



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

1	Introduction	3
1.1	Report Structure	3
2	Background	4
2.1	Procedural Generation: Background	4
2.2	Justifying My Choice of Engine: Godot	6
2.3	Justifying My Choice of Scenario: A 2D tile-map RPG-style roaming game . .	7
2.4	Justifying My Choice of Algorithms for the Above Scenario	8
3	Report Body	10
3.1	Algorithms	10
3.2	Implementations	19
4	Design & Specification	22
4.1	Section Heading	22
5	Implementation	23
5.1	Section Heading	23
6	Legal, Social, Ethical and Professional Issues	24
6.1	Section Heading	24
7	Results/Evaluation	25
7.1	Software Testing	25
7.2	Section Heading	25
8	Conclusion and Future Work	26
	Bibliography	28
A	Extra Information	29
A.1	Tables, proofs, graphs, test cases,	29
B	User Guide	30
B.1	Instructions	30

C	Source Code	31
C.1	Instructions	31
C.2	LSystemGrammarDemo	32
C.3	ProcGenRPG (L-System)	37
C.4	VoronoiCellsGD4	45
C.5	PoissonGD4	84

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, Voronoi 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 **Long-Term Support** 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 Terraria, 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 "DOL-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. Voronoi 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.[10] 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:

$$F, +, -$$

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:

$$F + F + F + F$$

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

$$F \rightarrow F + F - F - FF + F + F - F$$

We can represent this *parametric* L-system in the following form:[15]

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

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:



1

Code 3.1: Simple String Replacement for an L-System with 1 rule

The first 3 iterations of this operation are shown here:

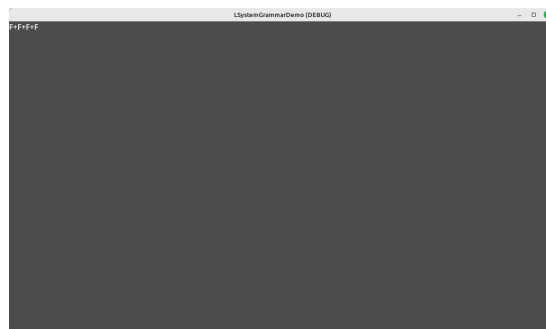


Figure 3.1: The axiom of the aforementioned simple L-System with just one rule. String size: 8.
Source: Own work.



Figure 3.2: The first iteration of the aforementioned simple L-System with just one rule. String size: 59. Source: Own work.

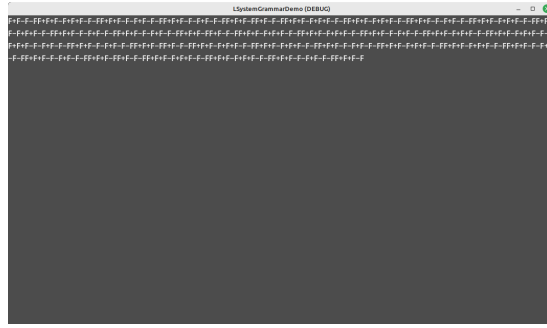


Figure 3.3: The second iteration of the aforementioned simple L-System with just one rule. String size: 475.
Source: Own work.

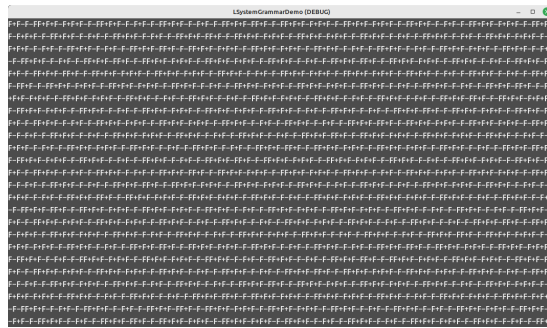


Figure 3.4: The third iteration of the aforementioned simple L-System with just one rule. String size: 3803. The string is too large to show in the window, as you can see here.
Source: Own work.

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

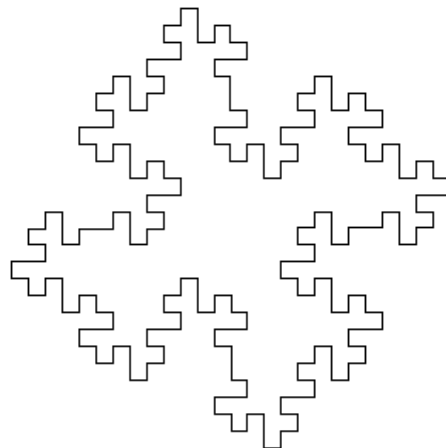


Figure 3.5: A lattice generated with the example grammar on a custom-written Classic Mac OS application specifically written for working with L-Systems.[2]

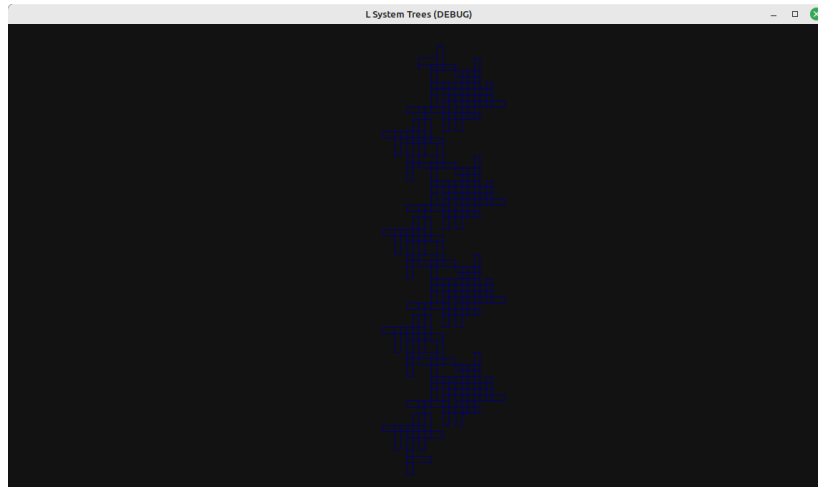


Figure 3.6: A lattice generated with the example grammar on a Godot project for drawing from L-Systems. Source: Initial project written by YouTuber Codat[3][4], and converted to Godot 4 (with the addition of the lattice grammar) by me.[5]

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 like so:



String replacement for an L-System with 2 rules

```

1  func get_new_replacement(character: String) -> String:
2      ^^Ifor rule in rules:
3          ^^I^^Iif rule["from"] == character:
4              ^^I^^I^^Ireturn rule["to"]
5          ^^Ireturn ""
6
7  func replace_string(string: String) -> String:
8      ^^Ivar new_string = ""
9      ^^Ifor character in string:
10         ^^I^^Inew_string += get_new_replacement(character)
11     ^^Ireturn new_string

```

Code 3.2: String replacement for an L-System with 2 rules

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:

$$a \rightarrow ab$$

$$b \rightarrow a$$

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

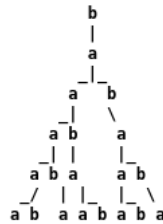


Figure 3.7: The first few steps of a derivation of our example grammar.[11]

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[12], as you can see in Figure 3.8.

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 3.9.[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[9], “it cuts low frequencies and emphasizes frequencies around and above the grid spacing.”

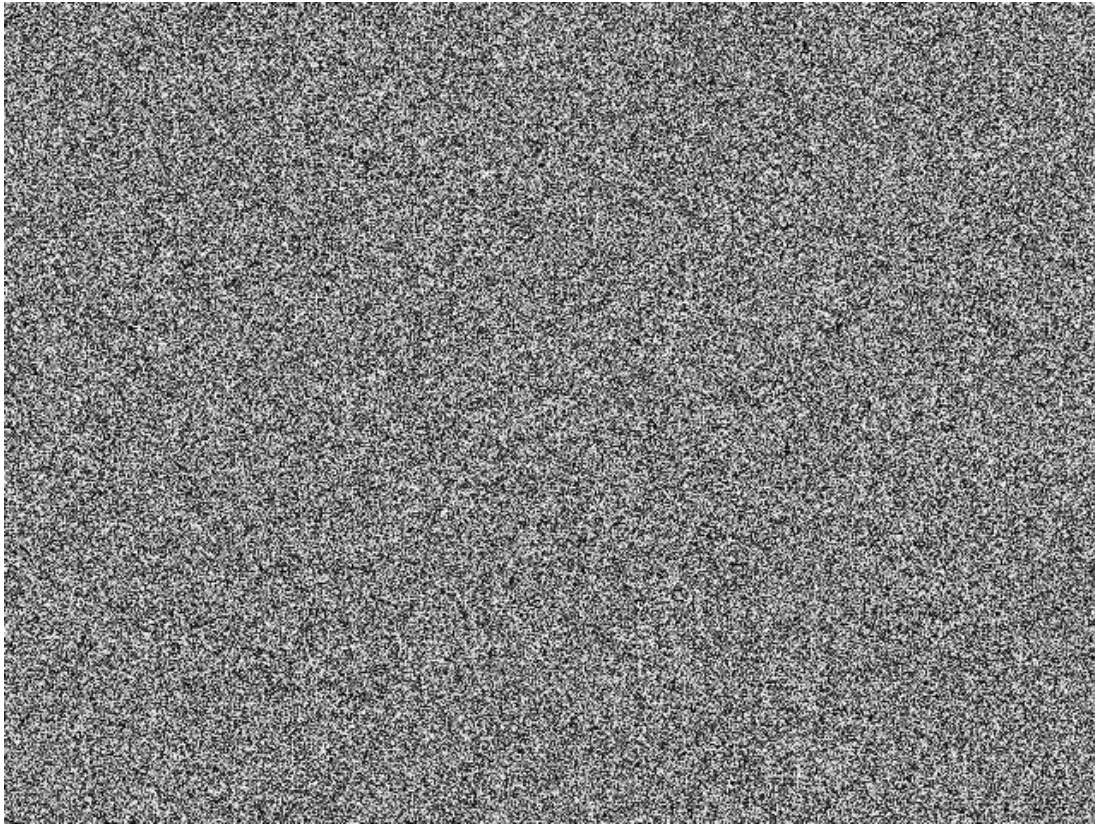
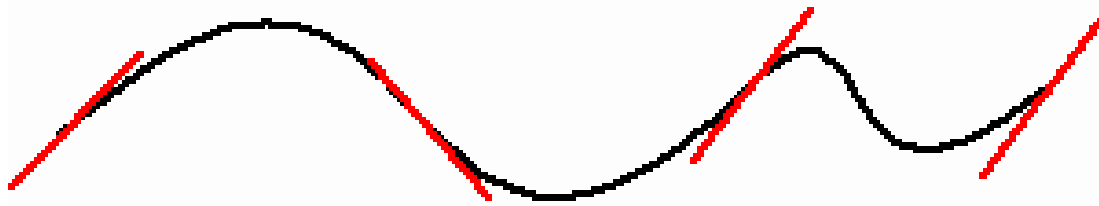


Figure 3.8: A white noise picture generated with Robson’s white noise image generator.[14]
Settings: 640 squares horizontally, 480 squares vertically, size of squares 1, colours greyscale, bias none.

