6CCS3PRJ Final Year Individual Project Report Title

Final Project Report

Author: Zishan Rahman

Supervisor: Senir Dinar

Student ID: 20071291

April 16, 2023

Abstract

Procedural generation refers to content in a medium that is produced algorithmically in lieu of by hand. Most notably, procedural generation algorithms are implemented in video games, for generating levels, terrain and other game contents programmatically. This project takes some of the more prominent algorithms for procedural generation- Lindenmayer Systems, Voronoi Points, Poisson Disk Generation and Simplex Noise- and implements them in a 2D tile-map-oriented RPG-like game in the open-source Godot game engine, and compares their workings and performance. My aim with this project is to (1) increase my knowledge of procedural generation in games beyond the surface level, by going in-depth into some of the algorithms that are used, and (2) use this knowledge to implement said algorithms in a 2D tiled RPG scenario in Godot, then compare how each algorithm works and performs.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Zishan Rahman

April 16, 2023

Acknowledgements

Thanks to my supervisor Senir Dinar, for providing me the guidance I so badly needed to make this project not only the best for my mark, but also one that I enjoy.

Thanks also to Kevin Lano, who helped me up when I was down and was able to offer as much additional feedback as he could.

Thanks, also, to Shaan Vieru, and to whom I dedicate this project. You were so polite and humble, fun to be around and I'm sure you would've loved to see the final product. Shine on, you crazy diamond, and fly high among the stars.

Contents

Chapter 1

Introduction

Procedural Content Generation, or PCG, refers to the use of algorithms and programming in lieu of human handiwork to design and implement various contents in video games, such as levels, terrains, trees and cities. A PCG algorithm is ontogenetic when it tries to produce a foreseeable end result as it goes along. For this project, I will be implementing several well-known ontogenetic algorithms in a basic 2D tile-map-oriented RPG-like game, using the open-source Godot game engine, and then comparing how each algorithm carries out the creation of levels in said game, both performance-wise and comparing the kinds of level layouts generated by each algorithm.

1.1 Report Structure

Chapter 2

Background

For my BSc individual project, I will be researching procedural content generation (PCG) algorithms and then implementing them each in a small 3D game made with the Godot Engine (and its domain-specific GDScript language).

2.1 Procedural Generation: Background

Procedural content generation (usually referred to as simply "procedural generation") refers to the creation of levels and other game objects programmatically and algorithmically, in lieu of a human being doing all the work. While procedural generation algorithms can be used to generate a myriad of things, from textures (for things like trees and clouds) to music ("generative music," as coined by legendary musician Brian Eno), by far its most common context is in automated level design, generating level layouts algorithmically in lieu of work from level designers. Game developers may opt to use procedural generation to save time and money designing levels or show off technical prowess in their games.

Procedural generation in video games has a rich history. Pioneering games such as Rogue (1980) took direct influence from tabletop role-playing games such as Dungeons and Dragons, and thus had a player navigate a randomly-generated world that expanded further as they went on. Such games spawned the *roguelike* and *roguelite* genres, which experienced immense popularity in the last decade. In the realm of first-person shooters, 2004's .kkrieger, as seen in Figure 2.1, used procedural generation to create intricate 3D levels and fit them all into a game that takes up just 96 kilobytes of space.

Figure 2.1: The game .kkrieger, which uses procedural generation to design maps while keeping the game at a 96 kilobyte file size.[8]

Other games that use procedural generation in its levels include Elite (originally published in 1984), Elite: Dangerous (2012), Minecraft (2009), No Man's Sky (2012) and Spelunky (2013). The latter game's use of procedural generation has notably been covered by video games journalist Mark Brown in a YouTube video.

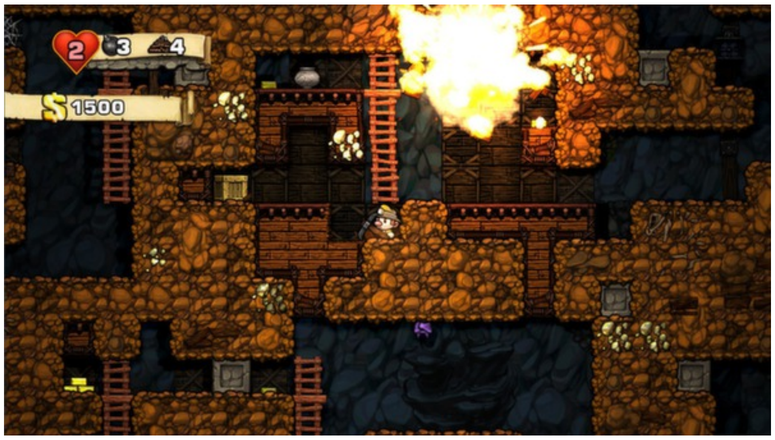


Figure 2.2: The roguelike game Spelunky, which uses procedural generation to build intricate levels for the player character to explore.
Source: https://store.steampowered.com/app/239350/Spelunky/

In many cases, these games end up having a **large** number of different environments that each game could generate for its players. However, by procedurally generating them upon the *loading* of the game level, in lieu of loading a layout from disk, they can save a lot of space (albeit with a considerable need for processing power, depending on the game's and algorithms' performance), as seen in Figure 2.1.

Using one or some different procedural generation algorithms, such as the use of Perlin, Simplex or other noise, Voronoï disks and also poisson disk generation, among others, games can load a seed to randomly generate a level every time it is played, meaning no two playthroughs of a game with procedurally generated content are ever the same.

2.2 Justifying My Choice of Engine: Godot

While a myriad of resources exist for procedurally generated game contents exist for Unity and Unreal, I want to implement them in Godot, for several reasons:

- It's the engine I have the most experience with, having already developed 2 published web games with it.
- It's not got as many resources on procedural generation compared to Unity, Unreal and some other popular game engines, particularly on the side of academic research (that is, there aren't as many papers on procedural generation that pertain to Godot as they do to Unity, Unreal and other engines).
  - However, it is still very powerful and feature-rich (it has its own Open Simplex noise class, for example) and I'm sure I can make procedural generation algorithms work on it.
- Compared to Unity and Unreal, Godot is a very light engine with a feature-rich editor, clocking in at under 100MB, with editors for Windows, macOS, Linux and even the web browser.

By the end of my allotted time, I plan to have implemented several procedurally generated environments in small Godot games, using a myriad of methods (such as Voronoï cells and poisson disk generation) in a myriad of contexts (anything from platformers to first-person games). With these games, I plan for the final report to be the centrepiece of my project, with it containing my research on how each environment was implemented, as well as my findings on the algorithms themselves and how they work.

This is somewhere between a research-oriented project and an implementation-oriented project, as while the produced software artifacts provide valid proof of my understanding of some commonly used procedural generation algorithms and how to implement them in Godot, it is also about how I understand their workings. Nonetheless, the implementations provide the weight behind my project's motivations and are the main focus of this dissertation. They will prove that Godot is just as adept at procedural content generation as the other major players in the game engine space, and I will have gained a wealth of knowledge on PCG in the process.

2.2.1 Note on Differing Versions of Godot

Godot currently is at version 4, which finally received a stable release in 4th March after years of development, but concurrently there is also Godot 3, the previous stable version which is now a **L**ong-**T**erm **S**upport release. The latter version of Godot contains several new features and breaking changes, so any project made in Godot 3 won't readily be compatible with Godot 4 (and vice-versa) without making the necessary changes and conversions. I have access to both versions of Godot and, for all the Godot projects I made and used in this project, I have used Godot 4. Any references to other Godot 3 projects will be clearly denoted as such.

2.3 Justifying My Choice of Scenario: A 2D tile-map RPG-style roaming game

The scenario of my choosing involves a monochrome tile-map created by Kenney.nl in a 2D RPG setting, in which the player character is a hollow "Golem" that is trying to search for and obtain a ring among a large 72x40 village, filled with trees, buildings and emptiness. The player can "chow down" trees by simply going to the cells where trees are and making them disappear. However, the player *will* stop at and collide with any buildings in the tile map. When the player collects the ring, they win the game and are able to either close the window or generate a new village to try and collect *another* ring.

The size of the tile map is determined by taking the window size, 1152x640 in **all** implementations, and then dividing it with the cell size, 16x16 in **all** implementations (again), hence returning a 72x40 tile map size. Using a large tile map like this, with 2880 available cells in total, allows for easy stress-testing of the algorithms, making them generate level layouts that are sufficiently large enough to produce a quantifiable performance result and time that can be easily compared across implementations, such that we can easily measure how one performs over the other. The use of a tile map *this* large with PCG algorithms also makes sense from a game developer's perspective as designing level layouts this large by hand, with such a small cell size as well (inherited from the size of the tile map assets), would add additional time and labour costs to them.

The use of a tiled role-playing game scenario, adapted to already-existing procedural generation algorithms, is relatively unusual in the context of procedural generation. However, it *will* allow me to go a degree beyond the scope of what is usually done for procedural content generation in games, which is usually seen in 2D and 3D roguelikes and platformers, as well as some other world-building games such as Minecraft and Terrraria, while also producing code that is relatively easy to process through and understand. The ability for the player character to consume trees and remove them from the level layout by moving into them allows that player to easily move around in what would otherwise be very crowded level layouts that would have been near-impossible to traverse. The addition of said player character, as well as the end goal of obtaining a randomly-placed ring within the given level, adds weight to the algorithms' practical use in games made with Godot, and not just for show or solely as demonstrations.

2.4 Justifying My Choice of Algorithms for the Above Scenario

For this project, I intend to use the following procedural content generation algorithms within my scenario:

1. Lindenmayer Systems (or L-Systems)
2. Perlin and Simplex Noise
3. Poisson Disk Sampling/Distribution
4. Voronoï Cells/Diagrams

Using an L-System for generating a level layout is relatively uncommon, compared to its use in generating structures such as trees and buildings. However, I plan to integrate a deterministic context-free L-System (or a "D0L-System") into an implementation of my scenario so I can compare it performance-wise to the other algorithms, and see how the repeated patterns generated from L-System grammars affect comparisons to the other implementations' level layouts.

Perlin and Simplex Noise are far more commonly used for level layouts, so I created an implementation of my scenario with one to see how it compares with the others, speed-wise and layout-wise, and see if it really is the best for my chosen scenario.

Poisson Disk Sampling is usually used for item placement in planes, even with grids, so using a grid-like implementation, I will compare how it works with in a tile map and what differences arise between its use there and in its usual uses.

Though efforts were made to make level layouts as similar as possible across implementations, there are noticeable differences between the level layouts generated by L-Systems, Simplex noise and Poisson disk samples, and I touch on those when discussing those implementations in the relevant sections of my report.

In my research and implementation of Voronoï Cells I realised the level layouts it generated for my scenario were wholly unique, when compared with the other algorithm implementations, so much so that I had to re-shape my scenario and game mechanics to make both the scenario and levels generated fit with each other. Nonetheless, I believe this will serve as a unique comparison to the other algorithms and will serve as additional knowledge of procedural generation algorithms as well as more work towards understanding how to make them work in Godot games (as proven by my implementations).

Chapter 3

Report Body

In this chapter, I will explain how each of my chosen algorithms work, and how I went around implementing them as a surface-level explanation. I will then briefly compare what challenges I faced for each of my implementations, and how they compare, both performance-wise and with regards to the kinds of layouts they produce, again as surface-level explanations. I go into greater detail on my implementations in the Implementation section, how the level layouts generated in each algorithm compare with each other in the Design & Specification section, and how each implementation compares overall (and also performance wise) in the Evaluation section. For this project, I chose to use the following 4 algorithms.

1. Lindenmayer Systems (or L-Systems)
2. Perlin/Simplex Noise
3. Poisson Disk Sampling
4. Voronoï Cells

3.1 Algorithms

In this section, I will explain how each of the algorithms I implemented work, then I will go into small detail as to how I implemented them. I go into further detail in the "Implementation" section of this report.

3.1.1 Lindenmayer Systems

Hungarian academic Aristid Lindenmayer devised a mathematical model for the reproduction of fungi in 1967.[12] His model involved a string of symbols, each unique symbol denoting a specific action and/or branch. Essentially, running that initial string, called the *axiom*, through a set of rules (called a *grammar*) gives us an ever-expanding string that is then taken as instructions to draw something from. Lindenmayer Systems, or L-Systems, have since been used in several scenarios beyond its initial purpose of modelling fungi, from trees to fractals. In video games, they are frequently used to aid in the creation of foliage in several environments, as well as buildings and, here, level layouts.

A Basic 0L-System

The most basic form of L-System is a *0L*-System, 0 in this case referring to the fact that the grammar is *context-free*.

For this example[2], consider an alphabet V, which consists of the following symbols:

$$\mathrm{F}, +, -$$

where F means "to go forward", and $+$ and $-$ denote turning right or left (respectively) a set number of degrees .

Take an axiom ω, for example:

$$\mathrm{F} + \mathrm{F} + \mathrm{F} + \mathrm{F}$$

And a set of rules P which, in this case, is of size 1:

$$\mathrm{F} \rightarrow \mathrm{F} + \mathrm{F} - \mathrm{F} - \mathrm{FF} + \mathrm{F} + \mathrm{F} - \mathrm{F}$$

We can represent this *parametric* L-system in the following form:[20]

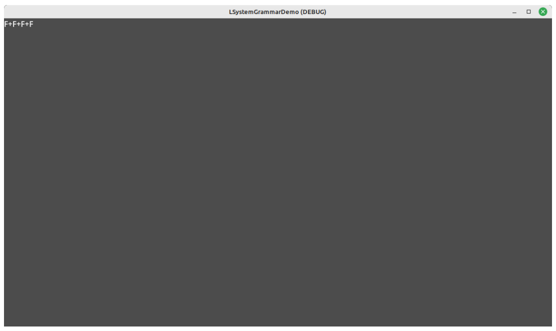
$$G = (V, \omega, P)$$

To implement G in Godot, we can take each rule and replace each string in accordance to our one rule, using the replace method, like so:

```python
string = string.replace(rule["from"], rule["to"]) #Here the rules
    were stored in dictionaries.
```

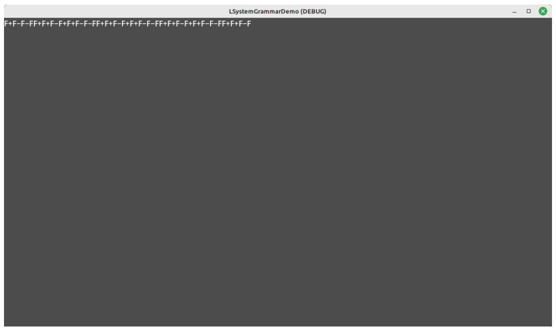
Figure 3.1: A line of code that demonstrates directly replacing characters in a string according to our L-System grammar's rules.

