40_Procedural_Content_Generation_An_Overv
[Accessed 25 October 2022].

Brownlee, Jason (2020). How Do Convolutional Layers Work in Deep Learning Neural Networks?, p. 1.
Available from: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
[Accessed 25 October 2022].