Perlin Noise



Value Noise

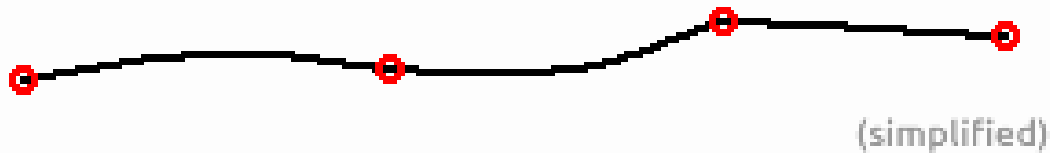


Figure 3.9: A comparison between the kinds of curves produced by Value noise interpolation and Perlin (and other Gradient) noise interpolation.[6]

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 filed a patent on his work in 2002 that was granted in 2005[13], which prompted the creation of the OpenSimplex noise algorithm[16] for free use; the patent has since expired in 2022, allowing free use to both Perlin and the original Simplex noise.[13]

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:

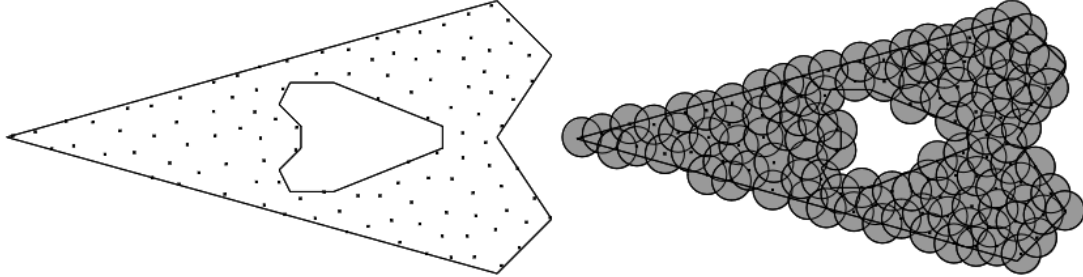


Figure 3.10: A diagram of a maximal Poisson disk distribution done on a concave plane, with the right side denoting maximality through the grey disks overlapping but not any points overlapping.[7]

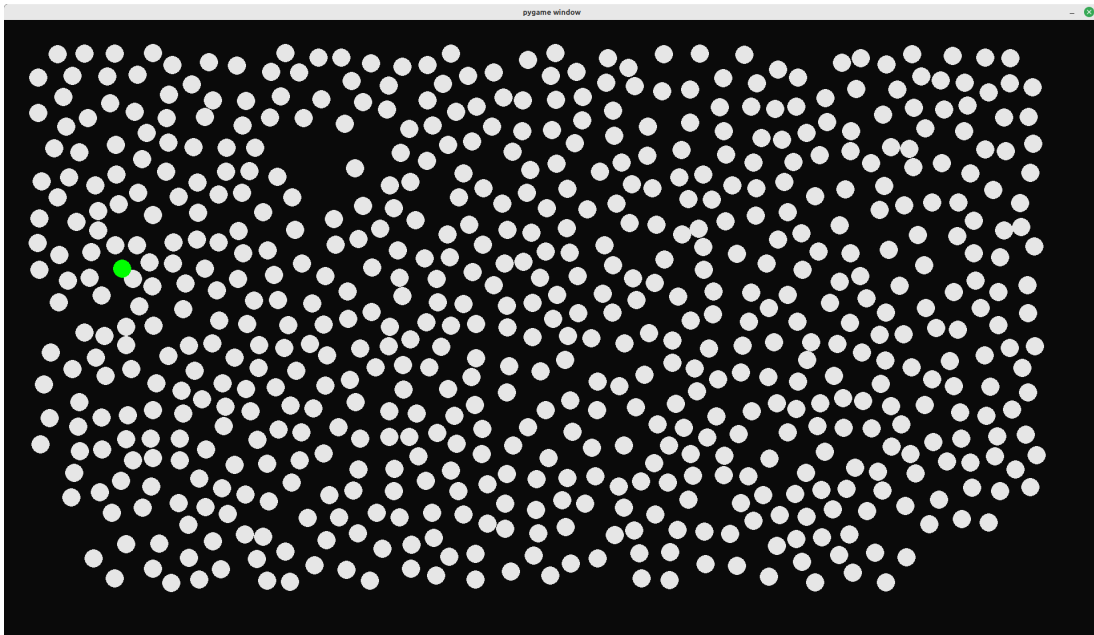


Figure 3.11: An implementation of Poisson disk sampling made in Pygame.[1] The screenshot was taken *after* all of the samples were taken.

3.1.4 Voronoï Cells

Named after the Ukranian mathematician Georgy Voronoy, Voronoi 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 Voronoi Diagram or Voronoi Tessellation.

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” tessellation than the Manhattan distance, the geometry of which is more “blocky” and resembles taxicabs (hence its alternate name “Taxicab Geometry”).

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 3.6. 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[11]:

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

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 3.7 and repeated below:

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

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

$$O \rightarrow OWO$$

$$W \rightarrow WB$$

$$B \rightarrow BWO$$

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.

3.2.3 Poisson Disk Sampling

3.2.4 Voronoï Cells

Chapter 4

Design & Specification

4.1 Section Heading

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).

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] Hernan J. González. "why is gradient noise better quality than value noise?" - stack exchange answer, August 2012. [Online, Accessed 16-April-2023, Stack Exchange Username "leonbloy"].
- [10] 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.

- [11] Gabriela Ochoa. An introduction to lindenmayer systems: D0l-system, February 1998. [Online; accessed 11-April-2023].
- [12] Guilherme Oliveira. "intro to procedural levels in godot 3.1: Perlin noise tutorial".
- [13] Kenneth H. Perlin. Standard for perlin noise. <https://patents.google.com/patent/US6867776>, January 2002. [Online patent document, Accessed 16-April-2023].
- [14] Robson. An interactive webpage for generating images that simulate white noise., April 2021. [Online, Accessed 16-April-2023].
- [15] Wikipedia contributors. L-system — Wikipedia, the free encyclopedia, 2022. [Online; accessed 23-February-2023].
- [16] 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

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

C.2.2 .gitignore

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

C.2.3 project.godot

```
1  ; Engine configuration file.
2  ; It's best edited using the editor UI and not directly,
3  ; since the parameters that go here are not all obvious.
4  ;
5  ; Format:
6  ;   [section] ; section goes between []
7  ;   param=value ; assign values to parameters
8
9  config_version=5
10
11  [application]
12
13  config/name="LSystemGrammarDemo"
14  run/main_scene="res://DemoNode.tscn"
15  config/features=PackedStringArray("4.0")
16
17  [display]
```

```

18
19 window/stretch/mode="canvas_items"
20 window/stretch/aspect="expand"
21
22 [gui]
23
24 common/drop_mouse_on_gui_input_disabled=true
25
26 [physics]
27
28 common/enable_pause_aware_picking=true

```

C.2.4 DemoNode.tscn

```

1  [gd_scene load_steps=2 format=3 uid="uid://bu380we4od0ln"]
2
3  [ext_resource type="Script" path="res://DemoNode.gd" id="1"]
4
5  [node name="DemoNode" type="Node"]
6  script = ExtResource("1")
7  choices = "deterministic"
8
9  [node name="Timer" type="Timer" parent="."]
10
11 [node name="Line2D" type="Line2D" parent="."]
12 points = PackedVector2Array(69, 297)
13
14 [node name="TextLabel" type="Label" parent="."]
15 offset_right = 1152.0
16 offset_bottom = 23.0
17 autowrap_mode = 3
18

```

```

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

```

C.2.5 DemoNode.gd

```

1  extends Node
2
3  # Basic: https://youtu.be/feNVBEPXAcE?t=77 (L = +)
4  # Choices: http://paulbourke.net/fractals/lsys/
5  # Deterministic: https://ww1.biology.uni-hamburg.de/b-online/
    e28_3/lsys.html#DOL-system
6
7  @export_enum("basic", "choices", "deterministic") var choices:
    String = "choices"
8  @export var axiom: String
9  @onready var string: String
10 @onready var timer = \${Timer}
11 @onready var line = \${Line2D}
12 @onready var label = \${TextLabel}
13 @onready var rules: Array[Dictionary]
14
15 func set_values():
16     if choices == "basic":
17         rules = [
18             {
19                 "from": "F",
20                 "to": "F+F"
21             }
22         ]
23         axiom = "F+"
24     elif choices == "choices":
25         rules = [

```

```

26         {
27             "from": "F",
28             "to": "F+--FFFF+F+-FF"
29         }
30     ]
31     axiom = "F+F+F+F"
32     elif choices == "deterministic":
33         rules = [
34             {
35                 "from": "a",
36                 "to": "ab"
37             },
38             {
39                 "from": "b",
40                 "to": "a"
41             }
42         ]
43         axiom = "b"
44
45     func _ready():
46         set_values()
47         string = axiom
48         label.size.x = get_viewport().size.x
49         label.text = string
50         timer.start()
51
52     func get_new_replacement(character: String) -> String:
53         for rule in rules:
54             if rule["from"] == character:
55                 return rule["to"]
56         return ""
57
58     func _on_Timer_timeout():

```

```

59     var new_string = ""
60     for character in string:
61         new_string += get_new_replacement(character)
62     string = new_string
63     label.text = string
64     print(len(string))

```

C.2.6 icon.svg.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://cwnnuqmej04q"
6  path="res://.godot/imported/icon.svg-218
      a8f2b3041327d8a5756f3a245f83b.ctex"
7  metadata={
8  "vram_texture": false
9  }
10
11 [deps]
12
13 source_file="res://icon.svg"
14 dest_files=["res://.godot/imported/icon.svg-218
      a8f2b3041327d8a5756f3a245f83b.ctex"]
15
16 [params]
17
18 compress/mode=0
19 compress/high_quality=false
20 compress/lossy_quality=0.7
21 compress/hdr_compression=1

```



```

22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
30  process/normal_map_invert_y=false
31  process/hdr_as_srgb=false
32  process/hdr_clamp_exposure=false
33  process/size_limit=0
34  detect_3d/compress_to=1
35  svg/scale=1.0
36  editor/scale_with_editor_scale=false
37  editor/convert_colors_with_editor_theme=false

```

C.3 ProcGenRPG (L-System)

C.3.1 .gitattributes

```

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

```

C.3.2 .gitignore

```

1  # Godot 4+ specific ignores
2  .godot/

```

C.3.3 project.godot

```
1 ; Engine configuration file.
2 ; It's best edited using the editor UI and not directly,
3 ; since the parameters that go here are not all obvious.
4 ;
5 ; Format:
6 ; [section] ; section goes between []
7 ; param=value ; assign values to parameters
8
9 config_version=5
10
11 [application]
12
13 config/name="Proc Gen RPG"
14 run/main_scene="res://tile_map.tscn"
15 config/features=PackedStringArray("4.0", "Forward Plus")
16 config/icon="res://icon.svg"
17
18 [display]
19
20 window/size/viewport_height=640
21 window/stretch/mode="canvas_items"
```

C.3.4 l_system.tscn

```
1 [gd_scene load_steps=2 format=3 uid="uid://d0v18e7ms571f"]
2
3 [ext_resource type="Script" path="res://l_system.gd" id="1_elydp"]
4
5 [node name="LSystem" type="Node"]
6 script = ExtResource("1_elydp")
```

C.3.5 l_system.gd

```
1  extends Node
2
3  class_name LSystem
4
5  @onready var tile_map: TileMap = get_parent()
6  @export var axiom: String = "OWB"
7  @onready var string: String = axiom
8  @export var rules: Array[Dictionary] = [
9      {
10         "from": "O",
11         "to": "OWO"
12     },
13     {
14         "from": "W",
15         "to": "WB"
16     },
17     {
18         "from": "B",
19         "to": "BWO"
20     }
21 ]
22
23  const FLOWERS_1: Vector2i = Vector2i(3, 7) # "O" = ORANGE
24  const FLOWERS_2: Vector2i = Vector2i(3, 10) # "W" = WHITE
25  const FLOWERS_3: Vector2i = Vector2i(3, 13) # "B" = BLUE
26
27  func get_new_replacement(character: String) -> String:
28      for rule in rules:
29          if rule["from"] == character:
30              return rule["to"]
31      return ""
```

```

32
33 func size() -> int:
34     return tile_map.x_tile_range * tile_map.y_tile_range
35
36 func parse() -> String:
37     var size: int = size()
38     while len(string) <= size:
39         var new_string = ""
40         for character in string:
41             new_string += get_new_replacement(character)
42         string = new_string
43     string = string.substr(0, size)
44     return string
45
46 func paint() -> void:
47     string = parse()
48     var size: int = size()
49     var i: int = -1
50     for x in range(tile_map.x_tile_range):
51         for y in range(tile_map.y_tile_range):
52             i += 1
53             if string[i] == "0":
54                 tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_1)
55             elif string[i] == "W":
56                 tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_2)
57             elif string[i] == "B":
58                 tile_map.set_cell(1, Vector2i(x, y), 0, FLOWERS_3)

```

C.3.6 tile_map.tscn

```

1 [gd_scene load_steps=4 format=3 uid="uid://bwhvtqld3yo8m"]
2

```

```

3  [ext_resource type="TileSet" uid="uid://c168x78r0tful" path="res://
    Tiles.tres" id="1_l3nwg"]
4  [ext_resource type="Script" path="res://tile_map.gd" id="2_wrxl8"]
5  [ext_resource type="PackedScene" uid="uid://d0v18e7ms571f" path="
    res://l_system.tscn" id="3_ktw1n"]
6
7  [node name="TileMap" type="TileMap"]
8  tile_set = ExtResource("1_l3nwg")
9  format = 2
10 layer_0/name = "Grass"
11 layer_1/name = "Things"
12 layer_1/enabled = true
13 layer_1/modulate = Color(1, 1, 1, 1)
14 layer_1/y_sort_enabled = false
15 layer_1/y_sort_origin = 0
16 layer_1/z_index = 0
17 layer_1/tile_data = PackedInt32Array()
18 script = ExtResource("2_wrxl8")
19
20 [node name="LSystem" parent="." instance=ExtResource("3_ktw1n")]

```

C.3.7 tile_map.gd