The first 3 iterations of this operation are shown here:



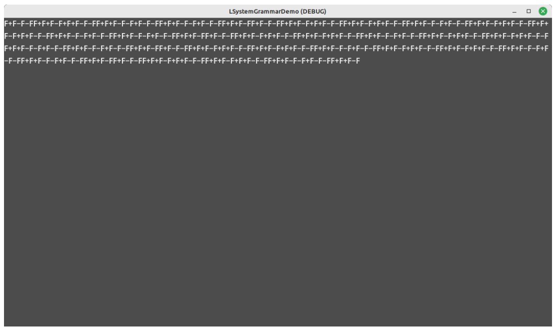
```
F+F+F+F
```

Figure 3.2: The axiom of the aforementioned simple L-System with just one rule. String size: 8. Source: Own work.



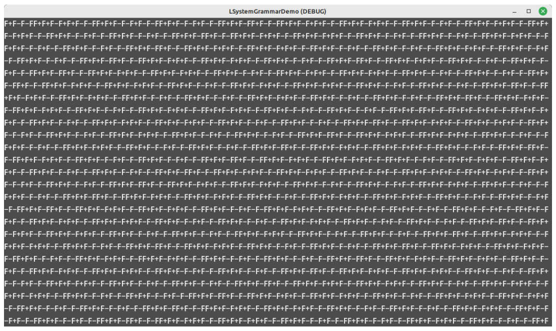
```
F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F
```

Figure 3.3: The first iteration of the aforementioned simple L-System with just one rule. String size: 59. Source: Own work.



```
F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+
F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-F
F+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F
-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F
```

Figure 3.4: The second iteration of the aforementioned simple L-System with just one rule. String size: 475. Source: Own work.



```
F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+
F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-F
F+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F
-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-
F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+
F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF
+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-
F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+
F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+
F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F
F+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-
F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+
F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+
F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F
F+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F
-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F
F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+
F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-F
F+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-
F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-
F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+
F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-F
F+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F
-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F
-F+F-F-FF+F+F-F-F+F-F-FF+F+F-FF+F-F-FF+F+F-F+F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F+F-F-F+F-F-FF+F+F-F+F+F-F-FF+F+F-F-F+F-F-FF+F
```

Figure 3.5: The third iteration of the aforementioned simple L-System with just one rule. String size: 3803. The string is too large to show in the window, as you can see here. Source: Own work.

The resulting string can be used to draw a lattice.[2] Examples of the above grammar in action are below.

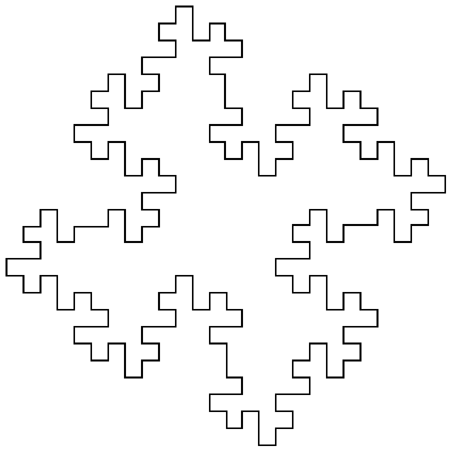


Figure 3.6: A lattice generated with the example grammar on a custom-written Classic Mac OS application specifically written for working with L-Systems.[2]

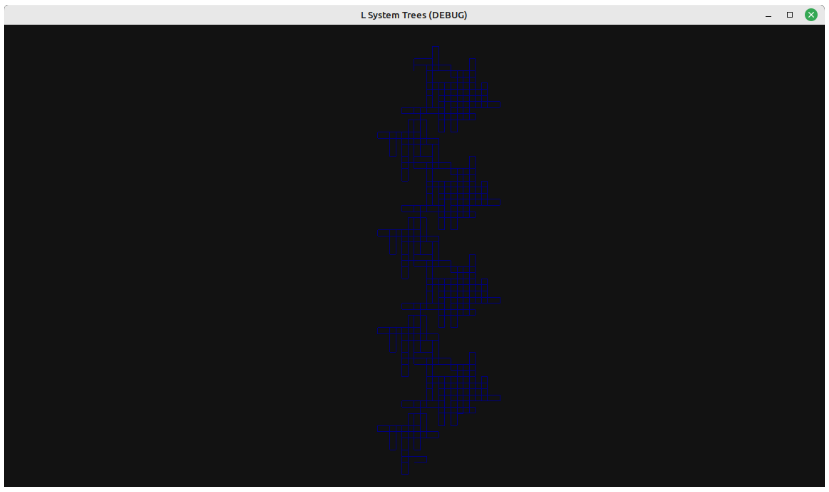


Figure 3.7: A lattice generated with the example grammar on a Godot project for drawing from L-Systems. Source: Initial project written by YouTuber Codat[3][4], and converted to Godot 4 (with the addition of the lattice grammar) by me.[5]

A More Complex D0L-System With More Than One Rule

For handling more than one rule, we can instead use a new string buffer variable where, for each character in our string, we can attain a new string and append it to our string buffer. The resulting string is then returned and interpreted. This can be represented in Godot as demonstrated in Figure 3.8, which uses two functions to perform string replacement. The first function `get_new_replacement` performs the character replacement according to the L-System's grammar rules, while the second function `replace_string` uses a string builder variable to allow for replacement of characters without directly affecting the original string and causing unwanted side effects.

```
func get_new_replacement(character: String) -> String:
    for rule in rules:
        if rule["from"] == character:
            return rule["to"]
    return character

func replace_string(string: String) -> String:
    var new_string = ""
    for character in string:
        new_string += get_new_replacement(character)
    return new_string
```

Figure 3.8: Two GDScript functions for replacing characters in an L-System grammar with more than one rule.

This can *then* be used to handle more complex grammars that can handle more than one rule in which characters in strings are replaced by other strings of variable length, as before.

The grammar in the following example represents a D0L-System[15], a **deterministic** L-System using a context-free grammar; the grammar in the first example was *also* deterministic.

For this example, consider a new grammar G with the alphabet V, where a and b are the only symbols. We start with the following axiom ω, which is just a. We now have a set of rules P which is, this time, of size *2*:

$$\text{a} \rightarrow \text{ab}$$

$$\text{b} \rightarrow \text{a}$$

The first few steps of the resulting derivation can be modelled like so:

```
        b
        |
        a
       _|_
      a   b
     _|   \
    a b    a
   _| |    |_
  a b a    a b
 _/  | |_  |_ \
a b  a a b a b a
```

Figure 3.9: The first few steps of a derivation of our example grammar.[15]

3.1.2 Perlin/Simplex Noise

Traditionally, white noise images, and most other noise types, place noise pixels completely randomly, without each pixel considering the values of its neighbours[16], as you can see in Figure 3.10.

However, there exists several types of **value** and **gradient** noise that *do* take surrounding pixel values into consideration, and will therefore serve more use in building levels in our games.

Value noise simply takes a lattice of points with random values and then interpolates those points based on their surrounding values. This *can* be used as a procedural texture. However, due to the simple nature of the algorithm, it's possible that the difference between several values in a region is minimal, while in other regions the values may differ immensely, resulting in a noise image that is not very smooth.

Gradient noise, on the other hand, takes point lattices and instead calculates the interpolation between tangents.[6] Since both tangents between a curve must be collinear[6], the flat and bumpy curves produced by value noise's interpolation calculations are now much less likely to be returned, as seen in Figure 3.11.[6] This results in noise images of higher and more appealing visual quality as, to quote a response from Stack Exchange by Hernan J. González[11], "it cuts low frequencies and emphasizes frequencies around and above the grid spacing."

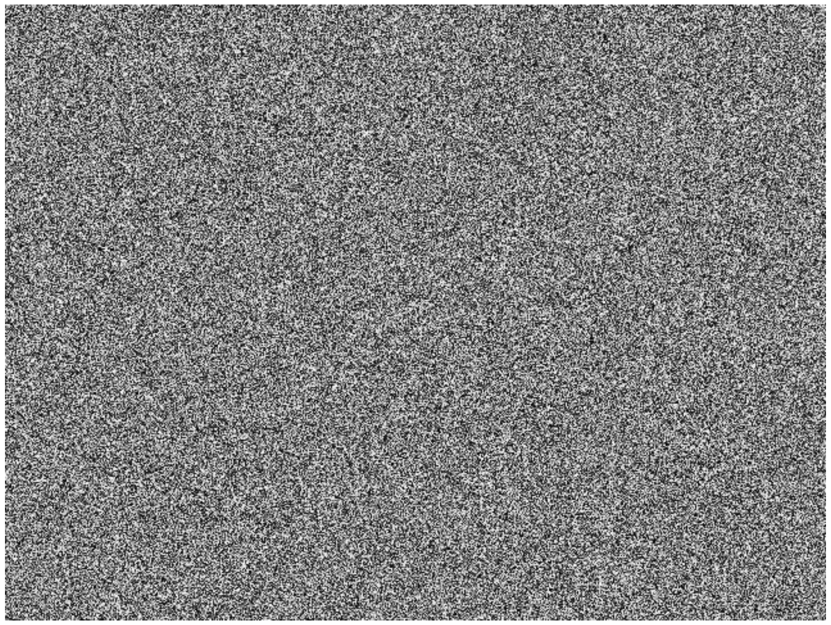


Figure 3.10: A white noise picture generated with Robson's white noise image generator.[18] Settings: 640 squares horizontally, 480 squares vertically, size of squares 1, colours greyscale, bias none.

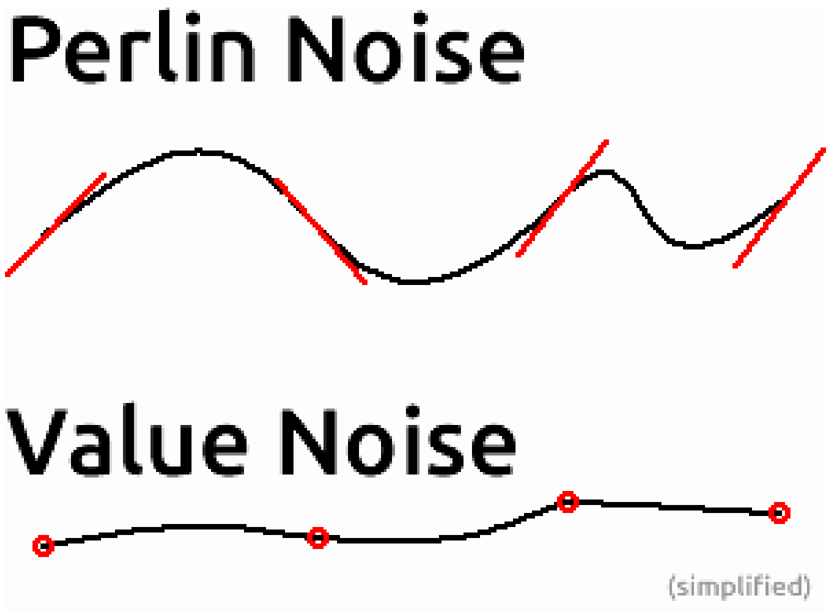


Figure 3.11: A comparison between the kinds of curves produced by Value noise interpolation and Perlin (and other Gradient) noise interpolation.[6]

Two particularly well-known Gradient noise algorithms that are commonly used for procedurally generating levels are the already mentioned Perlin Noise and Simplex Noise, both designed by American Computer Science professor Kenneth H. Perlin, with the former being an improvement on the former. Perlin Noise also takes a lattice of randomly assigned gradients, but the algorithm interpolates the dot products of those points instead of just their neighbouring values.[13] Perlin filed a patent on his work in 2002 that was granted in 2005[17], which prompted the creation of the OpenSimplex noise algorithm[21] for free use; the patent has since expired in 2022, allowing free use to both Perlin and the original Simplex noise.[17]

3.1.3 Poisson Disk Sampling

Poisson disk distributions are an easy way to randomly scatter objects across a field. It's commonly used for tree placement and placement of other random objects. Points are placed over a plane, with a single point placed randomly and subsequent points calculated such that a single point has no other point lying within a given radius of said point. Different implementations of Poisson disk distributions or samples can accommodate multiple radii for points in a plane, and some implementations produce *maximal* samples- that is, a set of samples that fully cover the given plane, while still adhering to the principle that no single point has other points lying within its radius (the implementation I made for this project does **not** guarantee maximality, however).

The following are some examples of Poisson disk distribution in action:

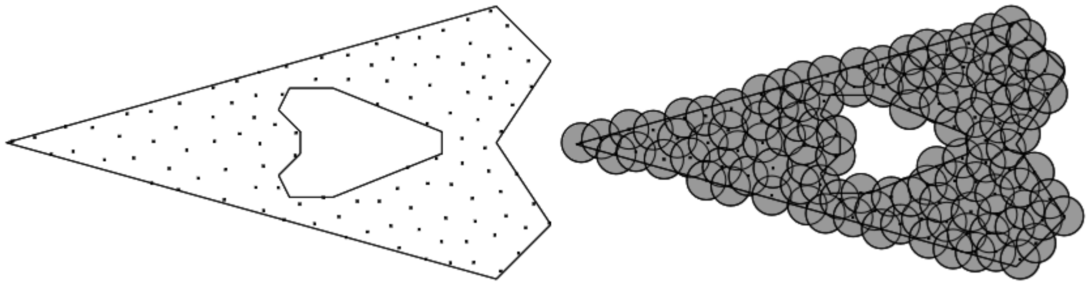


Figure 3.12: A diagram of a maximal Poisson disk distribution done on a concave plane, with the right side denoting maximality through the grey disks overlapping but not any points overlapping.[7]

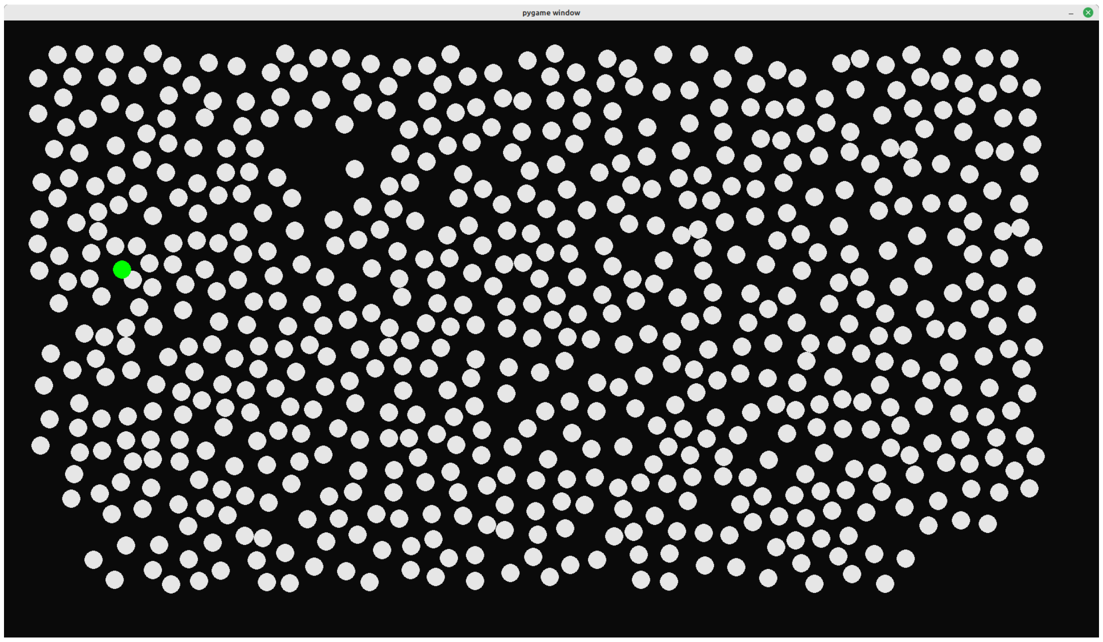


Figure 3.13: An implementation of Poisson disk sampling made in Pygame.[1] The screenshot was taken *after* all of the samples were taken.