```

1  extends TileMap
2
3  @onready var l_system: LSystem = \${LSystem}
4
5  var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
6  var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
7

```

```

8  const GRASS_1: Vector2i = Vector2i(5, 0)
9  const GRASS_2: Vector2i = Vector2i(5, 1)
10
11  func pick_grass_tile() -> Vector2i:
12      return [GRASS_1, GRASS_2].pick_random()
13
14  func cover_map_with_grass() -> void:
15      for x in range(-50, x_tile_range + 50):
16          for y in range(-50, y_tile_range + 50):
17              set_cell(0, Vector2i(x, y), 0, pick_grass_tile())
18
19  \# Called when the node enters the scene tree for the first time.
20  func _ready() -> void:
21      randomize()
22      cover_map_with_grass()
23      l_system.paint()

```

C.3.8 icon.svg.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://b45qexb3wmhym"
6  path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
7  metadata={
8  "vram_texture": false
9  }
10
11  [deps]
12

```

```

13  source_file="res://icon.svg"
14  dest_files=["res://.godot/imported/icon.svg-218
           a8f2b3041327d8a5756f3a245f83b.ctex"]
15
16  [params]
17
18  compress/mode=0
19  compress/high_quality=false
20  compress/lossy_quality=0.7
21  compress/hdr_compression=1
22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
30  process/normal_map_invert_y=false
31  process/hdr_as_srgb=false
32  process/hdr_clamp_exposure=false
33  process/size_limit=0
34  detect_3d/compress_to=1
35  svg/scale=1.0
36  editor/scale_with_editor_scale=false
37  editor/convert_colors_with_editor_theme=false

```

C.3.9 roguelikeSheet_transparent.png.import

```

1  [remap]
2
3  importer="texture"

```

```

4  type="CompressedTexture2D"
5  uid="uid://13ktp0qup5xb"
6  path="res://.godot/imported/roguelikeSheet_transparent.png-22
      f6b70da04549e371d1f15fe9d96005.ctex"
7  metadata={
8  "vram_texture": false
9  }
10
11  [deps]
12
13  source_file="res://roguelikeSheet_transparent.png"
14  dest_files=["res://.godot/imported/roguelikeSheet_transparent.png
      -22f6b70da04549e371d1f15fe9d96005.ctex"]
15
16  [params]
17
18  compress/mode=0
19  compress/high_quality=false
20  compress/lossy_quality=0.7
21  compress/hdr_compression=1
22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
30  process/normal_map_invert_y=false
31  process/hdr_as_srgb=false
32  process/hdr_clamp_exposure=false
33  process/size_limit=0
34  detect_3d/compress_to=1

```


C.4 VoronoiCellsGD4

C.4.1 .gitattributes

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

C.4.2 .gitignore

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

C.4.3 project.godot

```
1  ; Engine configuration file.
2  ; It's best edited using the editor UI and not directly,
3  ; since the parameters that go here are not all obvious.
4  ;
5  ; Format:
6  ;   [section] ; section goes between []
7  ;   param=value ; assign values to parameters
8
9  config_version=5
10
11  [application]
12
13  config/name="Voronoi Cells"
14  run/main_scene="res://tile_map.tscn"
15  config/features=PackedStringArray("4.0", "Forward Plus")
16  config/icon="res://icon.svg"
17
```

```
18 [display]
19
20 window/size/viewport_height=640
```

C.4.4 tile_map.tscn

```
1 [gd_scene load_steps=5 format=3 uid="uid://d6lxn5bdh1w"]
2
3 [ext_resource type="Texture2D" uid="uid://cpign73sfbsrt" path="res
  ://monochrome_packed.png" id="1_o183d"]
4 [ext_resource type="Script" path="res://tile_map.gd" id="2_lf4lw"]
5
6 [sub_resource type="TileSetAtlasSource" id="
  TileSetAtlasSource_6h0bd"]
7 texture = ExtResource("1_o183d")
8 0:0/0 = 0
9 1:0/0 = 0
10 2:0/0 = 0
11 3:0/0 = 0
12 4:0/0 = 0
13 5:0/0 = 0
14 6:0/0 = 0
15 7:0/0 = 0
16 8:0/0 = 0
17 9:0/0 = 0
18 10:0/0 = 0
19 11:0/0 = 0
20 12:0/0 = 0
21 13:0/0 = 0
22 14:0/0 = 0
23 15:0/0 = 0
24 16:0/0 = 0
```

25 17:0/0 = 0
26 18:0/0 = 0
27 19:0/0 = 0
28 20:0/0 = 0
29 21:0/0 = 0
30 22:0/0 = 0
31 23:0/0 = 0
32 24:0/0 = 0
33 25:0/0 = 0
34 26:0/0 = 0
35 27:0/0 = 0
36 28:0/0 = 0
37 29:0/0 = 0
38 30:0/0 = 0
39 31:0/0 = 0
40 32:0/0 = 0
41 33:0/0 = 0
42 34:0/0 = 0
43 35:0/0 = 0
44 36:0/0 = 0
45 37:0/0 = 0
46 38:0/0 = 0
47 39:0/0 = 0
48 40:0/0 = 0
49 41:0/0 = 0
50 42:0/0 = 0
51 43:0/0 = 0
52 44:0/0 = 0
53 45:0/0 = 0
54 46:0/0 = 0
55 47:0/0 = 0
56 48:0/0 = 0
57 0:1/0 = 0

58 1:1/0 = 0
59 2:1/0 = 0
60 3:1/0 = 0
61 4:1/0 = 0
62 5:1/0 = 0
63 6:1/0 = 0
64 7:1/0 = 0
65 8:1/0 = 0
66 9:1/0 = 0
67 10:1/0 = 0
68 11:1/0 = 0
69 12:1/0 = 0
70 13:1/0 = 0
71 14:1/0 = 0
72 15:1/0 = 0
73 16:1/0 = 0
74 17:1/0 = 0
75 18:1/0 = 0
76 19:1/0 = 0
77 20:1/0 = 0
78 21:1/0 = 0
79 22:1/0 = 0
80 23:1/0 = 0
81 24:1/0 = 0
82 25:1/0 = 0
83 26:1/0 = 0
84 27:1/0 = 0
85 28:1/0 = 0
86 29:1/0 = 0
87 30:1/0 = 0
88 31:1/0 = 0
89 32:1/0 = 0
90 33:1/0 = 0

91 34:1/0 = 0
92 35:1/0 = 0
93 36:1/0 = 0
94 37:1/0 = 0
95 38:1/0 = 0
96 39:1/0 = 0
97 40:1/0 = 0
98 41:1/0 = 0
99 42:1/0 = 0
100 43:1/0 = 0
101 44:1/0 = 0
102 45:1/0 = 0
103 46:1/0 = 0
104 47:1/0 = 0
105 48:1/0 = 0
106 0:2/0 = 0
107 1:2/0 = 0
108 2:2/0 = 0
109 3:2/0 = 0
110 4:2/0 = 0
111 5:2/0 = 0
112 6:2/0 = 0
113 7:2/0 = 0
114 8:2/0 = 0
115 9:2/0 = 0
116 10:2/0 = 0
117 11:2/0 = 0
118 12:2/0 = 0
119 13:2/0 = 0
120 14:2/0 = 0
121 15:2/0 = 0
122 16:2/0 = 0
123 17:2/0 = 0

124	$18:2/0 = 0$
125	$19:2/0 = 0$
126	$20:2/0 = 0$
127	$21:2/0 = 0$
128	$22:2/0 = 0$
129	$23:2/0 = 0$
130	$24:2/0 = 0$
131	$25:2/0 = 0$
132	$26:2/0 = 0$
133	$27:2/0 = 0$
134	$28:2/0 = 0$
135	$29:2/0 = 0$
136	$30:2/0 = 0$
137	$31:2/0 = 0$
138	$32:2/0 = 0$
139	$33:2/0 = 0$
140	$34:2/0 = 0$
141	$35:2/0 = 0$
142	$36:2/0 = 0$
143	$37:2/0 = 0$
144	$38:2/0 = 0$
145	$39:2/0 = 0$
146	$40:2/0 = 0$
147	$41:2/0 = 0$
148	$42:2/0 = 0$
149	$43:2/0 = 0$
150	$44:2/0 = 0$
151	$45:2/0 = 0$
152	$46:2/0 = 0$
153	$47:2/0 = 0$
154	$48:2/0 = 0$
155	$0:3/0 = 0$
156	$1:3/0 = 0$

157 2:3/0 = 0
158 3:3/0 = 0
159 4:3/0 = 0
160 5:3/0 = 0
161 6:3/0 = 0
162 7:3/0 = 0
163 8:3/0 = 0
164 9:3/0 = 0
165 10:3/0 = 0
166 11:3/0 = 0
167 12:3/0 = 0
168 13:3/0 = 0
169 14:3/0 = 0
170 15:3/0 = 0
171 16:3/0 = 0
172 17:3/0 = 0
173 18:3/0 = 0
174 19:3/0 = 0
175 20:3/0 = 0
176 21:3/0 = 0
177 22:3/0 = 0
178 23:3/0 = 0
179 24:3/0 = 0
180 25:3/0 = 0
181 26:3/0 = 0
182 27:3/0 = 0
183 28:3/0 = 0
184 29:3/0 = 0
185 30:3/0 = 0
186 31:3/0 = 0
187 32:3/0 = 0
188 33:3/0 = 0
189 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

223 19:4/0 = 0
224 20:4/0 = 0
225 21:4/0 = 0
226 22:4/0 = 0
227 23:4/0 = 0
228 24:4/0 = 0
229 25:4/0 = 0
230 26:4/0 = 0
231 27:4/0 = 0
232 28:4/0 = 0
233 29:4/0 = 0
234 30:4/0 = 0
235 31:4/0 = 0
236 32:4/0 = 0
237 33:4/0 = 0
238 34:4/0 = 0
239 35:4/0 = 0
240 36:4/0 = 0
241 37:4/0 = 0
242 38:4/0 = 0
243 39:4/0 = 0
244 40:4/0 = 0
245 41:4/0 = 0
246 42:4/0 = 0
247 43:4/0 = 0
248 44:4/0 = 0
249 45:4/0 = 0
250 46:4/0 = 0
251 47:4/0 = 0
252 48:4/0 = 0
253 0:5/0 = 0
254 1:5/0 = 0
255 2:5/0 = 0

256	$3:5/0 = 0$
257	$4:5/0 = 0$
258	$5:5/0 = 0$
259	$6:5/0 = 0$
260	$7:5/0 = 0$
261	$8:5/0 = 0$
262	$9:5/0 = 0$
263	$10:5/0 = 0$
264	$11:5/0 = 0$
265	$12:5/0 = 0$
266	$13:5/0 = 0$
267	$14:5/0 = 0$
268	$15:5/0 = 0$
269	$16:5/0 = 0$
270	$17:5/0 = 0$
271	$18:5/0 = 0$
272	$19:5/0 = 0$
273	$20:5/0 = 0$
274	$21:5/0 = 0$
275	$22:5/0 = 0$
276	$23:5/0 = 0$
277	$24:5/0 = 0$
278	$25:5/0 = 0$
279	$26:5/0 = 0$
280	$27:5/0 = 0$
281	$28:5/0 = 0$
282	$29:5/0 = 0$
283	$30:5/0 = 0$
284	$31:5/0 = 0$
285	$32:5/0 = 0$
286	$33:5/0 = 0$
287	$34:5/0 = 0$
288	$35:5/0 = 0$

289	$36:5/0 = 0$
290	$37:5/0 = 0$
291	$38:5/0 = 0$
292	$39:5/0 = 0$
293	$40:5/0 = 0$
294	$41:5/0 = 0$
295	$42:5/0 = 0$
296	$43:5/0 = 0$
297	$44:5/0 = 0$
298	$45:5/0 = 0$
299	$46:5/0 = 0$
300	$47:5/0 = 0$
301	$48:5/0 = 0$
302	$0:6/0 = 0$
303	$1:6/0 = 0$
304	$2:6/0 = 0$
305	$3:6/0 = 0$
306	$4:6/0 = 0$
307	$5:6/0 = 0$
308	$6:6/0 = 0$
309	$7:6/0 = 0$
310	$8:6/0 = 0$
311	$9:6/0 = 0$
312	$10:6/0 = 0$
313	$11:6/0 = 0$
314	$12:6/0 = 0$
315	$13:6/0 = 0$
316	$14:6/0 = 0$
317	$15:6/0 = 0$
318	$16:6/0 = 0$
319	$17:6/0 = 0$
320	$18:6/0 = 0$
321	$19:6/0 = 0$

322 20:6/0 = 0
323 21:6/0 = 0
324 22:6/0 = 0
325 23:6/0 = 0
326 24:6/0 = 0
327 25:6/0 = 0
328 26:6/0 = 0
329 27:6/0 = 0
330 28:6/0 = 0
331 29:6/0 = 0
332 30:6/0 = 0
333 31:6/0 = 0
334 32:6/0 = 0
335 33:6/0 = 0
336 34:6/0 = 0
337 35:6/0 = 0
338 36:6/0 = 0
339 37:6/0 = 0
340 38:6/0 = 0
341 39:6/0 = 0
342 40:6/0 = 0
343 41:6/0 = 0
344 42:6/0 = 0
345 43:6/0 = 0
346 44:6/0 = 0
347 45:6/0 = 0
348 46:6/0 = 0
349 47:6/0 = 0
350 48:6/0 = 0
351 0:7/0 = 0
352 1:7/0 = 0
353 2:7/0 = 0
354 3:7/0 = 0

355 $4:7/0 = 0$
356 $5:7/0 = 0$
357 $6:7/0 = 0$
358 $7:7/0 = 0$
359 $8:7/0 = 0$
360 $9:7/0 = 0$
361 $10:7/0 = 0$
362 $11:7/0 = 0$
363 $12:7/0 = 0$
364 $13:7/0 = 0$
365 $14:7/0 = 0$
366 $15:7/0 = 0$
367 $16:7/0 = 0$
368 $17:7/0 = 0$
369 $18:7/0 = 0$
370 $19:7/0 = 0$
371 $20:7/0 = 0$
372 $21:7/0 = 0$
373 $22:7/0 = 0$
374 $23:7/0 = 0$
375 $24:7/0 = 0$
376 $25:7/0 = 0$
377 $26:7/0 = 0$
378 $27:7/0 = 0$
379 $28:7/0 = 0$
380 $29:7/0 = 0$
381 $30:7/0 = 0$
382 $31:7/0 = 0$
383 $32:7/0 = 0$
384 $33:7/0 = 0$
385 $34:7/0 = 0$
386 $35:7/0 = 0$
387 $36:7/0 = 0$

388 $37:7/0 = 0$
389 $38:7/0 = 0$
390 $39:7/0 = 0$
391 $40:7/0 = 0$
392 $41:7/0 = 0$
393 $42:7/0 = 0$
394 $43:7/0 = 0$
395 $44:7/0 = 0$
396 $45:7/0 = 0$
397 $46:7/0 = 0$
398 $47:7/0 = 0$
399 $48:7/0 = 0$
400 $0:8/0 = 0$
401 $1:8/0 = 0$
402 $2:8/0 = 0$
403 $3:8/0 = 0$
404 $4:8/0 = 0$
405 $5:8/0 = 0$
406 $6:8/0 = 0$
407 $7:8/0 = 0$
408 $8:8/0 = 0$
409 $9:8/0 = 0$
410 $10:8/0 = 0$
411 $11:8/0 = 0$
412 $12:8/0 = 0$
413 $13:8/0 = 0$
414 $14:8/0 = 0$
415 $15:8/0 = 0$
416 $16:8/0 = 0$
417 $17:8/0 = 0$
418 $18:8/0 = 0$
419 $19:8/0 = 0$
420 $20:8/0 = 0$

421 21:8/0 = 0
422 22:8/0 = 0
423 23:8/0 = 0
424 24:8/0 = 0
425 25:8/0 = 0
426 26:8/0 = 0
427 27:8/0 = 0
428 28:8/0 = 0
429 29:8/0 = 0
430 30:8/0 = 0
431 31:8/0 = 0
432 32:8/0 = 0
433 33:8/0 = 0
434 34:8/0 = 0
435 35:8/0 = 0
436 36:8/0 = 0
437 37:8/0 = 0
438 38:8/0 = 0
439 39:8/0 = 0
440 40:8/0 = 0
441 41:8/0 = 0
442 42:8/0 = 0
443 43:8/0 = 0
444 44:8/0 = 0
445 45:8/0 = 0
446 46:8/0 = 0
447 47:8/0 = 0
448 48:8/0 = 0
449 0:9/0 = 0
450 1:9/0 = 0
451 2:9/0 = 0
452 3:9/0 = 0
453 4:9/0 = 0

454 $5:9/0 = 0$
455 $6:9/0 = 0$
456 $7:9/0 = 0$
457 $8:9/0 = 0$
458 $9:9/0 = 0$
459 $10:9/0 = 0$
460 $11:9/0 = 0$
461 $12:9/0 = 0$
462 $13:9/0 = 0$
463 $14:9/0 = 0$
464 $15:9/0 = 0$
465 $16:9/0 = 0$
466 $17:9/0 = 0$
467 $18:9/0 = 0$
468 $19:9/0 = 0$
469 $20:9/0 = 0$
470 $21:9/0 = 0$
471 $22:9/0 = 0$
472 $23:9/0 = 0$
473 $24:9/0 = 0$
474 $25:9/0 = 0$
475 $26:9/0 = 0$
476 $27:9/0 = 0$
477 $28:9/0 = 0$
478 $29:9/0 = 0$
479 $30:9/0 = 0$
480 $31:9/0 = 0$
481 $32:9/0 = 0$
482 $33:9/0 = 0$
483 $34:9/0 = 0$
484 $35:9/0 = 0$
485 $36:9/0 = 0$
486 $37:9/0 = 0$

487 38:9/0 = 0
488 39:9/0 = 0
489 40:9/0 = 0
490 41:9/0 = 0
491 42:9/0 = 0
492 43:9/0 = 0
493 44:9/0 = 0
494 45:9/0 = 0
495 46:9/0 = 0
496 47:9/0 = 0
497 48:9/0 = 0
498 0:10/0 = 0
499 1:10/0 = 0
500 2:10/0 = 0
501 3:10/0 = 0
502 4:10/0 = 0
503 5:10/0 = 0
504 6:10/0 = 0
505 7:10/0 = 0
506 8:10/0 = 0
507 9:10/0 = 0
508 10:10/0 = 0
509 11:10/0 = 0
510 12:10/0 = 0
511 13:10/0 = 0
512 14:10/0 = 0
513 15:10/0 = 0
514 16:10/0 = 0
515 17:10/0 = 0
516 18:10/0 = 0
517 19:10/0 = 0
518 20:10/0 = 0
519 21:10/0 = 0

520 22:10/0 = 0
521 23:10/0 = 0
522 24:10/0 = 0
523 25:10/0 = 0
524 26:10/0 = 0
525 27:10/0 = 0
526 28:10/0 = 0
527 29:10/0 = 0
528 30:10/0 = 0
529 31:10/0 = 0
530 32:10/0 = 0
531 33:10/0 = 0
532 34:10/0 = 0
533 35:10/0 = 0
534 36:10/0 = 0
535 37:10/0 = 0
536 38:10/0 = 0
537 39:10/0 = 0
538 40:10/0 = 0
539 41:10/0 = 0
540 42:10/0 = 0
541 43:10/0 = 0
542 44:10/0 = 0
543 45:10/0 = 0
544 46:10/0 = 0
545 47:10/0 = 0
546 48:10/0 = 0
547 0:11/0 = 0
548 1:11/0 = 0
549 2:11/0 = 0
550 3:11/0 = 0
551 4:11/0 = 0
552 5:11/0 = 0

553 6:11/0 = 0
554 7:11/0 = 0
555 8:11/0 = 0
556 9:11/0 = 0
557 10:11/0 = 0
558 11:11/0 = 0
559 12:11/0 = 0
560 13:11/0 = 0
561 14:11/0 = 0
562 15:11/0 = 0
563 16:11/0 = 0
564 17:11/0 = 0
565 18:11/0 = 0
566 19:11/0 = 0
567 20:11/0 = 0
568 21:11/0 = 0
569 22:11/0 = 0
570 23:11/0 = 0
571 24:11/0 = 0
572 25:11/0 = 0
573 26:11/0 = 0
574 27:11/0 = 0
575 28:11/0 = 0
576 29:11/0 = 0
577 30:11/0 = 0
578 31:11/0 = 0
579 32:11/0 = 0
580 33:11/0 = 0
581 34:11/0 = 0
582 35:11/0 = 0
583 36:11/0 = 0
584 37:11/0 = 0
585 38:11/0 = 0

586 39:11/0 = 0
587 40:11/0 = 0
588 41:11/0 = 0
589 42:11/0 = 0
590 43:11/0 = 0
591 44:11/0 = 0
592 45:11/0 = 0
593 46:11/0 = 0
594 47:11/0 = 0
595 48:11/0 = 0
596 0:12/0 = 0
597 1:12/0 = 0
598 2:12/0 = 0
599 3:12/0 = 0
600 4:12/0 = 0
601 5:12/0 = 0
602 6:12/0 = 0
603 7:12/0 = 0
604 8:12/0 = 0
605 9:12/0 = 0
606 10:12/0 = 0
607 11:12/0 = 0
608 12:12/0 = 0
609 13:12/0 = 0
610 14:12/0 = 0
611 15:12/0 = 0
612 16:12/0 = 0
613 17:12/0 = 0
614 18:12/0 = 0
615 19:12/0 = 0
616 20:12/0 = 0
617 21:12/0 = 0
618 22:12/0 = 0

619 23:12/0 = 0
620 24:12/0 = 0
621 25:12/0 = 0
622 26:12/0 = 0
623 27:12/0 = 0
624 28:12/0 = 0
625 29:12/0 = 0
626 30:12/0 = 0
627 31:12/0 = 0
628 32:12/0 = 0
629 33:12/0 = 0
630 34:12/0 = 0
631 35:12/0 = 0
632 36:12/0 = 0
633 37:12/0 = 0
634 38:12/0 = 0
635 39:12/0 = 0
636 40:12/0 = 0
637 41:12/0 = 0
638 42:12/0 = 0
639 43:12/0 = 0
640 44:12/0 = 0
641 45:12/0 = 0
642 46:12/0 = 0
643 47:12/0 = 0
644 48:12/0 = 0
645 0:13/0 = 0
646 1:13/0 = 0
647 2:13/0 = 0
648 3:13/0 = 0
649 4:13/0 = 0
650 5:13/0 = 0
651 6:13/0 = 0

652 7:13/0 = 0
653 8:13/0 = 0
654 9:13/0 = 0
655 10:13/0 = 0
656 11:13/0 = 0
657 12:13/0 = 0
658 13:13/0 = 0
659 14:13/0 = 0
660 15:13/0 = 0
661 16:13/0 = 0
662 17:13/0 = 0
663 18:13/0 = 0
664 19:13/0 = 0
665 20:13/0 = 0
666 21:13/0 = 0
667 22:13/0 = 0
668 23:13/0 = 0
669 24:13/0 = 0
670 25:13/0 = 0
671 26:13/0 = 0
672 27:13/0 = 0
673 28:13/0 = 0
674 29:13/0 = 0
675 30:13/0 = 0
676 31:13/0 = 0
677 32:13/0 = 0
678 33:13/0 = 0
679 34:13/0 = 0
680 35:13/0 = 0
681 36:13/0 = 0
682 37:13/0 = 0
683 38:13/0 = 0
684 39:13/0 = 0

685 40:13/0 = 0
686 41:13/0 = 0
687 42:13/0 = 0
688 43:13/0 = 0
689 44:13/0 = 0
690 45:13/0 = 0
691 46:13/0 = 0
692 47:13/0 = 0
693 48:13/0 = 0
694 0:14/0 = 0
695 1:14/0 = 0
696 2:14/0 = 0
697 3:14/0 = 0
698 4:14/0 = 0
699 5:14/0 = 0
700 6:14/0 = 0
701 7:14/0 = 0
702 8:14/0 = 0
703 9:14/0 = 0
704 10:14/0 = 0
705 11:14/0 = 0
706 12:14/0 = 0
707 13:14/0 = 0
708 14:14/0 = 0
709 15:14/0 = 0
710 16:14/0 = 0
711 17:14/0 = 0
712 18:14/0 = 0
713 19:14/0 = 0
714 20:14/0 = 0
715 21:14/0 = 0
716 22:14/0 = 0
717 23:14/0 = 0