3.1.4 Voronoï Cells

Named after the Ukranian mathematician Georgy Voronoy, Voronoï cells work by taking a map of points, and randomly selecting a group of points. Within that selected group, cells are formed by calculating, in each point of the grid, the closest of the selected points to it. That is, each cell represents the group of points that are the closest to that random point (including that point in the group as well). The final arrangement of cells represents a Voronoï Diagram or Voronoï Tesselation.

Distances between points can be calculated with either the Euclidean distance:

$$d_E(p,q) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2}$$

or the Manhattan distance:

$$d_M(p,q) = |q_x - p_x| + |q_y - p_y|$$

With the Euclidean distance producing a more “triangulated” tesselation than the Manhattan distance, the geometry of which is more ”blocky” and resembles taxicabs (hence its alternate name ”Taxicab Geometry”). A visual comparison of the kinds of cells generated with either distance calculation is shown in Figure 3.14.

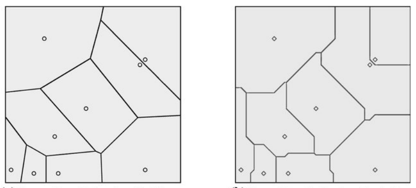

(a) Voronoi tessellation in a Euclidean space
(b) Voronoi tessellation in a Manhattan space

Figure 3.14: A visual comparison of the kinds of Voronoï cells generated with the Euclidean and Manhattan distance.[19]

3.2 Implementations

Here I will describe, at surface level, the methods I went about implementing the above algorithms and what references I used.

3.2.1 Lindenmayer System

The implementation of an L-System was very simple. I took inspiration from a YouTube video on implementing an L-System for drawing line graphics in Godot by Codat.[3] In the code from the Godot 3 project he made in that video[4][3], he created a custom “Rule” class in GDScript, with which he defined new rules. I forked his project, converted it to Godot 4 and used it to create the lattice graphics in Figure 3.7.[5] I did this mainly as a reference for my implementation of L-Systems in the game itself.

With the implementation in my *game*, I adapted the `get_new_character` method in that L-System to work with the dictionary I originally implemented my L-System in. The new `get_new_replacement` method in my implementation allows for there to be more than one grammar rule while the L-System still performs as it should. My original L-System iterated through the original string *directly*, which produced unintended consequences in grammars with multiple rules, as seen here when trying to implement the D0L-System I mentioned earlier[15]:

$$b \rightarrow a \rightarrow aa \rightarrow aaa \rightarrow aaaa \rightarrow aaaaa \ldots$$

By using an empty string buffer and inserting rule replacements there instead, my implementation is now able to perform substitutions accordingly; the correct computation of the D0L-System is denoted in Figure 3.9 and repeated below:

$$b \rightarrow a \rightarrow ab \rightarrow aba \rightarrow abaab \rightarrow abaababa \ldots$$

With the L-System string parsing algorithm in place, the next step was to paint the cells of each tile. With this, I iterated through every cell of the tilemap using a nested for-loop. With the parsed string, I then accessed the character of the string at an incremented index using an iterator variable I defined before the for-loops. The string consists of three different characters repeated multiple times, “O”, “W” and “B”. For each string index, if the character is “W”, paint a tree, if it is “B”, paint a building, and if it is an “O”, leave the cell blank and paint nothing. The player and ring then get placed afterwards.

Even for a large-sized tile map with 2880 cells, a constant L-System G, with the symbols O, W and B and the following grammar

$$\mathrm{O} \rightarrow \mathrm{OWO}$$

$$\mathrm{W} \rightarrow \mathrm{WB}$$

$$\mathrm{B} \rightarrow \mathrm{BWO}$$

can parse the axiom OWB, paint tile map tiles with the resulting string **and** place the player and ring in just a minimum of 19 milliseconds and a maximum of 22 milliseconds.

3.2.2 Perlin/Simplex Noise

The Simplex Noise implementation works with Godot’s built-in Noise library. Within a Sprite2D node’s Texture attribute, I set a new “NoiseTexture2D” field inside of it. In its “Noise” attribute I created a new “FastNoiseLite” scene, which generates a noise texture for us to use. The seed can be set in the sprite’s script file.

As with my other implementations, there are two separate arrays, one for trees and another for buildings. For each cell in the TileMap, I then took the noise pixel from the generated texture at that exact point (scaling with the cell size accordingly), using the `get_noise_2d` method built-in with Godot, and then, depending on the value retrieved, decided, firstly, whether or not to place a plant/tree tile there and, secondly, whether or not to place a building tile there. As a result, not every cell in the TileMap has tiles on it. On any one of those empty cells, the Player tile will then get placed.

For the generation of the noise itself, I *could've* added a `Sprite2D` node to the scene tree, the root of which was my `TileMap`, and gave it a `NoiseTexture2D` texture and set its `noise` property to a newly-created `FastNoiseLite` instance, the latter of which contains the actual noise data. In the early stages of this implementation's development, that's what I did, and I created a script that solely set the seed of the `FastNoiseLite` resource to a random integer (using the `randi` method). However, for a more authentic result, and to forgo the need of an additional node and noise texture that will not even be visible in the final product, I decided to create the noise for this algorithm implementation entirely programmatically. I stored the `FastNoiseLite` instance in its own class variable `noise`, and instantiated it with the `set_noise` method when starting the game (the `_ready` function automatically runs when the game starts).

Initially having done the noise integration with a sprite node and noise texture allowed me to experiment with some of the `FastNoiseLite` class's properties before finally resorting to programmatic noise creation. An instance of this class, by default, uses the "Simplex Smooth" noise algorithm, a version of the Simplex algorithm that produces higher quality noise images at the expense of slower speed.[13] We can also use just "Simplex" noise for higher speed, as well as the original "Perlin" noise algorithm.[13] Godot also allows us to use two kinds of Value noise, as well as a "Cellular" type that combines algorithms like Worley Noise and Voronoï diagrams to create "regions of the same value."[13] I had problems with the "Cellular" noise type when experimenting with it, for reasons I will get into later, but the other noise types I made readily accessible in an "export" variable in my script (that is, a variable that can be easily accessed in the Godot editor when the TileMap node is clicked on) when I removed the sprite node and decided to programmatically make the noise. When the `set_noise` function is called, the noise type is assigned through the `_get_noise_type` function, which returns an integer value depending on the type of noise selected, and the returned result is cast to `FastNoiseLite`'s `NoiseType` enumeration[13] before it gets assigned (this prevents an `INT_AS_ENUM_WITHOUT_CAST` warning from the Godot editor's linter for GDScript[14]).

Furthermore, I have 3 other export variables in the TileMap script for this implementation that directly correlate to some of `FastNoiseLite`'s properties. The `noise_frequency` variable in the script correlates to the `frequency` property in `FastNoiseLite`, which, as both names suggest, sets the noise frequency; the higher the frequency, the rougher and more granular the noise[13], which is probably why it is set to 0.01 by default.[13] The `fractal_type` and `cellular_distance_type` in the script **directly** correspond to the `fractal_type` and `cellular_distance_function` properties respectively, to the point where both even use the relevant enumerations from `FastNoiseLite` directly (`FractalType` and `CellularDistanceFunction` respectively).[13] The relevant values are all assigned accordingly in `set_noise`.

In terms of determining whether or not to place buildings or trees (or nothing), I took inspiration from a YouTube tutorial by Gingerageous Games utilising Godot 3[9][10] (which breaks in Godot 4). His tutorial used multiple `TileMap` nodes in a single scene tree with a `Node2D` root, and controlled each individual tile map, representing a specific part of the environment (such as grass and roads), and used a floating point "cap" to determine whether or not to place a tile in a cell based on the noise pixel retrieved at that cell's coordinate.[9][10] Since I'm using just one tile map for everything (trees and buildings), I had to mitigate a conflict where the building cap was smaller than the tree cap. If that were the case then, since the tree cells get painted first in my implementation, no buildings would ever get painted. To mitigate this, I added an additional condition to my if-statement for painting building cells (in the same line, to prevent creating a nested if-statement), which would allow the algorithm to overwrite an already painted tree cell with a building cell subject to a randomly generated floating point number (between 0 and 1 inclusive) being below a pre-defined floating point number in the exported variable `building_overtakes_tree`. This would then allow there to be a controlled proportion of buildings compared to trees (the higher the proportion, the more buildings compared to trees), regardless of whether the building cap was lower than the tree cap or not, and the algorithm would still perform as normal should the reverse be the case.

3.2.3 Poisson Disk Sampling

3.2.4 Voronoï Cells

Chapter 4

Design & Specification

Here, I will provide an abstract level of how I compared the performance of each content generation algorithm and how I made sure each implementation could produce as similar/like-for-like results as possible (and where they *couldn't* do so).

4.1 Performance

With the L-System implementation, I had no problems running the game very quickly on my machine, and quickly got results.

With Poisson Disk Sampling, the higher the number of rejection samples (that is, the higher the maximum number of times a cell was sampled before it was either accepted or ultimately rejected), the longer it took to generate a complete level layout, and even, due to the nature of the tile map compared to the algorithm's *usual* use (of scattering dots on a plane), it was not maximal (not all points had cells painted for them; some cells had their tiles overwritten as well). Using 8 rejection samples was usually enough to yield a satisfactroy level layout.

Voronoï Cells took the longest to compute on average.

4.2 Layouts

Of the 4 implementations I made, the Noise and Poisson Disk Sampling implementation were by far the most similar, followed by the L-System implementation, and then the Voronoï Cells implementation, which was far and away the most unique.

While the noise implementations were

Chapter 5

Implementation

Here I will go a bit deeper as to how I made each algorithm work.

5.1 Section Heading

Chapter 6

Legal, Social, Ethical and Professional Issues

Your report should include a chapter with a reasoned discussion about legal, social ethical and professional issues within the context of your project problem. You should also demonstrate that you are aware of the regulations governing your project area and the Code of Conduct & Code of Good Practice issued by the British Computer Society, and that you have applied their principles, where appropriate, as you carried out your project.

6.1 Section Heading

Chapter 7

Results/Evaluation

Here I will mention how I tested the small games and made sure they ran as they should.

7.1 Software Testing

7.2 Section Heading

Chapter 8

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

References

[1] Joseph Bakulikira. Poisson disc sampling in pygame. [Online; accessed 15-April-2023].

[2] Paul Bourke. L-system user notes. July 1991.

[3] Codat. Code that: L-system, 2020.

[4] Codat and contributors. An implementation of the rewriting system: Lindenmayer system., April 2020. [Online; accessed 10-April-2023].

[5] Codat, Zishan Rahman, and contributors. An implementation of the rewriting system: Lindenmayer system., March 2023. [Online; accessed 10-April-2023].

[6] Timm (Stack Exchange Contributor). "benefit of perlin noise over value noise" - stack exchange answer, June 2016. [Online, Accessed 16-April-2023].

[7] Mohamed S. Ebeida, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, Scott A. Mitchell, and John D. Owens. Efficient maximal poisson-disk sampling. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery.

[8] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1:27, 10 2017.

[9] Gingerageous Games. Autotiles opensimplex noise procedural generation godot 3.1 tutorial, August 2019. [Online, Accessed 11-March-2023].

[10] Gingerageous Games. Opensimplexnoisetilemaptutorial, March 2020. [Online, Accessed 11-March-2023].

[11] Hernan J. González. "why is gradient noise better quality than value noise?" - stack exckange answer, August 2012. [Online, Accessed 16-April-2023, Stack Exchange Username "leonbloy"].

[12] Aristid Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, 1968.

[13] Juan Linietsky, Ariel Manzur, and the Godot community. Fastnoiselite — godot engine (stable) documentation in english.

[14] Juan Linietsky, Ariel Manzur, and the Godot community. Projectsettings — godot engine (stable) documentation in english. [Online, Accessed 16-April-2023, GDScript Warnings Classed as Properties, License CC-BY-3.0].

[15] Gabriela Ochoa. An introduction to lindenmayer systems: D0l-system, February 1998. [Online; accessed 11-April-2023].

[16] Guilherme Oliveira. “intro to procedural levels in godot 3.1: Perlin noise tutorial”.

[17] Kenneth H. Perlin. Standard for perlin noise. `https://patents.google.com/patent/US6867776`, January 2002. [Online patent document, Accessed 16-April-2023].

[18] Robson. An interactive webpage for generating images that simulate white noise., April 2021. [Online, Accessed 16-April-2023].

[19] Corina Strößner. Criteria for naturalness in conceptual spaces. *Synthese*, 200, 04 2022.

[20] Wikipedia contributors. L-system — Wikipedia, the free encyclopedia, 2022. [Online; accessed 23-February-2023].

[21] Wikipedia contributors. Opensimplex noise — Wikipedia, the free encyclopedia, 2022. [Online; accessed 16-April-2023].

Appendix A

Extra Information

A.1 Tables, proofs, graphs, test cases, ...

The appendices contain information that is peripheral to the main body of the report. Information typically included in the Appendix are things like tables, proofs, graphs, test cases or any other material that would break up the theme of the text if it appeared in the body of the report. It is necessary to include your source code listings in an appendix that is separate from the body of your written report (see the information on Program Listings below).

Appendix B

User Guide

B.1 Instructions

To run the projects in the .zip file, extract the projects in one folder. Then open Godot 4 (at the moment all projects are Godot 4 projects), and, when opening the Godot editor, click ”Scan”, then go to that folder and select it. The projects can then be opened in the project manager and edited as needed in Godot. When you click on some of the scenes in the projects, there may be some ”exported” variables from scripts that are visible to you in the editor (examples include the ”Distance” and ”Random Starting Points” variables in the Voronoi Cells project).

Appendix C

Source Code

C.1 Instructions

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a “table of contents” (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: “I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary”. Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted to KEATS. You are required to keep safely several copies of this version of the program and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you have uploaded to KEATS. Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report. There are many free tools available that allow you to merge PDF files.

C.2 LSystemGrammarDemo

C.2.1 .gitattributes

```
# Normalize EOL for all files that Git considers text files.
* text=auto eol=lf
```

C.2.2 .gitignore

```
# Godot 4+ specific ignores
.godot/
```

C.2.3 project.godot

```
; Engine configuration file.
; It's best edited using the editor UI and not directly,
; since the parameters that go here are not all obvious.
;
; Format:
;   [section] ; section goes between []
;   param=value ; assign values to parameters

config_version=5

[application]

config/name="LSystemGrammarDemo"
run/main_scene="res://DemoNode.tscn"
config/features=PackedStringArray("4.0")

[display]
```

```

window/stretch/mode="canvas_items"
window/stretch/aspect="expand"

[gui]

common/drop_mouse_on_gui_input_disabled=true

[physics]

common/enable_pause_aware_picking=true
```

C.2.4 DemoNode.tscn

```
[gd_scene load_steps=2 format=3 uid="uid://bu380we4od0ln"]

[ext_resource type="Script" path="res://DemoNode.gd" id="1"]

[node name="DemoNode" type="Node"]
script = ExtResource("1")
choices = "deterministic"

[node name="Timer" type="Timer" parent="."]

[node name="Line2D" type="Line2D" parent="."]
points = PackedVector2Array(69, 297)

[node name="TextLabel" type="Label" parent="."]
offset_right = 1152.0
offset_bottom = 23.0
autowrap_mode = 3

```

```
[connection signal="timeout" from="Timer" to="." method="
    _on_Timer_timeout"]
```

C.2.5 DemoNode.gd

```
extends Node

# Basic: https://youtu.be/feNVBEPXAcE?t=77 (L = +)
# Choices: http://paulbourke.net/fractals/lsys/
# Deterministic: https://www1.biologie.uni-hamburg.de/b-online/
    e28_3/lsys.html#D0L-system

@export_enum("basic", "choices", "deterministic") var choices:
    String = "choices"
@export var axiom: String
@onready var string: String
@onready var timer = \${Timer}
@onready var line = \${Line2D}
@onready var label = \${TextLabel}
@onready var rules: Array[Dictionary]

func set_values():
   if choices == "basic":
      rules = [
         {
            "from": "F",
            "to": "F+F"
         }
      ]
      axiom = "F+"
   elif choices == "choices":
      rules = [
            {
                "from": "F",
                "to": "F+--FFFF+F+-FF"
            }
        ]
        axiom = "F+F+F+F"
    elif choices == "deterministic":
        rules = [
            {
                "from": "a",
                "to": "ab"
            },
            {
                "from": "b",
                "to": "a"
            }
        ]
        axiom = "b"

func _ready():
    set_values()
    string = axiom
    label.size.x = get_viewport().size.x
    label.text = string
    timer.start()

func get_new_replacement(character: String) -> String:
    for rule in rules:
        if rule["from"] == character:
            return rule["to"]
    return ""

func _on_Timer_timeout():
    var new_string = ""
    for character in string:
        new_string += get_new_replacement(character)
    string = new_string
    label.text = string
    print(len(string))
```

C.2.6 icon.svg.import

```
[remap]

importer="texture"
type="CompressedTexture2D"
uid="uid://cwnnuqmejj04q"
path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
metadata={
"vram_texture": false
}

[deps]

source_file="res://icon.svg"
dest_files=["res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
```

```
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
svg/scale=1.0
editor/scale_with_editor_scale=false
editor/convert_colors_with_editor_theme=false
```

C.3 ProcGenRPG (L-System)

C.3.1 .gitattributes

```
# Normalize EOL for all files that Git considers text files.
* text=auto eol=lf
```

C.3.2 .gitignore

```
# Godot 4+ specific ignores
.godot/
```

C.3.3 project.godot

```
; Engine configuration file.
; It's best edited using the editor UI and not directly,
; since the parameters that go here are not all obvious.
;
; Format:
;   [section] ; section goes between []
;   param=value ; assign values to parameters

config_version=5

[application]

config/name="Proc Gen RPG"
run/main_scene="res://tile_map.tscn"
config/features=PackedStringArray("4.0", "Forward Plus")
config/icon="res://icon.svg"

[display]

window/size/viewport_height=640
window/stretch/mode="canvas_items"
```

C.3.4 l_system.tscn

```
[gd_scene load_steps=2 format=3 uid="uid://d0v18e7ms571f"]

[ext_resource type="Script" path="res://l_system.gd" id="1_elydp"]

[node name="LSystem" type="Node"]
script = ExtResource("1_elydp")
```

C.3.5 l_system.gd

```
extends Node

class_name LSystem

@onready var tile_map: TileMap = get_parent()
@export var axiom: String = "OWB"
@onready var string: String = axiom
@export var rules: Array[Dictionary] = [
   {
      "from": "O",
      "to": "OWO"
   },
   {
      "from": "W",
      "to": "WB"
   },
   {
      "from": "B",
      "to": "BWO"
   }
]

const FLOWERS_1: Vector2i = Vector2i(3, 7) # "O" = ORANGE
const FLOWERS_2: Vector2i = Vector2i(3, 10) # "W" = WHITE
const FLOWERS_3: Vector2i = Vector2i(3, 13) # "B" = BLUE

func get_new_replacement(character: String) -> String:
   for rule in rules:
      if rule["from"] == character:
         return rule["to"]
   return ""
```

```

func size() -> int:
    return tile_map.x_tile_range * tile_map.y_tile_range

func parse() -> String:
    var size: int = size()
    while len(string) <= size:
        var new_string = ""
        for character in string:
            new_string += get_new_replacement(character)
        string = new_string
    string = string.substr(0, size)
    return string

func paint() -> void:
    string = parse()
    var size: int = size()
    var i: int = -1
    for x in range(tile_map.x_tile_range):
        for y in range(tile_map.y_tile_range):
            i += 1
            if string[i] == "O":
                tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_1)
            elif string[i] == "W":
                tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_2)
            elif string[i] == "B":
                tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_3)
```

C.3.6 tile_map.tscn

```
[gd_scene load_steps=4 format=3 uid="uid://bwhvtqld3yo8m"]

```

```
[ext_resource type="TileSet" uid="uid://c168x78r0tful" path="res://
    Tiles.tres" id="1_l3nwg"]
[ext_resource type="Script" path="res://tile_map.gd" id="2_wrxl8"]
[ext_resource type="PackedScene" uid="uid://d0v18e7ms571f" path="
    res://l_system.tscn" id="3_ktw1n"]

[node name="TileMap" type="TileMap"]
tile_set = ExtResource("1_l3nwg")
format = 2
layer_0/name = "Grass"
layer_1/name = "Things"
layer_1/enabled = true
layer_1/modulate = Color(1, 1, 1, 1)
layer_1/y_sort_enabled = false
layer_1/y_sort_origin = 0
layer_1/z_index = 0
layer_1/tile_data = PackedInt32Array()
script = ExtResource("2_wrxl8")

[node name="LSystem" parent="." instance=ExtResource("3_ktw1n")]
```

C.3.7 tile_map.gd

```
extends TileMap

@onready var l_system: LSystem = \${LSystem}

var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y

```

```
const GRASS_1: Vector2i = Vector2i(5, 0)
const GRASS_2: Vector2i = Vector2i(5, 1)

func pick_grass_tile() -> Vector2i:
    return [GRASS_1, GRASS_2].pick_random()

func cover_map_with_grass() -> void:
    for x in range(-50, x_tile_range + 50):
        for y in range(-50, y_tile_range + 50):
            set_cell(0, Vector2i(x, y), 0, pick_grass_tile())

\# Called when the node enters the scene tree for the first time.
func _ready() -> void:
    randomize()
    cover_map_with_grass()
    l_system.paint()
```

C.3.8 icon.svg.import

```
[remap]

importer="texture"
type="CompressedTexture2D"
uid="uid://b45qexb3wmhym"
path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
metadata={
"vram_texture": false
}

[deps]

```

```
source_file="res://icon.svg"
dest_files=["res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
svg/scale=1.0
editor/scale_with_editor_scale=false
editor/convert_colors_with_editor_theme=false
```

C.3.9 roguelikeSheet_transparent.png.import

```
[remap]

importer="texture"
```

```
type="CompressedTexture2D"
uid="uid://13ktp0qup5xb"
path="res://.godot/imported/roguelikeSheet_transparent.png-22
    f6b70da04549e371d1f15fe9d96005.ctex"
metadata={
"vram_texture": false
}

[deps]

source_file="res://roguelikeSheet_transparent.png"
dest_files=["res://.godot/imported/roguelikeSheet_transparent.png
    -22f6b70da04549e371d1f15fe9d96005.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
```

C.4 VoronoiCellsGD4

C.4.1 .gitattributes

```
# Normalize EOL for all files that Git considers text files.
* text=auto eol=lf
```

C.4.2 .gitignore

```
# Godot 4+ specific ignores
.godot/
```

C.4.3 project.godot

```
; Engine configuration file.
; It's best edited using the editor UI and not directly,
; since the parameters that go here are not all obvious.
;
; Format:
;   [section] ; section goes between []
;   param=value ; assign values to parameters

config_version=5

[application]

config/name="Voronoi Cells"
run/main_scene="res://tile_map.tscn"
config/features=PackedStringArray("4.0", "Forward Plus")
config/icon="res://icon.svg"