718	$24:14/0 = 0$
719	$25:14/0 = 0$
720	$26:14/0 = 0$
721	$27:14/0 = 0$
722	$28:14/0 = 0$
723	$29:14/0 = 0$
724	$30:14/0 = 0$
725	$31:14/0 = 0$
726	$32:14/0 = 0$
727	$33:14/0 = 0$
728	$34:14/0 = 0$
729	$35:14/0 = 0$
730	$36:14/0 = 0$
731	$37:14/0 = 0$
732	$38:14/0 = 0$
733	$39:14/0 = 0$
734	$40:14/0 = 0$
735	$41:14/0 = 0$
736	$42:14/0 = 0$
737	$43:14/0 = 0$
738	$44:14/0 = 0$
739	$45:14/0 = 0$
740	$46:14/0 = 0$
741	$47:14/0 = 0$
742	$48:14/0 = 0$
743	$0:15/0 = 0$
744	$1:15/0 = 0$
745	$2:15/0 = 0$
746	$3:15/0 = 0$
747	$4:15/0 = 0$
748	$5:15/0 = 0$
749	$6:15/0 = 0$
750	$7:15/0 = 0$

751 8:15/0 = 0
752 9:15/0 = 0
753 10:15/0 = 0
754 11:15/0 = 0
755 12:15/0 = 0
756 13:15/0 = 0
757 14:15/0 = 0
758 15:15/0 = 0
759 16:15/0 = 0
760 17:15/0 = 0
761 18:15/0 = 0
762 19:15/0 = 0
763 20:15/0 = 0
764 21:15/0 = 0
765 22:15/0 = 0
766 23:15/0 = 0
767 24:15/0 = 0
768 25:15/0 = 0
769 26:15/0 = 0
770 27:15/0 = 0
771 28:15/0 = 0
772 29:15/0 = 0
773 30:15/0 = 0
774 31:15/0 = 0
775 32:15/0 = 0
776 33:15/0 = 0
777 34:15/0 = 0
778 35:15/0 = 0
779 36:15/0 = 0
780 37:15/0 = 0
781 38:15/0 = 0
782 39:15/0 = 0
783 40:15/0 = 0

784	$41:15/0 = 0$
785	$42:15/0 = 0$
786	$43:15/0 = 0$
787	$44:15/0 = 0$
788	$45:15/0 = 0$
789	$46:15/0 = 0$
790	$47:15/0 = 0$
791	$48:15/0 = 0$
792	$0:16/0 = 0$
793	$1:16/0 = 0$
794	$2:16/0 = 0$
795	$3:16/0 = 0$
796	$4:16/0 = 0$
797	$5:16/0 = 0$
798	$6:16/0 = 0$
799	$7:16/0 = 0$
800	$8:16/0 = 0$
801	$9:16/0 = 0$
802	$10:16/0 = 0$
803	$11:16/0 = 0$
804	$12:16/0 = 0$
805	$13:16/0 = 0$
806	$14:16/0 = 0$
807	$15:16/0 = 0$
808	$16:16/0 = 0$
809	$17:16/0 = 0$
810	$18:16/0 = 0$
811	$19:16/0 = 0$
812	$20:16/0 = 0$
813	$21:16/0 = 0$
814	$22:16/0 = 0$
815	$23:16/0 = 0$
816	$24:16/0 = 0$

817 25:16/0 = 0
818 26:16/0 = 0
819 27:16/0 = 0
820 28:16/0 = 0
821 29:16/0 = 0
822 30:16/0 = 0
823 31:16/0 = 0
824 32:16/0 = 0
825 33:16/0 = 0
826 34:16/0 = 0
827 35:16/0 = 0
828 36:16/0 = 0
829 37:16/0 = 0
830 38:16/0 = 0
831 39:16/0 = 0
832 40:16/0 = 0
833 41:16/0 = 0
834 42:16/0 = 0
835 43:16/0 = 0
836 44:16/0 = 0
837 45:16/0 = 0
838 46:16/0 = 0
839 47:16/0 = 0
840 48:16/0 = 0
841 0:17/0 = 0
842 1:17/0 = 0
843 2:17/0 = 0
844 3:17/0 = 0
845 4:17/0 = 0
846 5:17/0 = 0
847 6:17/0 = 0
848 7:17/0 = 0
849 8:17/0 = 0

850 9:17/0 = 0
851 10:17/0 = 0
852 11:17/0 = 0
853 12:17/0 = 0
854 13:17/0 = 0
855 14:17/0 = 0
856 15:17/0 = 0
857 16:17/0 = 0
858 17:17/0 = 0
859 18:17/0 = 0
860 19:17/0 = 0
861 20:17/0 = 0
862 21:17/0 = 0
863 22:17/0 = 0
864 23:17/0 = 0
865 24:17/0 = 0
866 25:17/0 = 0
867 26:17/0 = 0
868 27:17/0 = 0
869 28:17/0 = 0
870 29:17/0 = 0
871 30:17/0 = 0
872 31:17/0 = 0
873 32:17/0 = 0
874 33:17/0 = 0
875 34:17/0 = 0
876 35:17/0 = 0
877 36:17/0 = 0
878 37:17/0 = 0
879 38:17/0 = 0
880 39:17/0 = 0
881 40:17/0 = 0
882 41:17/0 = 0

883 42:17/0 = 0
884 43:17/0 = 0
885 44:17/0 = 0
886 45:17/0 = 0
887 46:17/0 = 0
888 47:17/0 = 0
889 48:17/0 = 0
890 0:18/0 = 0
891 1:18/0 = 0
892 2:18/0 = 0
893 3:18/0 = 0
894 4:18/0 = 0
895 5:18/0 = 0
896 6:18/0 = 0
897 7:18/0 = 0
898 8:18/0 = 0
899 9:18/0 = 0
900 10:18/0 = 0
901 11:18/0 = 0
902 12:18/0 = 0
903 13:18/0 = 0
904 14:18/0 = 0
905 15:18/0 = 0
906 16:18/0 = 0
907 17:18/0 = 0
908 18:18/0 = 0
909 19:18/0 = 0
910 20:18/0 = 0
911 21:18/0 = 0
912 22:18/0 = 0
913 23:18/0 = 0
914 24:18/0 = 0
915 25:18/0 = 0

916 26:18/0 = 0
917 27:18/0 = 0
918 28:18/0 = 0
919 29:18/0 = 0
920 30:18/0 = 0
921 31:18/0 = 0
922 32:18/0 = 0
923 33:18/0 = 0
924 34:18/0 = 0
925 35:18/0 = 0
926 36:18/0 = 0
927 37:18/0 = 0
928 38:18/0 = 0
929 39:18/0 = 0
930 40:18/0 = 0
931 41:18/0 = 0
932 42:18/0 = 0
933 43:18/0 = 0
934 44:18/0 = 0
935 45:18/0 = 0
936 46:18/0 = 0
937 47:18/0 = 0
938 48:18/0 = 0
939 0:19/0 = 0
940 1:19/0 = 0
941 2:19/0 = 0
942 3:19/0 = 0
943 4:19/0 = 0
944 5:19/0 = 0
945 6:19/0 = 0
946 7:19/0 = 0
947 8:19/0 = 0
948 9:19/0 = 0

949 10:19/0 = 0
950 11:19/0 = 0
951 12:19/0 = 0
952 13:19/0 = 0
953 14:19/0 = 0
954 15:19/0 = 0
955 16:19/0 = 0
956 17:19/0 = 0
957 18:19/0 = 0
958 19:19/0 = 0
959 20:19/0 = 0
960 21:19/0 = 0
961 22:19/0 = 0
962 23:19/0 = 0
963 24:19/0 = 0
964 25:19/0 = 0
965 26:19/0 = 0
966 27:19/0 = 0
967 28:19/0 = 0
968 29:19/0 = 0
969 30:19/0 = 0
970 31:19/0 = 0
971 32:19/0 = 0
972 33:19/0 = 0
973 34:19/0 = 0
974 35:19/0 = 0
975 36:19/0 = 0
976 37:19/0 = 0
977 38:19/0 = 0
978 39:19/0 = 0
979 40:19/0 = 0
980 41:19/0 = 0
981 42:19/0 = 0

982	$43:19/0 = 0$
983	$44:19/0 = 0$
984	$45:19/0 = 0$
985	$46:19/0 = 0$
986	$47:19/0 = 0$
987	$48:19/0 = 0$
988	$0:20/0 = 0$
989	$1:20/0 = 0$
990	$2:20/0 = 0$
991	$3:20/0 = 0$
992	$4:20/0 = 0$
993	$5:20/0 = 0$
994	$6:20/0 = 0$
995	$7:20/0 = 0$
996	$8:20/0 = 0$
997	$9:20/0 = 0$
998	$10:20/0 = 0$
999	$11:20/0 = 0$
1000	$12:20/0 = 0$
1001	$13:20/0 = 0$
1002	$14:20/0 = 0$
1003	$15:20/0 = 0$
1004	$16:20/0 = 0$
1005	$17:20/0 = 0$
1006	$18:20/0 = 0$
1007	$19:20/0 = 0$
1008	$20:20/0 = 0$
1009	$21:20/0 = 0$
1010	$22:20/0 = 0$
1011	$23:20/0 = 0$
1012	$24:20/0 = 0$
1013	$25:20/0 = 0$
1014	$26:20/0 = 0$

1015	$27:20/0 = 0$
1016	$28:20/0 = 0$
1017	$29:20/0 = 0$
1018	$30:20/0 = 0$
1019	$31:20/0 = 0$
1020	$32:20/0 = 0$
1021	$33:20/0 = 0$
1022	$34:20/0 = 0$
1023	$35:20/0 = 0$
1024	$36:20/0 = 0$
1025	$37:20/0 = 0$
1026	$38:20/0 = 0$
1027	$39:20/0 = 0$
1028	$40:20/0 = 0$
1029	$41:20/0 = 0$
1030	$42:20/0 = 0$
1031	$43:20/0 = 0$
1032	$44:20/0 = 0$
1033	$45:20/0 = 0$
1034	$46:20/0 = 0$
1035	$47:20/0 = 0$
1036	$48:20/0 = 0$
1037	$0:21/0 = 0$
1038	$1:21/0 = 0$
1039	$2:21/0 = 0$
1040	$3:21/0 = 0$
1041	$4:21/0 = 0$
1042	$5:21/0 = 0$
1043	$6:21/0 = 0$
1044	$7:21/0 = 0$
1045	$8:21/0 = 0$
1046	$9:21/0 = 0$
1047	$10:21/0 = 0$

1048	$11:21/0 = 0$
1049	$12:21/0 = 0$
1050	$13:21/0 = 0$
1051	$14:21/0 = 0$
1052	$15:21/0 = 0$
1053	$16:21/0 = 0$
1054	$17:21/0 = 0$
1055	$18:21/0 = 0$
1056	$19:21/0 = 0$
1057	$20:21/0 = 0$
1058	$21:21/0 = 0$
1059	$22:21/0 = 0$
1060	$23:21/0 = 0$
1061	$24:21/0 = 0$
1062	$25:21/0 = 0$
1063	$26:21/0 = 0$
1064	$27:21/0 = 0$
1065	$28:21/0 = 0$
1066	$29:21/0 = 0$
1067	$30:21/0 = 0$
1068	$31:21/0 = 0$
1069	$32:21/0 = 0$
1070	$33:21/0 = 0$
1071	$34:21/0 = 0$
1072	$35:21/0 = 0$
1073	$36:21/0 = 0$
1074	$37:21/0 = 0$
1075	$38:21/0 = 0$
1076	$39:21/0 = 0$
1077	$40:21/0 = 0$
1078	$41:21/0 = 0$
1079	$42:21/0 = 0$
1080	$43:21/0 = 0$

```

1081 44:21/0 = 0
1082 45:21/0 = 0
1083 46:21/0 = 0
1084 47:21/0 = 0
1085 48:21/0 = 0
1086
1087 [sub_resource type="TileSet" id="TileSet_3drs5"]
1088 sources/0 = SubResource("TileSetAtlasSource_6h0bd")
1089
1090 [node name="TileMap" type="TileMap"]
1091 tile_set = SubResource("TileSet_3drs5")
1092 format = 2
1093 script = ExtResource("2_lf4lw")

```

C.4.5 tile_map.gd

```

1  extends TileMap
2
3  var points: Array[Dictionary] = []
4  const EUCLIDEAN: String = "Euclidean distance"
5  const MANHATTAN: String = "Manhattan distance"
6  @export_enum(EUCLIDEAN, MANHATTAN) var distance: String = MANHATTAN
7  @export_range(2, 6, 1) var random_starting_points: int = 4
8  var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
9  var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
10
11 # Called when the node enters the scene tree for the first time.
12 func _ready() -> void:
13     randomize()
14     for x in range(-50, x_tile_range + 50):

```

```

15         for y in range(-50, y_tile_range + 50):
16             set_cell(0, Vector2(x, y), 0, Vector2(0, 0))
17         var start_time: float = Time.get_ticks_msec()
18         define_points(random_starting_points)
19         paint_points()
20         var new_time: float = Time.get_ticks_msec() - start_time
21         print("Time taken: " + str(new_time) + "ms")
22
23     func paint_points() -> void:
24         for point in points:
25             set_cell(0, Vector2(point["x"], point["y"]), 0, point["type"]
26                 ])
27             for citizen in point["citizens"]:
28                 set_cell(0, Vector2(point["x"] + citizen["dx"], point["y"]
29                     + citizen["dy"]), 0, point["type"])
30
31     func _squared(x: int) -> int:
32         return x ** 2
33
34     func calculate_points_delta(x: int, y: int, p: int) -> float:
35         if distance == EUCLIDEAN:
36             return sqrt(_squared(points[p]["x"] - x) + _squared(points[p]
37                 )["y"] - y))
38         return abs(points[p]["x"] - x) + abs(points[p]["y"] - y)
39
40     func define_points(num_points: int) -> void:
41         var types: Array[Vector2i] = [Vector2i(0,1),Vector2i(1,1),
42             Vector2i(2,1),Vector2i(5,1),Vector2i(6,1),Vector2i(4,2)]
43         for i in range(num_points):
44             var x: int = randi_range(0, x_tile_range)
45             var y: int = randi_range(0, y_tile_range)
46             var type: Vector2i = types.pick_random()
47             types.erase(type)

```

```

44     points.append(
45         {
46             "type": type,
47             "x": x,
48             "y": y,
49             "citizens": []
50         }
51     )
52     for x in range(x_tile_range):
53         for y in range(y_tile_range):
54             var lowest_delta: Dictionary = {
55                 "point_id": 0,
56                 "delta": x_tile_range * y_tile_range
57             }
58             for p in range(len(points)):
59                 var delta: float = calculate_points_delta(x, y, p)
60                 if delta < lowest_delta["delta"]:
61                     lowest_delta = {
62                         "point_id": p,
63                         "delta": delta
64                     }
65                 var active_point: Dictionary = points[lowest_delta["
66                     point_id"]]
67                 var dx: int = x - active_point["x"]
68                 var dy: int = y - active_point["y"]
69                 active_point["citizens"].append(
70                     {
71                         "dx": dx,
72                         "dy": dy
73                     }
74                 )

```

C.4.6 icon.svg.import

```
1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://du4v6taw8ssax"
6  path="res://.godot/imported/icon.svg-218
      a8f2b3041327d8a5756f3a245f83b.ctex"
7  metadata={
8    "vram_texture": false
9  }
10
11  [deps]
12
13  source_file="res://icon.svg"
14  dest_files=["res://.godot/imported/icon.svg-218
      a8f2b3041327d8a5756f3a245f83b.ctex"]
15
16  [params]
17
18  compress/mode=0
19  compress/high_quality=false
20  compress/lossy_quality=0.7
21  compress/hdr_compression=1
22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
```

```

30 process/normal_map_invert_y=false
31 process/hdr_as_srgb=false
32 process/hdr_clamp_exposure=false
33 process/size_limit=0
34 detect_3d/compress_to=1
35 svg/scale=1.0
36 editor/scale_with_editor_scale=false
37 editor/convert_colors_with_editor_theme=false

```

C.4.7 monochrome_packed.png.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://cpign73sfbsrt"
6  path="res://.godot/imported/monochrome_packed.png-6
      b9bd1c64dd50f72acd3afd14d1ac34f.ctex"
7  metadata={
8    "vram_texture": false
9  }
10
11  [deps]
12
13  source_file="res://monochrome_packed.png"
14  dest_files=["res://.godot/imported/monochrome_packed.png-6
      b9bd1c64dd50f72acd3afd14d1ac34f.ctex"]
15
16  [params]
17
18  compress/mode=0
19  compress/high_quality=false

```

```

20  compress/lossy_quality=0.7
21  compress/hdr_compression=1
22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
30  process/normal_map_invert_y=false
31  process/hdr_as_srgb=false
32  process/hdr_clamp_exposure=false
33  process/size_limit=0
34  detect_3d/compress_to=1

```

C.5 PoissonGD4

C.5.1 .gitattributes

```

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

```

C.5.2 .gitignore

```

1  # Godot 4+ specific ignores
2  .godot/

```

C.5.3 project.godot


```

1   ; Engine configuration file.
2   ; It's best edited using the editor UI and not directly,
3   ; since the parameters that go here are not all obvious.
4   ;
5   ; Format:
6   ;   [section] ; section goes between []
7   ;   param=value ; assign values to parameters
8
9   config_version=5
10
11  [application]
12
13  config/name="Poisson Sampling Project"
14  run/main_scene="res://tile_map.tscn"
15  config/features=PackedStringArray("4.0", "Forward Plus")
16  config/icon="res://icon.svg"
17
18  [display]
19
20  window/size/viewport_height=640
21
22  [rendering]
23
24  renderer/rendering_method="gl_compatibility"

```

C.5.4 tile_map.tscn

```

1  [gd_scene load_steps=5 format=3 uid="uid://f2kv7fett7"]
2
3  [ext_resource type="Texture2D" uid="uid://c3bpsm4r8t504" path="res
      ://monochrome_packed.png" id="1_uucm3"]
4  [ext_resource type="Script" path="res://tile_map.gd" id="2_iyhvf"]

```