```

```
[display]

window/size/viewport_height=640
```

C.4.4 tile_map.tscn

```
[gd_scene load_steps=5 format=3 uid="uid://d6lxnr5bdh1w"]

[ext_resource type="Texture2D" uid="uid://cpign73sfbsrt" path="res
    ://monochrome_packed.png" id="1_o183d"]
[ext_resource type="Script" path="res://tile_map.gd" id="2_lf4lw"]

[sub_resource type="TileSetAtlasSource" id="
    TileSetAtlasSource_6h0bd"]
texture = ExtResource("1_o183d")
0:0/0 = 0
1:0/0 = 0
2:0/0 = 0
3:0/0 = 0
4:0/0 = 0
5:0/0 = 0
6:0/0 = 0
7:0/0 = 0
8:0/0 = 0
9:0/0 = 0
10:0/0 = 0
11:0/0 = 0
12:0/0 = 0
13:0/0 = 0
14:0/0 = 0
15:0/0 = 0
16:0/0 = 0
```

```
17:0/0 = 0
18:0/0 = 0
19:0/0 = 0
20:0/0 = 0
21:0/0 = 0
22:0/0 = 0
23:0/0 = 0
24:0/0 = 0
25:0/0 = 0
26:0/0 = 0
27:0/0 = 0
28:0/0 = 0
29:0/0 = 0
30:0/0 = 0
31:0/0 = 0
32:0/0 = 0
33:0/0 = 0
34:0/0 = 0
35:0/0 = 0
36:0/0 = 0
37:0/0 = 0
38:0/0 = 0
39:0/0 = 0
40:0/0 = 0
41:0/0 = 0
42:0/0 = 0
43:0/0 = 0
44:0/0 = 0
45:0/0 = 0
46:0/0 = 0
47:0/0 = 0
48:0/0 = 0
0:1/0 = 0
```

```
1:1/0 = 0
2:1/0 = 0
3:1/0 = 0
4:1/0 = 0
5:1/0 = 0
6:1/0 = 0
7:1/0 = 0
8:1/0 = 0
9:1/0 = 0
10:1/0 = 0
11:1/0 = 0
12:1/0 = 0
13:1/0 = 0
14:1/0 = 0
15:1/0 = 0
16:1/0 = 0
17:1/0 = 0
18:1/0 = 0
19:1/0 = 0
20:1/0 = 0
21:1/0 = 0
22:1/0 = 0
23:1/0 = 0
24:1/0 = 0
25:1/0 = 0
26:1/0 = 0
27:1/0 = 0
28:1/0 = 0
29:1/0 = 0
30:1/0 = 0
31:1/0 = 0
32:1/0 = 0
33:1/0 = 0
```

```
34:1/0 = 0
35:1/0 = 0
36:1/0 = 0
37:1/0 = 0
38:1/0 = 0
39:1/0 = 0
40:1/0 = 0
41:1/0 = 0
42:1/0 = 0
43:1/0 = 0
44:1/0 = 0
45:1/0 = 0
46:1/0 = 0
47:1/0 = 0
48:1/0 = 0
0:2/0 = 0
1:2/0 = 0
2:2/0 = 0
3:2/0 = 0
4:2/0 = 0
5:2/0 = 0
6:2/0 = 0
7:2/0 = 0
8:2/0 = 0
9:2/0 = 0
10:2/0 = 0
11:2/0 = 0
12:2/0 = 0
13:2/0 = 0
14:2/0 = 0
15:2/0 = 0
16:2/0 = 0
17:2/0 = 0
```

```
18:2/0 = 0
19:2/0 = 0
20:2/0 = 0
21:2/0 = 0
22:2/0 = 0
23:2/0 = 0
24:2/0 = 0
25:2/0 = 0
26:2/0 = 0
27:2/0 = 0
28:2/0 = 0
29:2/0 = 0
30:2/0 = 0
31:2/0 = 0
32:2/0 = 0
33:2/0 = 0
34:2/0 = 0
35:2/0 = 0
36:2/0 = 0
37:2/0 = 0
38:2/0 = 0
39:2/0 = 0
40:2/0 = 0
41:2/0 = 0
42:2/0 = 0
43:2/0 = 0
44:2/0 = 0
45:2/0 = 0
46:2/0 = 0
47:2/0 = 0
48:2/0 = 0
0:3/0 = 0
1:3/0 = 0
```

```
2:3/0 = 0
3:3/0 = 0
4:3/0 = 0
5:3/0 = 0
6:3/0 = 0
7:3/0 = 0
8:3/0 = 0
9:3/0 = 0
10:3/0 = 0
11:3/0 = 0
12:3/0 = 0
13:3/0 = 0
14:3/0 = 0
15:3/0 = 0
16:3/0 = 0
17:3/0 = 0
18:3/0 = 0
19:3/0 = 0
20:3/0 = 0
21:3/0 = 0
22:3/0 = 0
23:3/0 = 0
24:3/0 = 0
25:3/0 = 0
26:3/0 = 0
27:3/0 = 0
28:3/0 = 0
29:3/0 = 0
30:3/0 = 0
31:3/0 = 0
32:3/0 = 0
33:3/0 = 0
34:3/0 = 0
```

```
190    35:3/0 = 0
191    36:3/0 = 0
192    37:3/0 = 0
193    38:3/0 = 0
194    39:3/0 = 0
195    40:3/0 = 0
196    41:3/0 = 0
197    42:3/0 = 0
198    43:3/0 = 0
199    44:3/0 = 0
200    45:3/0 = 0
201    46:3/0 = 0
202    47:3/0 = 0
203    48:3/0 = 0
204    0:4/0 = 0
205    1:4/0 = 0
206    2:4/0 = 0
207    3:4/0 = 0
208    4:4/0 = 0
209    5:4/0 = 0
210    6:4/0 = 0
211    7:4/0 = 0
212    8:4/0 = 0
213    9:4/0 = 0
214    10:4/0 = 0
215    11:4/0 = 0
216    12:4/0 = 0
217    13:4/0 = 0
218    14:4/0 = 0
219    15:4/0 = 0
220    16:4/0 = 0
221    17:4/0 = 0
222    18:4/0 = 0
```

```
19:4/0 = 0
20:4/0 = 0
21:4/0 = 0
22:4/0 = 0
23:4/0 = 0
24:4/0 = 0
25:4/0 = 0
26:4/0 = 0
27:4/0 = 0
28:4/0 = 0
29:4/0 = 0
30:4/0 = 0
31:4/0 = 0
32:4/0 = 0
33:4/0 = 0
34:4/0 = 0
35:4/0 = 0
36:4/0 = 0
37:4/0 = 0
38:4/0 = 0
39:4/0 = 0
40:4/0 = 0
41:4/0 = 0
42:4/0 = 0
43:4/0 = 0
44:4/0 = 0
45:4/0 = 0
46:4/0 = 0
47:4/0 = 0
48:4/0 = 0
0:5/0 = 0
1:5/0 = 0
2:5/0 = 0
```

```
3:5/0 = 0
4:5/0 = 0
5:5/0 = 0
6:5/0 = 0
7:5/0 = 0
8:5/0 = 0
9:5/0 = 0
10:5/0 = 0
11:5/0 = 0
12:5/0 = 0
13:5/0 = 0
14:5/0 = 0
15:5/0 = 0
16:5/0 = 0
17:5/0 = 0
18:5/0 = 0
19:5/0 = 0
20:5/0 = 0
21:5/0 = 0
22:5/0 = 0
23:5/0 = 0
24:5/0 = 0
25:5/0 = 0
26:5/0 = 0
27:5/0 = 0
28:5/0 = 0
29:5/0 = 0
30:5/0 = 0
31:5/0 = 0
32:5/0 = 0
33:5/0 = 0
34:5/0 = 0
35:5/0 = 0
```

```
36:5/0 = 0
37:5/0 = 0
38:5/0 = 0
39:5/0 = 0
40:5/0 = 0
41:5/0 = 0
42:5/0 = 0
43:5/0 = 0
44:5/0 = 0
45:5/0 = 0
46:5/0 = 0
47:5/0 = 0
48:5/0 = 0
0:6/0 = 0
1:6/0 = 0
2:6/0 = 0
3:6/0 = 0
4:6/0 = 0
5:6/0 = 0
6:6/0 = 0
7:6/0 = 0
8:6/0 = 0
9:6/0 = 0
10:6/0 = 0
11:6/0 = 0
12:6/0 = 0
13:6/0 = 0
14:6/0 = 0
15:6/0 = 0
16:6/0 = 0
17:6/0 = 0
18:6/0 = 0
19:6/0 = 0
```

```
20:6/0 = 0
21:6/0 = 0
22:6/0 = 0
23:6/0 = 0
24:6/0 = 0
25:6/0 = 0
26:6/0 = 0
27:6/0 = 0
28:6/0 = 0
29:6/0 = 0
30:6/0 = 0
31:6/0 = 0
32:6/0 = 0
33:6/0 = 0
34:6/0 = 0
35:6/0 = 0
36:6/0 = 0
37:6/0 = 0
38:6/0 = 0
39:6/0 = 0
40:6/0 = 0
41:6/0 = 0
42:6/0 = 0
43:6/0 = 0
44:6/0 = 0
45:6/0 = 0
46:6/0 = 0
47:6/0 = 0
48:6/0 = 0
0:7/0 = 0
1:7/0 = 0
2:7/0 = 0
3:7/0 = 0
```

```
4:7/0 = 0
5:7/0 = 0
6:7/0 = 0
7:7/0 = 0
8:7/0 = 0
9:7/0 = 0
10:7/0 = 0
11:7/0 = 0
12:7/0 = 0
13:7/0 = 0
14:7/0 = 0
15:7/0 = 0
16:7/0 = 0
17:7/0 = 0
18:7/0 = 0
19:7/0 = 0
20:7/0 = 0
21:7/0 = 0
22:7/0 = 0
23:7/0 = 0
24:7/0 = 0
25:7/0 = 0
26:7/0 = 0
27:7/0 = 0
28:7/0 = 0
29:7/0 = 0
30:7/0 = 0
31:7/0 = 0
32:7/0 = 0
33:7/0 = 0
34:7/0 = 0
35:7/0 = 0
36:7/0 = 0
```

```
37:7/0 = 0
38:7/0 = 0
39:7/0 = 0
40:7/0 = 0
41:7/0 = 0
42:7/0 = 0
43:7/0 = 0
44:7/0 = 0
45:7/0 = 0
46:7/0 = 0
47:7/0 = 0
48:7/0 = 0
0:8/0 = 0
1:8/0 = 0
2:8/0 = 0
3:8/0 = 0
4:8/0 = 0
5:8/0 = 0
6:8/0 = 0
7:8/0 = 0
8:8/0 = 0
9:8/0 = 0
10:8/0 = 0
11:8/0 = 0
12:8/0 = 0
13:8/0 = 0
14:8/0 = 0
15:8/0 = 0
16:8/0 = 0
17:8/0 = 0
18:8/0 = 0
19:8/0 = 0
20:8/0 = 0
```

```
21:8/0 = 0
22:8/0 = 0
23:8/0 = 0
24:8/0 = 0
25:8/0 = 0
26:8/0 = 0
27:8/0 = 0
28:8/0 = 0
29:8/0 = 0
30:8/0 = 0
31:8/0 = 0
32:8/0 = 0
33:8/0 = 0
34:8/0 = 0
35:8/0 = 0
36:8/0 = 0
37:8/0 = 0
38:8/0 = 0
39:8/0 = 0
40:8/0 = 0
41:8/0 = 0
42:8/0 = 0
43:8/0 = 0
44:8/0 = 0
45:8/0 = 0
46:8/0 = 0
47:8/0 = 0
48:8/0 = 0
0:9/0 = 0
1:9/0 = 0
2:9/0 = 0
3:9/0 = 0
4:9/0 = 0
```

```
5:9/0 = 0
6:9/0 = 0
7:9/0 = 0
8:9/0 = 0
9:9/0 = 0
10:9/0 = 0
11:9/0 = 0
12:9/0 = 0
13:9/0 = 0
14:9/0 = 0
15:9/0 = 0
16:9/0 = 0
17:9/0 = 0
18:9/0 = 0
19:9/0 = 0
20:9/0 = 0
21:9/0 = 0
22:9/0 = 0
23:9/0 = 0
24:9/0 = 0
25:9/0 = 0
26:9/0 = 0
27:9/0 = 0
28:9/0 = 0
29:9/0 = 0
30:9/0 = 0
31:9/0 = 0
32:9/0 = 0
33:9/0 = 0
34:9/0 = 0
35:9/0 = 0
36:9/0 = 0
37:9/0 = 0
```

```
38:9/0 = 0
39:9/0 = 0
40:9/0 = 0
41:9/0 = 0
42:9/0 = 0
43:9/0 = 0
44:9/0 = 0
45:9/0 = 0
46:9/0 = 0
47:9/0 = 0
48:9/0 = 0
0:10/0 = 0
1:10/0 = 0
2:10/0 = 0
3:10/0 = 0
4:10/0 = 0
5:10/0 = 0
6:10/0 = 0
7:10/0 = 0
8:10/0 = 0
9:10/0 = 0
10:10/0 = 0
11:10/0 = 0
12:10/0 = 0
13:10/0 = 0
14:10/0 = 0
15:10/0 = 0
16:10/0 = 0
17:10/0 = 0
18:10/0 = 0
19:10/0 = 0
20:10/0 = 0
21:10/0 = 0
```

```
22:10/0 = 0
23:10/0 = 0
24:10/0 = 0
25:10/0 = 0
26:10/0 = 0
27:10/0 = 0
28:10/0 = 0
29:10/0 = 0
30:10/0 = 0
31:10/0 = 0
32:10/0 = 0
33:10/0 = 0
34:10/0 = 0
35:10/0 = 0
36:10/0 = 0
37:10/0 = 0
38:10/0 = 0
39:10/0 = 0
40:10/0 = 0
41:10/0 = 0
42:10/0 = 0
43:10/0 = 0
44:10/0 = 0
45:10/0 = 0
46:10/0 = 0
47:10/0 = 0
48:10/0 = 0
0:11/0 = 0
1:11/0 = 0
2:11/0 = 0
3:11/0 = 0
4:11/0 = 0
5:11/0 = 0
```

```
6:11/0 = 0
7:11/0 = 0
8:11/0 = 0
9:11/0 = 0
10:11/0 = 0
11:11/0 = 0
12:11/0 = 0
13:11/0 = 0
14:11/0 = 0
15:11/0 = 0
16:11/0 = 0
17:11/0 = 0
18:11/0 = 0
19:11/0 = 0
20:11/0 = 0
21:11/0 = 0
22:11/0 = 0
23:11/0 = 0
24:11/0 = 0
25:11/0 = 0
26:11/0 = 0
27:11/0 = 0
28:11/0 = 0
29:11/0 = 0
30:11/0 = 0
31:11/0 = 0
32:11/0 = 0
33:11/0 = 0
34:11/0 = 0
35:11/0 = 0
36:11/0 = 0
37:11/0 = 0
38:11/0 = 0
```

```
39:11/0 = 0
40:11/0 = 0
41:11/0 = 0
42:11/0 = 0
43:11/0 = 0
44:11/0 = 0
45:11/0 = 0
46:11/0 = 0
47:11/0 = 0
48:11/0 = 0
0:12/0 = 0
1:12/0 = 0
2:12/0 = 0
3:12/0 = 0
4:12/0 = 0
5:12/0 = 0
6:12/0 = 0
7:12/0 = 0
8:12/0 = 0
9:12/0 = 0
10:12/0 = 0
11:12/0 = 0
12:12/0 = 0
13:12/0 = 0
14:12/0 = 0
15:12/0 = 0
16:12/0 = 0
17:12/0 = 0
18:12/0 = 0
19:12/0 = 0
20:12/0 = 0
21:12/0 = 0
22:12/0 = 0
```

```
23:12/0 = 0
24:12/0 = 0
25:12/0 = 0
26:12/0 = 0
27:12/0 = 0
28:12/0 = 0
29:12/0 = 0
30:12/0 = 0
31:12/0 = 0
32:12/0 = 0
33:12/0 = 0
34:12/0 = 0
35:12/0 = 0
36:12/0 = 0
37:12/0 = 0
38:12/0 = 0
39:12/0 = 0
40:12/0 = 0
41:12/0 = 0
42:12/0 = 0
43:12/0 = 0
44:12/0 = 0
45:12/0 = 0
46:12/0 = 0
47:12/0 = 0
48:12/0 = 0
0:13/0 = 0
1:13/0 = 0
2:13/0 = 0
3:13/0 = 0
4:13/0 = 0
5:13/0 = 0
6:13/0 = 0
```

```
7:13/0 = 0
8:13/0 = 0
9:13/0 = 0
10:13/0 = 0
11:13/0 = 0
12:13/0 = 0
13:13/0 = 0
14:13/0 = 0
15:13/0 = 0
16:13/0 = 0
17:13/0 = 0
18:13/0 = 0
19:13/0 = 0
20:13/0 = 0
21:13/0 = 0
22:13/0 = 0
23:13/0 = 0
24:13/0 = 0
25:13/0 = 0
26:13/0 = 0
27:13/0 = 0
28:13/0 = 0
29:13/0 = 0
30:13/0 = 0
31:13/0 = 0
32:13/0 = 0
33:13/0 = 0
34:13/0 = 0
35:13/0 = 0
36:13/0 = 0
37:13/0 = 0
38:13/0 = 0
39:13/0 = 0
```

```
40:13/0 = 0
41:13/0 = 0
42:13/0 = 0
43:13/0 = 0
44:13/0 = 0
45:13/0 = 0
46:13/0 = 0
47:13/0 = 0
48:13/0 = 0
0:14/0 = 0
1:14/0 = 0
2:14/0 = 0
3:14/0 = 0
4:14/0 = 0
5:14/0 = 0
6:14/0 = 0
7:14/0 = 0
8:14/0 = 0
9:14/0 = 0
10:14/0 = 0
11:14/0 = 0
12:14/0 = 0
13:14/0 = 0
14:14/0 = 0
15:14/0 = 0
16:14/0 = 0
17:14/0 = 0
18:14/0 = 0
19:14/0 = 0
20:14/0 = 0
21:14/0 = 0
22:14/0 = 0
23:14/0 = 0
```

```
24:14/0 = 0
25:14/0 = 0
26:14/0 = 0
27:14/0 = 0
28:14/0 = 0
29:14/0 = 0
30:14/0 = 0
31:14/0 = 0
32:14/0 = 0
33:14/0 = 0
34:14/0 = 0
35:14/0 = 0
36:14/0 = 0
37:14/0 = 0
38:14/0 = 0
39:14/0 = 0
40:14/0 = 0
41:14/0 = 0
42:14/0 = 0
43:14/0 = 0
44:14/0 = 0
45:14/0 = 0
46:14/0 = 0
47:14/0 = 0
48:14/0 = 0
0:15/0 = 0
1:15/0 = 0
2:15/0 = 0
3:15/0 = 0
4:15/0 = 0
5:15/0 = 0
6:15/0 = 0
7:15/0 = 0
```

```
8:15/0 = 0
9:15/0 = 0
10:15/0 = 0
11:15/0 = 0
12:15/0 = 0
13:15/0 = 0
14:15/0 = 0
15:15/0 = 0
16:15/0 = 0
17:15/0 = 0
18:15/0 = 0
19:15/0 = 0
20:15/0 = 0
21:15/0 = 0
22:15/0 = 0
23:15/0 = 0
24:15/0 = 0
25:15/0 = 0
26:15/0 = 0
27:15/0 = 0
28:15/0 = 0
29:15/0 = 0
30:15/0 = 0
31:15/0 = 0
32:15/0 = 0
33:15/0 = 0
34:15/0 = 0
35:15/0 = 0
36:15/0 = 0
37:15/0 = 0
38:15/0 = 0
39:15/0 = 0
40:15/0 = 0
```

```
41:15/0 = 0
42:15/0 = 0
43:15/0 = 0
44:15/0 = 0
45:15/0 = 0
46:15/0 = 0
47:15/0 = 0
48:15/0 = 0
0:16/0 = 0
1:16/0 = 0
2:16/0 = 0
3:16/0 = 0
4:16/0 = 0
5:16/0 = 0
6:16/0 = 0
7:16/0 = 0
8:16/0 = 0
9:16/0 = 0
10:16/0 = 0
11:16/0 = 0
12:16/0 = 0
13:16/0 = 0
14:16/0 = 0
15:16/0 = 0
16:16/0 = 0
17:16/0 = 0
18:16/0 = 0
19:16/0 = 0
20:16/0 = 0
21:16/0 = 0
22:16/0 = 0
23:16/0 = 0
24:16/0 = 0
```

```
25:16/0 = 0
26:16/0 = 0
27:16/0 = 0
28:16/0 = 0
29:16/0 = 0
30:16/0 = 0
31:16/0 = 0
32:16/0 = 0
33:16/0 = 0
34:16/0 = 0
35:16/0 = 0
36:16/0 = 0
37:16/0 = 0
38:16/0 = 0
39:16/0 = 0
40:16/0 = 0
41:16/0 = 0
42:16/0 = 0
43:16/0 = 0
44:16/0 = 0
45:16/0 = 0
46:16/0 = 0
47:16/0 = 0
48:16/0 = 0
0:17/0 = 0
1:17/0 = 0
2:17/0 = 0
3:17/0 = 0
4:17/0 = 0
5:17/0 = 0
6:17/0 = 0
7:17/0 = 0
8:17/0 = 0
```

```
9:17/0 = 0
10:17/0 = 0
11:17/0 = 0
12:17/0 = 0
13:17/0 = 0
14:17/0 = 0
15:17/0 = 0
16:17/0 = 0
17:17/0 = 0
18:17/0 = 0
19:17/0 = 0
20:17/0 = 0
21:17/0 = 0
22:17/0 = 0
23:17/0 = 0
24:17/0 = 0
25:17/0 = 0
26:17/0 = 0
27:17/0 = 0
28:17/0 = 0
29:17/0 = 0
30:17/0 = 0
31:17/0 = 0
32:17/0 = 0
33:17/0 = 0
34:17/0 = 0
35:17/0 = 0
36:17/0 = 0
37:17/0 = 0
38:17/0 = 0
39:17/0 = 0
40:17/0 = 0
41:17/0 = 0
```

```
42:17/0 = 0
43:17/0 = 0
44:17/0 = 0
45:17/0 = 0
46:17/0 = 0
47:17/0 = 0
48:17/0 = 0
0:18/0 = 0
1:18/0 = 0
2:18/0 = 0
3:18/0 = 0
4:18/0 = 0
5:18/0 = 0
6:18/0 = 0
7:18/0 = 0
8:18/0 = 0
9:18/0 = 0
10:18/0 = 0
11:18/0 = 0
12:18/0 = 0
13:18/0 = 0
14:18/0 = 0
15:18/0 = 0
16:18/0 = 0
17:18/0 = 0
18:18/0 = 0
19:18/0 = 0
20:18/0 = 0
21:18/0 = 0
22:18/0 = 0
23:18/0 = 0
24:18/0 = 0
25:18/0 = 0
```

```
26:18/0 = 0
27:18/0 = 0
28:18/0 = 0
29:18/0 = 0
30:18/0 = 0
31:18/0 = 0
32:18/0 = 0
33:18/0 = 0
34:18/0 = 0
35:18/0 = 0
36:18/0 = 0
37:18/0 = 0
38:18/0 = 0
39:18/0 = 0
40:18/0 = 0
41:18/0 = 0
42:18/0 = 0
43:18/0 = 0
44:18/0 = 0
45:18/0 = 0
46:18/0 = 0
47:18/0 = 0
48:18/0 = 0
0:19/0 = 0
1:19/0 = 0
2:19/0 = 0
3:19/0 = 0
4:19/0 = 0
5:19/0 = 0
6:19/0 = 0
7:19/0 = 0
8:19/0 = 0
9:19/0 = 0
```

```
10:19/0 = 0
11:19/0 = 0
12:19/0 = 0
13:19/0 = 0
14:19/0 = 0
15:19/0 = 0
16:19/0 = 0
17:19/0 = 0
18:19/0 = 0
19:19/0 = 0
20:19/0 = 0
21:19/0 = 0
22:19/0 = 0
23:19/0 = 0
24:19/0 = 0
25:19/0 = 0
26:19/0 = 0
27:19/0 = 0
28:19/0 = 0
29:19/0 = 0
30:19/0 = 0
31:19/0 = 0
32:19/0 = 0
33:19/0 = 0
34:19/0 = 0
35:19/0 = 0
36:19/0 = 0
37:19/0 = 0
38:19/0 = 0
39:19/0 = 0
40:19/0 = 0
41:19/0 = 0
42:19/0 = 0
```

```
43:19/0 = 0
44:19/0 = 0
45:19/0 = 0
46:19/0 = 0
47:19/0 = 0
48:19/0 = 0
0:20/0 = 0
1:20/0 = 0
2:20/0 = 0
3:20/0 = 0
4:20/0 = 0
5:20/0 = 0
6:20/0 = 0
7:20/0 = 0
8:20/0 = 0
9:20/0 = 0
10:20/0 = 0
11:20/0 = 0
12:20/0 = 0
13:20/0 = 0
14:20/0 = 0
15:20/0 = 0
16:20/0 = 0
17:20/0 = 0
18:20/0 = 0
19:20/0 = 0
20:20/0 = 0
21:20/0 = 0
22:20/0 = 0
23:20/0 = 0
24:20/0 = 0
25:20/0 = 0
26:20/0 = 0
```

```
27:20/0 = 0
28:20/0 = 0
29:20/0 = 0
30:20/0 = 0
31:20/0 = 0
32:20/0 = 0
33:20/0 = 0
34:20/0 = 0
35:20/0 = 0
36:20/0 = 0
37:20/0 = 0
38:20/0 = 0
39:20/0 = 0
40:20/0 = 0
41:20/0 = 0
42:20/0 = 0
43:20/0 = 0
44:20/0 = 0
45:20/0 = 0
46:20/0 = 0
47:20/0 = 0
48:20/0 = 0
0:21/0 = 0
1:21/0 = 0
2:21/0 = 0
3:21/0 = 0
4:21/0 = 0
5:21/0 = 0
6:21/0 = 0
7:21/0 = 0
8:21/0 = 0
9:21/0 = 0
10:21/0 = 0
```

```
11:21/0 = 0
12:21/0 = 0
13:21/0 = 0
14:21/0 = 0
15:21/0 = 0
16:21/0 = 0
17:21/0 = 0
18:21/0 = 0
19:21/0 = 0
20:21/0 = 0
21:21/0 = 0
22:21/0 = 0
23:21/0 = 0
24:21/0 = 0
25:21/0 = 0
26:21/0 = 0
27:21/0 = 0
28:21/0 = 0
29:21/0 = 0
30:21/0 = 0
31:21/0 = 0
32:21/0 = 0
33:21/0 = 0
34:21/0 = 0
35:21/0 = 0
36:21/0 = 0
37:21/0 = 0
38:21/0 = 0
39:21/0 = 0
40:21/0 = 0
41:21/0 = 0
42:21/0 = 0
43:21/0 = 0
```

```
44:21/0 = 0
45:21/0 = 0
46:21/0 = 0
47:21/0 = 0
48:21/0 = 0

[sub_resource type="TileSet" id="TileSet_3drs5"]
sources/0 = SubResource("TileSetAtlasSource_6h0bd")

[node name="TileMap" type="TileMap"]
tile_set = SubResource("TileSet_3drs5")
format = 2
script = ExtResource("2_1f4lw")
```

C.4.5 tile_map.gd

```
extends TileMap

var points: Array[Dictionary] = []
const EUCLIDEAN: String = "Euclidean distance"
const MANHATTAN: String = "Manhattan distance"
@export_enum(EUCLIDEAN, MANHATTAN) var distance: String = MANHATTAN
@export_range(2, 6, 1) var random_starting_points: int = 4
var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
   randomize()
   for x in range(-50, x_tile_range + 50):
        for y in range(-50, y_tile_range + 50):
            set_cell(0, Vector2(x, y), 0, Vector2(0, 0))
    var start_time: float = Time.get_ticks_msec()
    define_points(random_starting_points)
    paint_points()
    var new_time: float = Time.get_ticks_msec() - start_time
    print("Time taken: " + str(new_time) + "ms")

func paint_points() -> void:
    for point in points:
        set_cell(0, Vector2(point["x"], point["y"]), 0, point["type"
            ])
        for citizen in point["citizens"]:
            set_cell(0, Vector2(point["x"] + citizen["dx"], point["y"]
                + citizen["dy"]), 0, point["type"])

func _squared(x: int) -> int:
    return x ** 2

func calculate_points_delta(x: int, y: int, p: int) -> float:
    if distance == EUCLIDEAN:
        return sqrt(_squared(points[p]["x"] - x) + _squared(points[p
            ]["y"] - y))
    return abs(points[p]["x"] - x) + abs(points[p]["y"] - y)

func define_points(num_points: int) -> void:
    var types: Array[Vector2i] = [Vector2i(0,1),Vector2i(1,1),
        Vector2i(2,1),Vector2i(5,1),Vector2i(6,1),Vector2i(4,2)]
    for i in range(num_points):
        var x: int = randi_range(0, x_tile_range)
        var y: int = randi_range(0, y_tile_range)
        var type: Vector2i = types.pick_random()
        types.erase(type)
        points.append(
            {
                "type": type,
                "x": x,
                "y": y,
                "citizens": []
            }
        )
    for x in range(x_tile_range):
        for y in range(y_tile_range):
            var lowest_delta: Dictionary = {
                "point_id": 0,
                "delta": x_tile_range * y_tile_range
            }
            for p in range(len(points)):
                var delta: float = calculate_points_delta(x, y, p)
                if delta < lowest_delta["delta"]:
                    lowest_delta = {
                        "point_id": p,
                        "delta": delta
                    }
                var active_point: Dictionary = points[lowest_delta["point_id"]]
                var dx: int = x - active_point["x"]
                var dy: int = y - active_point["y"]
                active_point["citizens"].append(
                    {
                        "dx": dx,
                        "dy": dy
                    }
                )
```

C.4.6 icon.svg.import

```
[remap]

importer="texture"
type="CompressedTexture2D"
uid="uid://du4v6taw8ssax"
path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
metadata={
"vram_texture": false
}

[deps]

source_file="res://icon.svg"
dest_files=["res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
```

```
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
svg/scale=1.0
editor/scale_with_editor_scale=false
editor/convert_colors_with_editor_theme=false
```

C.4.7 monochrome_packed.png.import

```
[remap]

importer="texture"
type="CompressedTexture2D"
uid="uid://cpign73sfbsrt"
path="res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"
metadata={
"vram_texture": false
}

[deps]

source_file="res://monochrome_packed.png"
dest_files=["res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"]

[params]

compress/mode=0
compress/high_quality=false
```

```
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
```

C.5 PoissonGD4

C.5.1 .gitattributes

```
# Normalize EOL for all files that Git considers text files.
* text=auto eol=lf
```

C.5.2 .gitignore

```
# Godot 4+ specific ignores
.godot/
```

C.5.3 project.godot

```
; Engine configuration file.
; It's best edited using the editor UI and not directly,
; since the parameters that go here are not all obvious.
;
; Format:
;   [section] ; section goes between []
;   param=value ; assign values to parameters

config_version=5

[application]

config/name="Poisson Sampling Project"
run/main_scene="res://tile_map.tscn"
config/features=PackedStringArray("4.0", "Forward Plus")
config/icon="res://icon.svg"

[display]

window/size/viewport_height=640

[rendering]

renderer/rendering_method="gl_compatibility"
```