```

5
6   [sub_resource type="TileSetAtlasSource" id="
      TileSetAtlasSource_j4usm"]
7   texture = ExtResource("1_uucm3")
8   0:0/0 = 0
9   1:0/0 = 0
10  2:0/0 = 0
11  3:0/0 = 0
12  4:0/0 = 0
13  5:0/0 = 0
14  6:0/0 = 0
15  7:0/0 = 0
16  8:0/0 = 0
17  9:0/0 = 0
18  10:0/0 = 0
19  11:0/0 = 0
20  12:0/0 = 0
21  13:0/0 = 0
22  14:0/0 = 0
23  15:0/0 = 0
24  16:0/0 = 0
25  17:0/0 = 0
26  18:0/0 = 0
27  19:0/0 = 0
28  20:0/0 = 0
29  21:0/0 = 0
30  22:0/0 = 0
31  23:0/0 = 0
32  24:0/0 = 0
33  25:0/0 = 0
34  26:0/0 = 0
35  27:0/0 = 0
36  28:0/0 = 0

```

37 29:0/0 = 0
38 30:0/0 = 0
39 31:0/0 = 0
40 32:0/0 = 0
41 33:0/0 = 0
42 34:0/0 = 0
43 35:0/0 = 0
44 36:0/0 = 0
45 37:0/0 = 0
46 38:0/0 = 0
47 39:0/0 = 0
48 40:0/0 = 0
49 41:0/0 = 0
50 42:0/0 = 0
51 43:0/0 = 0
52 44:0/0 = 0
53 45:0/0 = 0
54 46:0/0 = 0
55 47:0/0 = 0
56 48:0/0 = 0
57 0:1/0 = 0
58 1:1/0 = 0
59 2:1/0 = 0
60 3:1/0 = 0
61 4:1/0 = 0
62 5:1/0 = 0
63 6:1/0 = 0
64 7:1/0 = 0
65 8:1/0 = 0
66 9:1/0 = 0
67 10:1/0 = 0
68 11:1/0 = 0
69 12:1/0 = 0

70 13:1/0 = 0
71 14:1/0 = 0
72 15:1/0 = 0
73 16:1/0 = 0
74 17:1/0 = 0
75 18:1/0 = 0
76 19:1/0 = 0
77 20:1/0 = 0
78 21:1/0 = 0
79 22:1/0 = 0
80 23:1/0 = 0
81 24:1/0 = 0
82 25:1/0 = 0
83 26:1/0 = 0
84 27:1/0 = 0
85 28:1/0 = 0
86 29:1/0 = 0
87 30:1/0 = 0
88 31:1/0 = 0
89 32:1/0 = 0
90 33:1/0 = 0
91 34:1/0 = 0
92 35:1/0 = 0
93 36:1/0 = 0
94 37:1/0 = 0
95 38:1/0 = 0
96 39:1/0 = 0
97 40:1/0 = 0
98 41:1/0 = 0
99 42:1/0 = 0
100 43:1/0 = 0
101 44:1/0 = 0
102 45:1/0 = 0

103 $46:1/0 = 0$
104 $47:1/0 = 0$
105 $48:1/0 = 0$
106 $0:2/0 = 0$
107 $1:2/0 = 0$
108 $2:2/0 = 0$
109 $3:2/0 = 0$
110 $4:2/0 = 0$
111 $5:2/0 = 0$
112 $6:2/0 = 0$
113 $7:2/0 = 0$
114 $8:2/0 = 0$
115 $9:2/0 = 0$
116 $10:2/0 = 0$
117 $11:2/0 = 0$
118 $12:2/0 = 0$
119 $13:2/0 = 0$
120 $14:2/0 = 0$
121 $15:2/0 = 0$
122 $16:2/0 = 0$
123 $17:2/0 = 0$
124 $18:2/0 = 0$
125 $19:2/0 = 0$
126 $20:2/0 = 0$
127 $21:2/0 = 0$
128 $22:2/0 = 0$
129 $23:2/0 = 0$
130 $24:2/0 = 0$
131 $25:2/0 = 0$
132 $26:2/0 = 0$
133 $27:2/0 = 0$
134 $28:2/0 = 0$
135 $29:2/0 = 0$

136 30:2/0 = 0
137 31:2/0 = 0
138 32:2/0 = 0
139 33:2/0 = 0
140 34:2/0 = 0
141 35:2/0 = 0
142 36:2/0 = 0
143 37:2/0 = 0
144 38:2/0 = 0
145 39:2/0 = 0
146 40:2/0 = 0
147 41:2/0 = 0
148 42:2/0 = 0
149 43:2/0 = 0
150 44:2/0 = 0
151 45:2/0 = 0
152 46:2/0 = 0
153 47:2/0 = 0
154 48:2/0 = 0
155 0:3/0 = 0
156 1:3/0 = 0
157 2:3/0 = 0
158 3:3/0 = 0
159 4:3/0 = 0
160 5:3/0 = 0
161 6:3/0 = 0
162 7:3/0 = 0
163 8:3/0 = 0
164 9:3/0 = 0
165 10:3/0 = 0
166 11:3/0 = 0
167 12:3/0 = 0
168 13:3/0 = 0

169	$14:3/0 = 0$
170	$15:3/0 = 0$
171	$16:3/0 = 0$
172	$17:3/0 = 0$
173	$18:3/0 = 0$
174	$19:3/0 = 0$
175	$20:3/0 = 0$
176	$21:3/0 = 0$
177	$22:3/0 = 0$
178	$23:3/0 = 0$
179	$24:3/0 = 0$
180	$25:3/0 = 0$
181	$26:3/0 = 0$
182	$27:3/0 = 0$
183	$28:3/0 = 0$
184	$29:3/0 = 0$
185	$30:3/0 = 0$
186	$31:3/0 = 0$
187	$32:3/0 = 0$
188	$33:3/0 = 0$
189	$34:3/0 = 0$
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	$18:4/0 = 0$
223	$19:4/0 = 0$
224	$20:4/0 = 0$
225	$21:4/0 = 0$
226	$22:4/0 = 0$
227	$23:4/0 = 0$
228	$24:4/0 = 0$
229	$25:4/0 = 0$
230	$26:4/0 = 0$
231	$27:4/0 = 0$
232	$28:4/0 = 0$
233	$29:4/0 = 0$
234	$30:4/0 = 0$

235 31:4/0 = 0
236 32:4/0 = 0
237 33:4/0 = 0
238 34:4/0 = 0
239 35:4/0 = 0
240 36:4/0 = 0
241 37:4/0 = 0
242 38:4/0 = 0
243 39:4/0 = 0
244 40:4/0 = 0
245 41:4/0 = 0
246 42:4/0 = 0
247 43:4/0 = 0
248 44:4/0 = 0
249 45:4/0 = 0
250 46:4/0 = 0
251 47:4/0 = 0
252 48:4/0 = 0
253 0:5/0 = 0
254 1:5/0 = 0
255 2:5/0 = 0
256 3:5/0 = 0
257 4:5/0 = 0
258 5:5/0 = 0
259 6:5/0 = 0
260 7:5/0 = 0
261 8:5/0 = 0
262 9:5/0 = 0
263 10:5/0 = 0
264 11:5/0 = 0
265 12:5/0 = 0
266 13:5/0 = 0
267 14:5/0 = 0

268	$15:5/0 = 0$
269	$16:5/0 = 0$
270	$17:5/0 = 0$
271	$18:5/0 = 0$
272	$19:5/0 = 0$
273	$20:5/0 = 0$
274	$21:5/0 = 0$
275	$22:5/0 = 0$
276	$23:5/0 = 0$
277	$24:5/0 = 0$
278	$25:5/0 = 0$
279	$26:5/0 = 0$
280	$27:5/0 = 0$
281	$28:5/0 = 0$
282	$29:5/0 = 0$
283	$30:5/0 = 0$
284	$31:5/0 = 0$
285	$32:5/0 = 0$
286	$33:5/0 = 0$
287	$34:5/0 = 0$
288	$35:5/0 = 0$
289	$36:5/0 = 0$
290	$37:5/0 = 0$
291	$38:5/0 = 0$
292	$39:5/0 = 0$
293	$40:5/0 = 0$
294	$41:5/0 = 0$
295	$42:5/0 = 0$
296	$43:5/0 = 0$
297	$44:5/0 = 0$
298	$45:5/0 = 0$
299	$46:5/0 = 0$
300	$47:5/0 = 0$

301 48:5/0 = 0
302 0:6/0 = 0
303 1:6/0 = 0
304 2:6/0 = 0
305 3:6/0 = 0
306 4:6/0 = 0
307 5:6/0 = 0
308 6:6/0 = 0
309 7:6/0 = 0
310 8:6/0 = 0
311 9:6/0 = 0
312 10:6/0 = 0
313 11:6/0 = 0
314 12:6/0 = 0
315 13:6/0 = 0
316 14:6/0 = 0
317 15:6/0 = 0
318 16:6/0 = 0
319 17:6/0 = 0
320 18:6/0 = 0
321 19:6/0 = 0
322 20:6/0 = 0
323 21:6/0 = 0
324 22:6/0 = 0
325 23:6/0 = 0
326 24:6/0 = 0
327 25:6/0 = 0
328 26:6/0 = 0
329 27:6/0 = 0
330 28:6/0 = 0
331 29:6/0 = 0
332 30:6/0 = 0
333 31:6/0 = 0

334 32:6/0 = 0
335 33:6/0 = 0
336 34:6/0 = 0
337 35:6/0 = 0
338 36:6/0 = 0
339 37:6/0 = 0
340 38:6/0 = 0
341 39:6/0 = 0
342 40:6/0 = 0
343 41:6/0 = 0
344 42:6/0 = 0
345 43:6/0 = 0
346 44:6/0 = 0
347 45:6/0 = 0
348 46:6/0 = 0
349 47:6/0 = 0
350 48:6/0 = 0
351 0:7/0 = 0
352 1:7/0 = 0
353 2:7/0 = 0
354 3:7/0 = 0
355 4:7/0 = 0
356 5:7/0 = 0
357 6:7/0 = 0
358 7:7/0 = 0
359 8:7/0 = 0
360 9:7/0 = 0
361 10:7/0 = 0
362 11:7/0 = 0
363 12:7/0 = 0
364 13:7/0 = 0
365 14:7/0 = 0
366 15:7/0 = 0

367 16:7/0 = 0
368 17:7/0 = 0
369 18:7/0 = 0
370 19:7/0 = 0
371 20:7/0 = 0
372 21:7/0 = 0
373 22:7/0 = 0
374 23:7/0 = 0
375 24:7/0 = 0
376 25:7/0 = 0
377 26:7/0 = 0
378 27:7/0 = 0
379 28:7/0 = 0
380 29:7/0 = 0
381 30:7/0 = 0
382 31:7/0 = 0
383 32:7/0 = 0
384 33:7/0 = 0
385 34:7/0 = 0
386 35:7/0 = 0
387 36:7/0 = 0
388 37:7/0 = 0
389 38:7/0 = 0
390 39:7/0 = 0
391 40:7/0 = 0
392 41:7/0 = 0
393 42:7/0 = 0
394 43:7/0 = 0
395 44:7/0 = 0
396 45:7/0 = 0
397 46:7/0 = 0
398 47:7/0 = 0
399 48:7/0 = 0

400	$0:8/0 = 0$
401	$1:8/0 = 0$
402	$2:8/0 = 0$
403	$3:8/0 = 0$
404	$4:8/0 = 0$
405	$5:8/0 = 0$
406	$6:8/0 = 0$
407	$7:8/0 = 0$
408	$8:8/0 = 0$
409	$9:8/0 = 0$
410	$10:8/0 = 0$
411	$11:8/0 = 0$
412	$12:8/0 = 0$
413	$13:8/0 = 0$
414	$14:8/0 = 0$
415	$15:8/0 = 0$
416	$16:8/0 = 0$
417	$17:8/0 = 0$
418	$18:8/0 = 0$
419	$19:8/0 = 0$
420	$20:8/0 = 0$
421	$21:8/0 = 0$
422	$22:8/0 = 0$
423	$23:8/0 = 0$
424	$24:8/0 = 0$
425	$25:8/0 = 0$
426	$26:8/0 = 0$
427	$27:8/0 = 0$
428	$28:8/0 = 0$
429	$29:8/0 = 0$
430	$30:8/0 = 0$
431	$31:8/0 = 0$
432	$32:8/0 = 0$

433 33:8/0 = 0
434 34:8/0 = 0
435 35:8/0 = 0
436 36:8/0 = 0
437 37:8/0 = 0
438 38:8/0 = 0
439 39:8/0 = 0
440 40:8/0 = 0
441 41:8/0 = 0
442 42:8/0 = 0
443 43:8/0 = 0
444 44:8/0 = 0
445 45:8/0 = 0
446 46:8/0 = 0
447 47:8/0 = 0
448 48:8/0 = 0
449 0:9/0 = 0
450 1:9/0 = 0
451 2:9/0 = 0
452 3:9/0 = 0
453 4:9/0 = 0
454 5:9/0 = 0
455 6:9/0 = 0
456 7:9/0 = 0
457 8:9/0 = 0
458 9:9/0 = 0
459 10:9/0 = 0
460 11:9/0 = 0
461 12:9/0 = 0
462 13:9/0 = 0
463 14:9/0 = 0
464 15:9/0 = 0
465 16:9/0 = 0

466 17:9/0 = 0
467 18:9/0 = 0
468 19:9/0 = 0
469 20:9/0 = 0
470 21:9/0 = 0
471 22:9/0 = 0
472 23:9/0 = 0
473 24:9/0 = 0
474 25:9/0 = 0
475 26:9/0 = 0
476 27:9/0 = 0
477 28:9/0 = 0
478 29:9/0 = 0
479 30:9/0 = 0
480 31:9/0 = 0
481 32:9/0 = 0
482 33:9/0 = 0
483 34:9/0 = 0
484 35:9/0 = 0
485 36:9/0 = 0
486 37:9/0 = 0
487 38:9/0 = 0
488 39:9/0 = 0
489 40:9/0 = 0
490 41:9/0 = 0
491 42:9/0 = 0
492 43:9/0 = 0
493 44:9/0 = 0
494 45:9/0 = 0
495 46:9/0 = 0
496 47:9/0 = 0
497 48:9/0 = 0
498 0:10/0 = 0

499 1:10/0 = 0
500 2:10/0 = 0
501 3:10/0 = 0
502 4:10/0 = 0
503 5:10/0 = 0
504 6:10/0 = 0
505 7:10/0 = 0
506 8:10/0 = 0
507 9:10/0 = 0
508 10:10/0 = 0
509 11:10/0 = 0
510 12:10/0 = 0
511 13:10/0 = 0
512 14:10/0 = 0
513 15:10/0 = 0
514 16:10/0 = 0
515 17:10/0 = 0
516 18:10/0 = 0
517 19:10/0 = 0
518 20:10/0 = 0
519 21:10/0 = 0
520 22:10/0 = 0
521 23:10/0 = 0
522 24:10/0 = 0
523 25:10/0 = 0
524 26:10/0 = 0
525 27:10/0 = 0
526 28:10/0 = 0
527 29:10/0 = 0
528 30:10/0 = 0
529 31:10/0 = 0
530 32:10/0 = 0
531 33:10/0 = 0

532 34:10/0 = 0
533 35:10/0 = 0
534 36:10/0 = 0
535 37:10/0 = 0
536 38:10/0 = 0
537 39:10/0 = 0
538 40:10/0 = 0
539 41:10/0 = 0
540 42:10/0 = 0
541 43:10/0 = 0
542 44:10/0 = 0
543 45:10/0 = 0
544 46:10/0 = 0
545 47:10/0 = 0
546 48:10/0 = 0
547 0:11/0 = 0
548 1:11/0 = 0
549 2:11/0 = 0
550 3:11/0 = 0
551 4:11/0 = 0
552 5:11/0 = 0
553 6:11/0 = 0
554 7:11/0 = 0
555 8:11/0 = 0
556 9:11/0 = 0
557 10:11/0 = 0
558 11:11/0 = 0
559 12:11/0 = 0
560 13:11/0 = 0
561 14:11/0 = 0
562 15:11/0 = 0
563 16:11/0 = 0