C.5.4 tile_map.tscn

```
[gd_scene load_steps=5 format=3 uid="uid://f2kv7fettdo7"]

[ext_resource type="Texture2D" uid="uid://c3bpsm4r8t504" path="res
    ://monochrome_packed.png" id="1_uucm3"]
[ext_resource type="Script" path="res://tile_map.gd" id="2_iyhvf"]
```

```

[sub_resource type="TileSetAtlasSource" id="
    TileSetAtlasSource_j4usm"]
texture = ExtResource("1_uucm3")
0:0/0 = 0
1:0/0 = 0
2:0/0 = 0
3:0/0 = 0
4:0/0 = 0
5:0/0 = 0
6:0/0 = 0
7:0/0 = 0
8:0/0 = 0
9:0/0 = 0
10:0/0 = 0
11:0/0 = 0
12:0/0 = 0
13:0/0 = 0
14:0/0 = 0
15:0/0 = 0
16:0/0 = 0
17:0/0 = 0
18:0/0 = 0
19:0/0 = 0
20:0/0 = 0
21:0/0 = 0
22:0/0 = 0
23:0/0 = 0
24:0/0 = 0
25:0/0 = 0
26:0/0 = 0
27:0/0 = 0
28:0/0 = 0
```

```
29:0/0 = 0
30:0/0 = 0
31:0/0 = 0
32:0/0 = 0
33:0/0 = 0
34:0/0 = 0
35:0/0 = 0
36:0/0 = 0
37:0/0 = 0
38:0/0 = 0
39:0/0 = 0
40:0/0 = 0
41:0/0 = 0
42:0/0 = 0
43:0/0 = 0
44:0/0 = 0
45:0/0 = 0
46:0/0 = 0
47:0/0 = 0
48:0/0 = 0
0:1/0 = 0
1:1/0 = 0
2:1/0 = 0
3:1/0 = 0
4:1/0 = 0
5:1/0 = 0
6:1/0 = 0
7:1/0 = 0
8:1/0 = 0
9:1/0 = 0
10:1/0 = 0
11:1/0 = 0
12:1/0 = 0
```

```
13:1/0 = 0
14:1/0 = 0
15:1/0 = 0
16:1/0 = 0
17:1/0 = 0
18:1/0 = 0
19:1/0 = 0
20:1/0 = 0
21:1/0 = 0
22:1/0 = 0
23:1/0 = 0
24:1/0 = 0
25:1/0 = 0
26:1/0 = 0
27:1/0 = 0
28:1/0 = 0
29:1/0 = 0
30:1/0 = 0
31:1/0 = 0
32:1/0 = 0
33:1/0 = 0
34:1/0 = 0
35:1/0 = 0
36:1/0 = 0
37:1/0 = 0
38:1/0 = 0
39:1/0 = 0
40:1/0 = 0
41:1/0 = 0
42:1/0 = 0
43:1/0 = 0
44:1/0 = 0
45:1/0 = 0
```

```
46:1/0 = 0
47:1/0 = 0
48:1/0 = 0
0:2/0 = 0
1:2/0 = 0
2:2/0 = 0
3:2/0 = 0
4:2/0 = 0
5:2/0 = 0
6:2/0 = 0
7:2/0 = 0
8:2/0 = 0
9:2/0 = 0
10:2/0 = 0
11:2/0 = 0
12:2/0 = 0
13:2/0 = 0
14:2/0 = 0
15:2/0 = 0
16:2/0 = 0
17:2/0 = 0
18:2/0 = 0
19:2/0 = 0
20:2/0 = 0
21:2/0 = 0
22:2/0 = 0
23:2/0 = 0
24:2/0 = 0
25:2/0 = 0
26:2/0 = 0
27:2/0 = 0
28:2/0 = 0
29:2/0 = 0
```

```
30:2/0 = 0
31:2/0 = 0
32:2/0 = 0
33:2/0 = 0
34:2/0 = 0
35:2/0 = 0
36:2/0 = 0
37:2/0 = 0
38:2/0 = 0
39:2/0 = 0
40:2/0 = 0
41:2/0 = 0
42:2/0 = 0
43:2/0 = 0
44:2/0 = 0
45:2/0 = 0
46:2/0 = 0
47:2/0 = 0
48:2/0 = 0
0:3/0 = 0
1:3/0 = 0
2:3/0 = 0
3:3/0 = 0
4:3/0 = 0
5:3/0 = 0
6:3/0 = 0
7:3/0 = 0
8:3/0 = 0
9:3/0 = 0
10:3/0 = 0
11:3/0 = 0
12:3/0 = 0
13:3/0 = 0
```

```
14:3/0 = 0
15:3/0 = 0
16:3/0 = 0
17:3/0 = 0
18:3/0 = 0
19:3/0 = 0
20:3/0 = 0
21:3/0 = 0
22:3/0 = 0
23:3/0 = 0
24:3/0 = 0
25:3/0 = 0
26:3/0 = 0
27:3/0 = 0
28:3/0 = 0
29:3/0 = 0
30:3/0 = 0
31:3/0 = 0
32:3/0 = 0
33:3/0 = 0
34:3/0 = 0
35:3/0 = 0
36:3/0 = 0
37:3/0 = 0
38:3/0 = 0
39:3/0 = 0
40:3/0 = 0
41:3/0 = 0
42:3/0 = 0
43:3/0 = 0
44:3/0 = 0
45:3/0 = 0
46:3/0 = 0
```

```
47:3/0 = 0
48:3/0 = 0
0:4/0 = 0
1:4/0 = 0
2:4/0 = 0
3:4/0 = 0
4:4/0 = 0
5:4/0 = 0
6:4/0 = 0
7:4/0 = 0
8:4/0 = 0
9:4/0 = 0
10:4/0 = 0
11:4/0 = 0
12:4/0 = 0
13:4/0 = 0
14:4/0 = 0
15:4/0 = 0
16:4/0 = 0
17:4/0 = 0
18:4/0 = 0
19:4/0 = 0
20:4/0 = 0
21:4/0 = 0
22:4/0 = 0
23:4/0 = 0
24:4/0 = 0
25:4/0 = 0
26:4/0 = 0
27:4/0 = 0
28:4/0 = 0
29:4/0 = 0
30:4/0 = 0
```

```
31:4/0 = 0
32:4/0 = 0
33:4/0 = 0
34:4/0 = 0
35:4/0 = 0
36:4/0 = 0
37:4/0 = 0
38:4/0 = 0
39:4/0 = 0
40:4/0 = 0
41:4/0 = 0
42:4/0 = 0
43:4/0 = 0
44:4/0 = 0
45:4/0 = 0
46:4/0 = 0
47:4/0 = 0
48:4/0 = 0
0:5/0 = 0
1:5/0 = 0
2:5/0 = 0
3:5/0 = 0
4:5/0 = 0
5:5/0 = 0
6:5/0 = 0
7:5/0 = 0
8:5/0 = 0
9:5/0 = 0
10:5/0 = 0
11:5/0 = 0
12:5/0 = 0
13:5/0 = 0
14:5/0 = 0
```

```
15:5/0 = 0
16:5/0 = 0
17:5/0 = 0
18:5/0 = 0
19:5/0 = 0
20:5/0 = 0
21:5/0 = 0
22:5/0 = 0
23:5/0 = 0
24:5/0 = 0
25:5/0 = 0
26:5/0 = 0
27:5/0 = 0
28:5/0 = 0
29:5/0 = 0
30:5/0 = 0
31:5/0 = 0
32:5/0 = 0
33:5/0 = 0
34:5/0 = 0
35:5/0 = 0
36:5/0 = 0
37:5/0 = 0
38:5/0 = 0
39:5/0 = 0
40:5/0 = 0
41:5/0 = 0
42:5/0 = 0
43:5/0 = 0
44:5/0 = 0
45:5/0 = 0
46:5/0 = 0
47:5/0 = 0
```

```
48:5/0 = 0
0:6/0 = 0
1:6/0 = 0
2:6/0 = 0
3:6/0 = 0
4:6/0 = 0
5:6/0 = 0
6:6/0 = 0
7:6/0 = 0
8:6/0 = 0
9:6/0 = 0
10:6/0 = 0
11:6/0 = 0
12:6/0 = 0
13:6/0 = 0
14:6/0 = 0
15:6/0 = 0
16:6/0 = 0
17:6/0 = 0
18:6/0 = 0
19:6/0 = 0
20:6/0 = 0
21:6/0 = 0
22:6/0 = 0
23:6/0 = 0
24:6/0 = 0
25:6/0 = 0
26:6/0 = 0
27:6/0 = 0
28:6/0 = 0
29:6/0 = 0
30:6/0 = 0
31:6/0 = 0
```

```
32:6/0 = 0
33:6/0 = 0
34:6/0 = 0
35:6/0 = 0
36:6/0 = 0
37:6/0 = 0
38:6/0 = 0
39:6/0 = 0
40:6/0 = 0
41:6/0 = 0
42:6/0 = 0
43:6/0 = 0
44:6/0 = 0
45:6/0 = 0
46:6/0 = 0
47:6/0 = 0
48:6/0 = 0
0:7/0 = 0
1:7/0 = 0
2:7/0 = 0
3:7/0 = 0
4:7/0 = 0
5:7/0 = 0
6:7/0 = 0
7:7/0 = 0
8:7/0 = 0
9:7/0 = 0
10:7/0 = 0
11:7/0 = 0
12:7/0 = 0
13:7/0 = 0
14:7/0 = 0
15:7/0 = 0
```

```
16:7/0 = 0
17:7/0 = 0
18:7/0 = 0
19:7/0 = 0
20:7/0 = 0
21:7/0 = 0
22:7/0 = 0
23:7/0 = 0
24:7/0 = 0
25:7/0 = 0
26:7/0 = 0
27:7/0 = 0
28:7/0 = 0
29:7/0 = 0
30:7/0 = 0
31:7/0 = 0
32:7/0 = 0
33:7/0 = 0
34:7/0 = 0
35:7/0 = 0
36:7/0 = 0
37:7/0 = 0
38:7/0 = 0
39:7/0 = 0
40:7/0 = 0
41:7/0 = 0
42:7/0 = 0
43:7/0 = 0
44:7/0 = 0
45:7/0 = 0
46:7/0 = 0
47:7/0 = 0
48:7/0 = 0
```

```
0:8/0 = 0
1:8/0 = 0
2:8/0 = 0
3:8/0 = 0
4:8/0 = 0
5:8/0 = 0
6:8/0 = 0
7:8/0 = 0
8:8/0 = 0
9:8/0 = 0
10:8/0 = 0
11:8/0 = 0
12:8/0 = 0
13:8/0 = 0
14:8/0 = 0
15:8/0 = 0
16:8/0 = 0
17:8/0 = 0
18:8/0 = 0
19:8/0 = 0
20:8/0 = 0
21:8/0 = 0
22:8/0 = 0
23:8/0 = 0
24:8/0 = 0
25:8/0 = 0
26:8/0 = 0
27:8/0 = 0
28:8/0 = 0
29:8/0 = 0
30:8/0 = 0
31:8/0 = 0
32:8/0 = 0
```

```
33:8/0 = 0
34:8/0 = 0
35:8/0 = 0
36:8/0 = 0
37:8/0 = 0
38:8/0 = 0
39:8/0 = 0
40:8/0 = 0
41:8/0 = 0
42:8/0 = 0
43:8/0 = 0
44:8/0 = 0
45:8/0 = 0
46:8/0 = 0
47:8/0 = 0
48:8/0 = 0
0:9/0 = 0
1:9/0 = 0
2:9/0 = 0
3:9/0 = 0
4:9/0 = 0
5:9/0 = 0
6:9/0 = 0
7:9/0 = 0
8:9/0 = 0
9:9/0 = 0
10:9/0 = 0
11:9/0 = 0
12:9/0 = 0
13:9/0 = 0
14:9/0 = 0
15:9/0 = 0
16:9/0 = 0
```

```
17:9/0 = 0
18:9/0 = 0
19:9/0 = 0
20:9/0 = 0
21:9/0 = 0
22:9/0 = 0
23:9/0 = 0
24:9/0 = 0
25:9/0 = 0
26:9/0 = 0
27:9/0 = 0
28:9/0 = 0
29:9/0 = 0
30:9/0 = 0
31:9/0 = 0
32:9/0 = 0
33:9/0 = 0
34:9/0 = 0
35:9/0 = 0
36:9/0 = 0
37:9/0 = 0
38:9/0 = 0
39:9/0 = 0
40:9/0 = 0
41:9/0 = 0
42:9/0 = 0
43:9/0 = 0
44:9/0 = 0
45:9/0 = 0
46:9/0 = 0
47:9/0 = 0
48:9/0 = 0
0:10/0 = 0
```

```
1:10/0 = 0
2:10/0 = 0
3:10/0 = 0
4:10/0 = 0
5:10/0 = 0
6:10/0 = 0
7:10/0 = 0
8:10/0 = 0
9:10/0 = 0
10:10/0 = 0
11:10/0 = 0
12:10/0 = 0
13:10/0 = 0
14:10/0 = 0
15:10/0 = 0
16:10/0 = 0
17:10/0 = 0
18:10/0 = 0
19:10/0 = 0
20:10/0 = 0
21:10/0 = 0
22:10/0 = 0
23:10/0 = 0
24:10/0 = 0
25:10/0 = 0
26:10/0 = 0
27:10/0 = 0
28:10/0 = 0
29:10/0 = 0
30:10/0 = 0
31:10/0 = 0
32:10/0 = 0
33:10/0 = 0
```

```
34:10/0 = 0
35:10/0 = 0
36:10/0 = 0
37:10/0 = 0
38:10/0 = 0
39:10/0 = 0
40:10/0 = 0
41:10/0 = 0
42:10/0 = 0
43:10/0 = 0
44:10/0 = 0
45:10/0 = 0
46:10/0 = 0
47:10/0 = 0
48:10/0 = 0
0:11/0 = 0
1:11/0 = 0
2:11/0 = 0
3:11/0 = 0
4:11/0 = 0
5:11/0 = 0
6:11/0 = 0
7:11/0 = 0
8:11/0 = 0
9:11/0 = 0
10:11/0 = 0
11:11/0 = 0
12:11/0 = 0
13:11/0 = 0
14:11/0 = 0
15:11/0 = 0
16:11/0 = 0
17:11/0 = 0
```

```
18:11/0 = 0
19:11/0 = 0
20:11/0 = 0
21:11/0 = 0
22:11/0 = 0
23:11/0 = 0
24:11/0 = 0
25:11/0 = 0
26:11/0 = 0
27:11/0 = 0
28:11/0 = 0
29:11/0 = 0
30:11/0 = 0
31:11/0 = 0
32:11/0 = 0
33:11/0 = 0
34:11/0 = 0
35:11/0 = 0
36:11/0 = 0
37:11/0 = 0
38:11/0 = 0
39:11/0 = 0
40:11/0 = 0
41:11/0 = 0
42:11/0 = 0
43:11/0 = 0
44:11/0 = 0
45:11/0 = 0
46:11/0 = 0
47:11/0 = 0
48:11/0 = 0
0:12/0 = 0
1:12/0 = 0
```

```
2:12/0 = 0
3:12/0 = 0
4:12/0 = 0
5:12/0 = 0
6:12/0 = 0
7:12/0 = 0
8:12/0 = 0
9:12/0 = 0
10:12/0 = 0
11:12/0 = 0
12:12/0 = 0
13:12/0 = 0
14:12/0 = 0
15:12/0 = 0
16:12/0 = 0
17:12/0 = 0
18:12/0 = 0
19:12/0 = 0
20:12/0 = 0
21:12/0 = 0
22:12/0 = 0
23:12/0 = 0
24:12/0 = 0
25:12/0 = 0
26:12/0 = 0
27:12/0 = 0
28:12/0 = 0
29:12/0 = 0
30:12/0 = 0
31:12/0 = 0
32:12/0 = 0
33:12/0 = 0
34:12/0 = 0
```

```
35:12/0 = 0
36:12/0 = 0
37:12/0 = 0
38:12/0 = 0
39:12/0 = 0
40:12/0 = 0
41:12/0 = 0
42:12/0 = 0
43:12/0 = 0
44:12/0 = 0
45:12/0 = 0
46:12/0 = 0
47:12/0 = 0
48:12/0 = 0
0:13/0 = 0
1:13/0 = 0
2:13/0 = 0
3:13/0 = 0
4:13/0 = 0
5:13/0 = 0
6:13/0 = 0
7:13/0 = 0
8:13/0 = 0
9:13/0 = 0
10:13/0 = 0
11:13/0 = 0
12:13/0 = 0
13:13/0 = 0
14:13/0 = 0
15:13/0 = 0
16:13/0 = 0
17:13/0 = 0
18:13/0 = 0
```

```
19:13/0 = 0
20:13/0 = 0
21:13/0 = 0
22:13/0 = 0
23:13/0 = 0
24:13/0 = 0
25:13/0 = 0
26:13/0 = 0
27:13/0 = 0
28:13/0 = 0
29:13/0 = 0
30:13/0 = 0
31:13/0 = 0
32:13/0 = 0
33:13/0 = 0
34:13/0 = 0
35:13/0 = 0
36:13/0 = 0
37:13/0 = 0
38:13/0 = 0
39:13/0 = 0
40:13/0 = 0
41:13/0 = 0
42:13/0 = 0
43:13/0 = 0
44:13/0 = 0
45:13/0 = 0
46:13/0 = 0
47:13/0 = 0
48:13/0 = 0
0:14/0 = 0
1:14/0 = 0
2:14/0 = 0
```

```
3:14/0 = 0
4:14/0 = 0
5:14/0 = 0
6:14/0 = 0
7:14/0 = 0
8:14/0 = 0
9:14/0 = 0
10:14/0 = 0
11:14/0 = 0
12:14/0 = 0
13:14/0 = 0
14:14/0 = 0
15:14/0 = 0
16:14/0 = 0
17:14/0 = 0
18:14/0 = 0
19:14/0 = 0
20:14/0 = 0
21:14/0 = 0
22:14/0 = 0
23:14/0 = 0
24:14/0 = 0
25:14/0 = 0
26:14/0 = 0
27:14/0 = 0
28:14/0 = 0
29:14/0 = 0
30:14/0 = 0
31:14/0 = 0
32:14/0 = 0
33:14/0 = 0
34:14/0 = 0
35:14/0 = 0
```

```
36:14/0 = 0
37:14/0 = 0
38:14/0 = 0
39:14/0 = 0
40:14/0 = 0
41:14/0 = 0
42:14/0 = 0
43:14/0 = 0
44:14/0 = 0
45:14/0 = 0
46:14/0 = 0
47:14/0 = 0
48:14/0 = 0
0:15/0 = 0
1:15/0 = 0
2:15/0 = 0
3:15/0 = 0
4:15/0 = 0
5:15/0 = 0
6:15/0 = 0
7:15/0 = 0
8:15/0 = 0
9:15/0 = 0
10:15/0 = 0
11:15/0 = 0
12:15/0 = 0
13:15/0 = 0
14:15/0 = 0
15:15/0 = 0
16:15/0 = 0
17:15/0 = 0
18:15/0 = 0
19:15/0 = 0
```

```
20:15/0 = 0
21:15/0 = 0
22:15/0 = 0
23:15/0 = 0
24:15/0 = 0
25:15/0 = 0
26:15/0 = 0
27:15/0 = 0
28:15/0 = 0
29:15/0 = 0
30:15/0 = 0
31:15/0 = 0
32:15/0 = 0
33:15/0 = 0
34:15/0 = 0
35:15/0 = 0
36:15/0 = 0
37:15/0 = 0
38:15/0 = 0
39:15/0 = 0
40:15/0 = 0
41:15/0 = 0
42:15/0 = 0
43:15/0 = 0
44:15/0 = 0
45:15/0 = 0
46:15/0 = 0
47:15/0 = 0
48:15/0 = 0
0:16/0 = 0
1:16/0 = 0
2:16/0 = 0
3:16/0 = 0
```

```
4:16/0 = 0
5:16/0 = 0
6:16/0 = 0
7:16/0 = 0
8:16/0 = 0
9:16/0 = 0
10:16/0 = 0
11:16/0 = 0
12:16/0 = 0
13:16/0 = 0
14:16/0 = 0
15:16/0 = 0
16:16/0 = 0
17:16/0 = 0
18:16/0 = 0
19:16/0 = 0
20:16/0 = 0
21:16/0 = 0
22:16/0 = 0
23:16/0 = 0
24:16/0 = 0
25:16/0 = 0
26:16/0 = 0
27:16/0 = 0
28:16/0 = 0
29:16/0 = 0
30:16/0 = 0
31:16/0 = 0
32:16/0 = 0
33:16/0 = 0
34:16/0 = 0
35:16/0 = 0
36:16/0 = 0
```

```
37:16/0 = 0
38:16/0 = 0
39:16/0 = 0
40:16/0 = 0
41:16/0 = 0
42:16/0 = 0
43:16/0 = 0
44:16/0 = 0
45:16/0 = 0
46:16/0 = 0
47:16/0 = 0
48:16/0 = 0
0:17/0 = 0
1:17/0 = 0
2:17/0 = 0
3:17/0 = 0
4:17/0 = 0
5:17/0 = 0
6:17/0 = 0
7:17/0 = 0
8:17/0 = 0
9:17/0 = 0
10:17/0 = 0
11:17/0 = 0
12:17/0 = 0
13:17/0 = 0
14:17/0 = 0
15:17/0 = 0
16:17/0 = 0
17:17/0 = 0
18:17/0 = 0
19:17/0 = 0
20:17/0 = 0
```

```
21:17/0 = 0
22:17/0 = 0
23:17/0 = 0
24:17/0 = 0
25:17/0 = 0
26:17/0 = 0
27:17/0 = 0
28:17/0 = 0
29:17/0 = 0
30:17/0 = 0
31:17/0 = 0
32:17/0 = 0
33:17/0 = 0
34:17/0 = 0
35:17/0 = 0
36:17/0 = 0
37:17/0 = 0
38:17/0 = 0
39:17/0 = 0
40:17/0 = 0
41:17/0 = 0
42:17/0 = 0
43:17/0 = 0
44:17/0 = 0
45:17/0 = 0
46:17/0 = 0
47:17/0 = 0
48:17/0 = 0
0:18/0 = 0
1:18/0 = 0
2:18/0 = 0
3:18/0 = 0
4:18/0 = 0
```

```
5:18/0 = 0
6:18/0 = 0
7:18/0 = 0
8:18/0 = 0
9:18/0 = 0
10:18/0 = 0
11:18/0 = 0
12:18/0 = 0
13:18/0 = 0
14:18/0 = 0
15:18/0 = 0
16:18/0 = 0
17:18/0 = 0
18:18/0 = 0
19:18/0 = 0
20:18/0 = 0
21:18/0 = 0
22:18/0 = 0
23:18/0 = 0
24:18/0 = 0
25:18/0 = 0
26:18/0 = 0
27:18/0 = 0
28:18/0 = 0
29:18/0 = 0
30:18/0 = 0
31:18/0 = 0
32:18/0 = 0
33:18/0 = 0
34:18/0 = 0
35:18/0 = 0
36:18/0 = 0
37:18/0 = 0
```

```
38:18/0 = 0
39:18/0 = 0
40:18/0 = 0
41:18/0 = 0
42:18/0 = 0
43:18/0 = 0
44:18/0 = 0
45:18/0 = 0
46:18/0 = 0
47:18/0 = 0
48:18/0 = 0
0:19/0 = 0
1:19/0 = 0
2:19/0 = 0
3:19/0 = 0
4:19/0 = 0
5:19/0 = 0
6:19/0 = 0
7:19/0 = 0
8:19/0 = 0
9:19/0 = 0
10:19/0 = 0
11:19/0 = 0
12:19/0 = 0
13:19/0 = 0
14:19/0 = 0
15:19/0 = 0
16:19/0 = 0
17:19/0 = 0
18:19/0 = 0
19:19/0 = 0
20:19/0 = 0
21:19/0 = 0
```

```
22:19/0 = 0
23:19/0 = 0
24:19/0 = 0
25:19/0 = 0
26:19/0 = 0
27:19/0 = 0
28:19/0 = 0
29:19/0 = 0
30:19/0 = 0
31:19/0 = 0
32:19/0 = 0
33:19/0 = 0
34:19/0 = 0
35:19/0 = 0
36:19/0 = 0
37:19/0 = 0
38:19/0 = 0
39:19/0 = 0
40:19/0 = 0
41:19/0 = 0
42:19/0 = 0
43:19/0 = 0
44:19/0 = 0
45:19/0 = 0
46:19/0 = 0
47:19/0 = 0
48:19/0 = 0
0:20/0 = 0
1:20/0 = 0
2:20/0 = 0
3:20/0 = 0
4:20/0 = 0
5:20/0 = 0
```

```
6:20/0 = 0
7:20/0 = 0
8:20/0 = 0
9:20/0 = 0
10:20/0 = 0
11:20/0 = 0
12:20/0 = 0
13:20/0 = 0
14:20/0 = 0
15:20/0 = 0
16:20/0 = 0
17:20/0 = 0
18:20/0 = 0
19:20/0 = 0
20:20/0 = 0
21:20/0 = 0
22:20/0 = 0
23:20/0 = 0
24:20/0 = 0
25:20/0 = 0
26:20/0 = 0
27:20/0 = 0
28:20/0 = 0
29:20/0 = 0
30:20/0 = 0
31:20/0 = 0
32:20/0 = 0
33:20/0 = 0
34:20/0 = 0
35:20/0 = 0
36:20/0 = 0
37:20/0 = 0
38:20/0 = 0
```

```
39:20/0 = 0
40:20/0 = 0
41:20/0 = 0
42:20/0 = 0
43:20/0 = 0
44:20/0 = 0
45:20/0 = 0
46:20/0 = 0
47:20/0 = 0
48:20/0 = 0
0:21/0 = 0
1:21/0 = 0
2:21/0 = 0
3:21/0 = 0
4:21/0 = 0
5:21/0 = 0
6:21/0 = 0
7:21/0 = 0
8:21/0 = 0
9:21/0 = 0
10:21/0 = 0
11:21/0 = 0
12:21/0 = 0
13:21/0 = 0
14:21/0 = 0
15:21/0 = 0
16:21/0 = 0
17:21/0 = 0
18:21/0 = 0
19:21/0 = 0
20:21/0 = 0
21:21/0 = 0
22:21/0 = 0
```

```
23:21/0 = 0
24:21/0 = 0
25:21/0 = 0
26:21/0 = 0
27:21/0 = 0
28:21/0 = 0
29:21/0 = 0
30:21/0 = 0
31:21/0 = 0
32:21/0 = 0
33:21/0 = 0
34:21/0 = 0
35:21/0 = 0
36:21/0 = 0
37:21/0 = 0
38:21/0 = 0
39:21/0 = 0
40:21/0 = 0
41:21/0 = 0
42:21/0 = 0
43:21/0 = 0
44:21/0 = 0
45:21/0 = 0
46:21/0 = 0
47:21/0 = 0
48:21/0 = 0

[sub_resource type="TileSet" id="TileSet_8pb5m"]
sources/0 = SubResource("TileSetAtlasSource_j4usm")

[node name="TileMap" type="TileMap"]
tile_set = SubResource("TileSet_8pb5m")
format = 2
```

```
script = ExtResource("2_iyhvf")
```

C.5.5 tile_map.gd

```
extends TileMap

var cell_points: Array[Vector2]
@export var point_radius: float = 1.0
@export var region_size: Vector2 = Vector2.ONE
@export var rejection_samples: int = 30

var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y

# Called when the node enters the scene tree for the first time.
func _ready():
   randomize()
   for x in range(-50, x_tile_range + 50):
      for y in range(-50, y_tile_range + 50):
         set_cell(0, Vector2(x, y), 0, Vector2(0, 0))
   cell_points = generate_points(point_radius, region_size,
       rejection_samples)

func generate_points(radius: float, sample_region_size: Vector2,
    number_of_samples_before_rejection: int = 30) -> Array[Vector2
    ]:
   var cell_size: float = radius / sqrt(2)
   var grid: Array[Array] = []
   var points: Array[Vector2] = []
   var spawn_points: Array[Vector2] = []
```

```

    spawn_points.append(sample_region_size/2)

    while spawn_points.size() > 0:
        var spawn_index: int = randi_range(0, spawn_points.size() - 1)
        var spawn_centre: Vector2 = spawn_points[spawn_index]
        var candidate_accepted: bool = false

        for i in range(number_of_samples_before_rejection):
            var angle: float = randf_range(0.0, 1.0) * TAU # TAU = PI * 2
            var direction: Vector2 = Vector2(sin(angle), cos(angle))
            var candidate: Vector2 = spawn_centre + direction * randf_range(radius, 2 * radius)
            if is_valid(candidate, sample_region_size, cell_size, radius, points, grid):
                points.append(candidate)
                spawn_points.append(candidate)
                grid[int(candidate.x/cell_size)][int(candidate.y/cell_size)] = len(points)
                candidate_accepted = true
                break

        if not candidate_accepted:
            spawn_points.remove_at(spawn_index)

    return points

func is_valid(candidate: Vector2, sample_region_size: Vector2, cell_size: float, radius: float, points: Array[Vector2], grid: Array[Array]):
    if candidate.x >= 0 and candidate.x < sample_region_size.x and
```

```
        candidate.y >= 0 and candidate.y < sample_region_size.y:
        var cell_x: int = candidate.x / cell_size
        var cell_y: int = candidate.y / cell_size
        var search_start_x: int = max(0, cell_x - 2)
        var search_end_x: int = min(cell_x + 2, x_tile_range - 1)
        var search_start_y: int = max(0, cell_y - 2)
        var search_end_y: int = min(cell_y + 2, y_tile_range - 1)
        for x in range(search_start_x, search_end_x):
            for y in range(search_start_y, search_end_y):
                var point_index: int = grid[x][y]
                if point_index != -1:
                    var distance: float = (candidate - points[
                        point_index]).length()
                    if distance < radius:
                        return false
        return true
    return false
```

C.5.6 icon.svg.import

```
[remap]

importer="texture"
type="CompressedTexture2D"
uid="uid://uotfe6soknht"
path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
metadata={
"vram_texture": false
}

[deps]
```

```

source_file="res://icon.svg"
dest_files=["res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
detect_3d/compress_to=1
svg/scale=1.0
editor/scale_with_editor_scale=false
editor/convert_colors_with_editor_theme=false
```

C.5.7 monochrome_packed.png.import

```
[remap]

```

```
importer="texture"
type="CompressedTexture2D"
uid="uid://c3bpsm4r8t504"
path="res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"
metadata={
"vram_texture": false
}

[deps]

source_file="res://monochrome_packed.png"
dest_files=["res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"]

[params]

compress/mode=0
compress/high_quality=false
compress/lossy_quality=0.7
compress/hdr_compression=1
compress/normal_map=0
compress/channel_pack=0
mipmaps/generate=false
mipmaps/limit=-1
roughness/mode=0
roughness/src_normal=""
process/fix_alpha_border=true
process/premult_alpha=false
process/normal_map_invert_y=false
process/hdr_as_srgb=false
process/hdr_clamp_exposure=false
process/size_limit=0
```

```
detect_3d/compress_to=1
```