564 17:11/0 = 0

565 18:11/0 = 0
566 19:11/0 = 0
567 20:11/0 = 0
568 21:11/0 = 0
569 22:11/0 = 0
570 23:11/0 = 0
571 24:11/0 = 0
572 25:11/0 = 0
573 26:11/0 = 0
574 27:11/0 = 0
575 28:11/0 = 0
576 29:11/0 = 0
577 30:11/0 = 0
578 31:11/0 = 0
579 32:11/0 = 0
580 33:11/0 = 0
581 34:11/0 = 0
582 35:11/0 = 0
583 36:11/0 = 0
584 37:11/0 = 0
585 38:11/0 = 0
586 39:11/0 = 0
587 40:11/0 = 0
588 41:11/0 = 0
589 42:11/0 = 0
590 43:11/0 = 0
591 44:11/0 = 0
592 45:11/0 = 0
593 46:11/0 = 0
594 47:11/0 = 0
595 48:11/0 = 0
596 0:12/0 = 0
597 1:12/0 = 0

598 2:12/0 = 0
599 3:12/0 = 0
600 4:12/0 = 0
601 5:12/0 = 0
602 6:12/0 = 0
603 7:12/0 = 0
604 8:12/0 = 0
605 9:12/0 = 0
606 10:12/0 = 0
607 11:12/0 = 0
608 12:12/0 = 0
609 13:12/0 = 0
610 14:12/0 = 0
611 15:12/0 = 0
612 16:12/0 = 0
613 17:12/0 = 0
614 18:12/0 = 0
615 19:12/0 = 0
616 20:12/0 = 0
617 21:12/0 = 0
618 22:12/0 = 0
619 23:12/0 = 0
620 24:12/0 = 0
621 25:12/0 = 0
622 26:12/0 = 0
623 27:12/0 = 0
624 28:12/0 = 0
625 29:12/0 = 0
626 30:12/0 = 0
627 31:12/0 = 0
628 32:12/0 = 0
629 33:12/0 = 0
630 34:12/0 = 0

631 35:12/0 = 0
632 36:12/0 = 0
633 37:12/0 = 0
634 38:12/0 = 0
635 39:12/0 = 0
636 40:12/0 = 0
637 41:12/0 = 0
638 42:12/0 = 0
639 43:12/0 = 0
640 44:12/0 = 0
641 45:12/0 = 0
642 46:12/0 = 0
643 47:12/0 = 0
644 48:12/0 = 0
645 0:13/0 = 0
646 1:13/0 = 0
647 2:13/0 = 0
648 3:13/0 = 0
649 4:13/0 = 0
650 5:13/0 = 0
651 6:13/0 = 0
652 7:13/0 = 0
653 8:13/0 = 0
654 9:13/0 = 0
655 10:13/0 = 0
656 11:13/0 = 0
657 12:13/0 = 0
658 13:13/0 = 0
659 14:13/0 = 0
660 15:13/0 = 0
661 16:13/0 = 0
662 17:13/0 = 0
663 18:13/0 = 0

664 19:13/0 = 0
665 20:13/0 = 0
666 21:13/0 = 0
667 22:13/0 = 0
668 23:13/0 = 0
669 24:13/0 = 0
670 25:13/0 = 0
671 26:13/0 = 0
672 27:13/0 = 0
673 28:13/0 = 0
674 29:13/0 = 0
675 30:13/0 = 0
676 31:13/0 = 0
677 32:13/0 = 0
678 33:13/0 = 0
679 34:13/0 = 0
680 35:13/0 = 0
681 36:13/0 = 0
682 37:13/0 = 0
683 38:13/0 = 0
684 39:13/0 = 0
685 40:13/0 = 0
686 41:13/0 = 0
687 42:13/0 = 0
688 43:13/0 = 0
689 44:13/0 = 0
690 45:13/0 = 0
691 46:13/0 = 0
692 47:13/0 = 0
693 48:13/0 = 0
694 0:14/0 = 0
695 1:14/0 = 0
696 2:14/0 = 0

697 3:14/0 = 0
698 4:14/0 = 0
699 5:14/0 = 0
700 6:14/0 = 0
701 7:14/0 = 0
702 8:14/0 = 0
703 9:14/0 = 0
704 10:14/0 = 0
705 11:14/0 = 0
706 12:14/0 = 0
707 13:14/0 = 0
708 14:14/0 = 0
709 15:14/0 = 0
710 16:14/0 = 0
711 17:14/0 = 0
712 18:14/0 = 0
713 19:14/0 = 0
714 20:14/0 = 0
715 21:14/0 = 0
716 22:14/0 = 0
717 23:14/0 = 0
718 24:14/0 = 0
719 25:14/0 = 0
720 26:14/0 = 0
721 27:14/0 = 0
722 28:14/0 = 0
723 29:14/0 = 0
724 30:14/0 = 0
725 31:14/0 = 0
726 32:14/0 = 0
727 33:14/0 = 0
728 34:14/0 = 0
729 35:14/0 = 0

730 36:14/0 = 0
731 37:14/0 = 0
732 38:14/0 = 0
733 39:14/0 = 0
734 40:14/0 = 0
735 41:14/0 = 0
736 42:14/0 = 0
737 43:14/0 = 0
738 44:14/0 = 0
739 45:14/0 = 0
740 46:14/0 = 0
741 47:14/0 = 0
742 48:14/0 = 0
743 0:15/0 = 0
744 1:15/0 = 0
745 2:15/0 = 0
746 3:15/0 = 0
747 4:15/0 = 0
748 5:15/0 = 0
749 6:15/0 = 0
750 7:15/0 = 0
751 8:15/0 = 0
752 9:15/0 = 0
753 10:15/0 = 0
754 11:15/0 = 0
755 12:15/0 = 0
756 13:15/0 = 0
757 14:15/0 = 0
758 15:15/0 = 0
759 16:15/0 = 0
760 17:15/0 = 0
761 18:15/0 = 0
762 19:15/0 = 0

763 20:15/0 = 0
764 21:15/0 = 0
765 22:15/0 = 0
766 23:15/0 = 0
767 24:15/0 = 0
768 25:15/0 = 0
769 26:15/0 = 0
770 27:15/0 = 0
771 28:15/0 = 0
772 29:15/0 = 0
773 30:15/0 = 0
774 31:15/0 = 0
775 32:15/0 = 0
776 33:15/0 = 0
777 34:15/0 = 0
778 35:15/0 = 0
779 36:15/0 = 0
780 37:15/0 = 0
781 38:15/0 = 0
782 39:15/0 = 0
783 40:15/0 = 0
784 41:15/0 = 0
785 42:15/0 = 0
786 43:15/0 = 0
787 44:15/0 = 0
788 45:15/0 = 0
789 46:15/0 = 0
790 47:15/0 = 0
791 48:15/0 = 0
792 0:16/0 = 0
793 1:16/0 = 0
794 2:16/0 = 0
795 3:16/0 = 0

796 4:16/0 = 0
797 5:16/0 = 0
798 6:16/0 = 0
799 7:16/0 = 0
800 8:16/0 = 0
801 9:16/0 = 0
802 10:16/0 = 0
803 11:16/0 = 0
804 12:16/0 = 0
805 13:16/0 = 0
806 14:16/0 = 0
807 15:16/0 = 0
808 16:16/0 = 0
809 17:16/0 = 0
810 18:16/0 = 0
811 19:16/0 = 0
812 20:16/0 = 0
813 21:16/0 = 0
814 22:16/0 = 0
815 23:16/0 = 0
816 24:16/0 = 0
817 25:16/0 = 0
818 26:16/0 = 0
819 27:16/0 = 0
820 28:16/0 = 0
821 29:16/0 = 0
822 30:16/0 = 0
823 31:16/0 = 0
824 32:16/0 = 0
825 33:16/0 = 0
826 34:16/0 = 0
827 35:16/0 = 0
828 36:16/0 = 0

829 37:16/0 = 0
830 38:16/0 = 0
831 39:16/0 = 0
832 40:16/0 = 0
833 41:16/0 = 0
834 42:16/0 = 0
835 43:16/0 = 0
836 44:16/0 = 0
837 45:16/0 = 0
838 46:16/0 = 0
839 47:16/0 = 0
840 48:16/0 = 0
841 0:17/0 = 0
842 1:17/0 = 0
843 2:17/0 = 0
844 3:17/0 = 0
845 4:17/0 = 0
846 5:17/0 = 0
847 6:17/0 = 0
848 7:17/0 = 0
849 8:17/0 = 0
850 9:17/0 = 0
851 10:17/0 = 0
852 11:17/0 = 0
853 12:17/0 = 0
854 13:17/0 = 0
855 14:17/0 = 0
856 15:17/0 = 0
857 16:17/0 = 0
858 17:17/0 = 0
859 18:17/0 = 0
860 19:17/0 = 0
861 20:17/0 = 0

862 21:17/0 = 0
863 22:17/0 = 0
864 23:17/0 = 0
865 24:17/0 = 0
866 25:17/0 = 0
867 26:17/0 = 0
868 27:17/0 = 0
869 28:17/0 = 0
870 29:17/0 = 0
871 30:17/0 = 0
872 31:17/0 = 0
873 32:17/0 = 0
874 33:17/0 = 0
875 34:17/0 = 0
876 35:17/0 = 0
877 36:17/0 = 0
878 37:17/0 = 0
879 38:17/0 = 0
880 39:17/0 = 0
881 40:17/0 = 0
882 41:17/0 = 0
883 42:17/0 = 0
884 43:17/0 = 0
885 44:17/0 = 0
886 45:17/0 = 0
887 46:17/0 = 0
888 47:17/0 = 0
889 48:17/0 = 0
890 0:18/0 = 0
891 1:18/0 = 0
892 2:18/0 = 0
893 3:18/0 = 0
894 4:18/0 = 0

895 5:18/0 = 0
896 6:18/0 = 0
897 7:18/0 = 0
898 8:18/0 = 0
899 9:18/0 = 0
900 10:18/0 = 0
901 11:18/0 = 0
902 12:18/0 = 0
903 13:18/0 = 0
904 14:18/0 = 0
905 15:18/0 = 0
906 16:18/0 = 0
907 17:18/0 = 0
908 18:18/0 = 0
909 19:18/0 = 0
910 20:18/0 = 0
911 21:18/0 = 0
912 22:18/0 = 0
913 23:18/0 = 0
914 24:18/0 = 0
915 25:18/0 = 0
916 26:18/0 = 0
917 27:18/0 = 0
918 28:18/0 = 0
919 29:18/0 = 0
920 30:18/0 = 0
921 31:18/0 = 0
922 32:18/0 = 0
923 33:18/0 = 0
924 34:18/0 = 0
925 35:18/0 = 0
926 36:18/0 = 0
927 37:18/0 = 0

928 38:18/0 = 0
929 39:18/0 = 0
930 40:18/0 = 0
931 41:18/0 = 0
932 42:18/0 = 0
933 43:18/0 = 0
934 44:18/0 = 0
935 45:18/0 = 0
936 46:18/0 = 0
937 47:18/0 = 0
938 48:18/0 = 0
939 0:19/0 = 0
940 1:19/0 = 0
941 2:19/0 = 0
942 3:19/0 = 0
943 4:19/0 = 0
944 5:19/0 = 0
945 6:19/0 = 0
946 7:19/0 = 0
947 8:19/0 = 0
948 9:19/0 = 0
949 10:19/0 = 0
950 11:19/0 = 0
951 12:19/0 = 0
952 13:19/0 = 0
953 14:19/0 = 0
954 15:19/0 = 0
955 16:19/0 = 0
956 17:19/0 = 0
957 18:19/0 = 0
958 19:19/0 = 0
959 20:19/0 = 0
960 21:19/0 = 0

961 22:19/0 = 0
962 23:19/0 = 0
963 24:19/0 = 0
964 25:19/0 = 0
965 26:19/0 = 0
966 27:19/0 = 0
967 28:19/0 = 0
968 29:19/0 = 0
969 30:19/0 = 0
970 31:19/0 = 0
971 32:19/0 = 0
972 33:19/0 = 0
973 34:19/0 = 0
974 35:19/0 = 0
975 36:19/0 = 0
976 37:19/0 = 0
977 38:19/0 = 0
978 39:19/0 = 0
979 40:19/0 = 0
980 41:19/0 = 0
981 42:19/0 = 0
982 43:19/0 = 0
983 44:19/0 = 0
984 45:19/0 = 0
985 46:19/0 = 0
986 47:19/0 = 0
987 48:19/0 = 0
988 0:20/0 = 0
989 1:20/0 = 0
990 2:20/0 = 0
991 3:20/0 = 0
992 4:20/0 = 0
993 5:20/0 = 0

994 6:20/0 = 0
995 7:20/0 = 0
996 8:20/0 = 0
997 9:20/0 = 0
998 10:20/0 = 0
999 11:20/0 = 0
1000 12:20/0 = 0
1001 13:20/0 = 0
1002 14:20/0 = 0
1003 15:20/0 = 0
1004 16:20/0 = 0
1005 17:20/0 = 0
1006 18:20/0 = 0
1007 19:20/0 = 0
1008 20:20/0 = 0
1009 21:20/0 = 0
1010 22:20/0 = 0
1011 23:20/0 = 0
1012 24:20/0 = 0
1013 25:20/0 = 0
1014 26:20/0 = 0
1015 27:20/0 = 0
1016 28:20/0 = 0
1017 29:20/0 = 0
1018 30:20/0 = 0
1019 31:20/0 = 0
1020 32:20/0 = 0
1021 33:20/0 = 0
1022 34:20/0 = 0
1023 35:20/0 = 0
1024 36:20/0 = 0
1025 37:20/0 = 0
1026 38:20/0 = 0

1027	$39:20/0 = 0$
1028	$40:20/0 = 0$
1029	$41:20/0 = 0$
1030	$42:20/0 = 0$
1031	$43:20/0 = 0$
1032	$44:20/0 = 0$
1033	$45:20/0 = 0$
1034	$46:20/0 = 0$
1035	$47:20/0 = 0$
1036	$48:20/0 = 0$
1037	$0:21/0 = 0$
1038	$1:21/0 = 0$
1039	$2:21/0 = 0$
1040	$3:21/0 = 0$
1041	$4:21/0 = 0$
1042	$5:21/0 = 0$
1043	$6:21/0 = 0$
1044	$7:21/0 = 0$
1045	$8:21/0 = 0$
1046	$9:21/0 = 0$
1047	$10:21/0 = 0$
1048	$11:21/0 = 0$
1049	$12:21/0 = 0$
1050	$13:21/0 = 0$
1051	$14:21/0 = 0$
1052	$15:21/0 = 0$
1053	$16:21/0 = 0$
1054	$17:21/0 = 0$
1055	$18:21/0 = 0$
1056	$19:21/0 = 0$
1057	$20:21/0 = 0$
1058	$21:21/0 = 0$
1059	$22:21/0 = 0$

```

1060    23:21/0 = 0
1061    24:21/0 = 0
1062    25:21/0 = 0
1063    26:21/0 = 0
1064    27:21/0 = 0
1065    28:21/0 = 0
1066    29:21/0 = 0
1067    30:21/0 = 0
1068    31:21/0 = 0
1069    32:21/0 = 0
1070    33:21/0 = 0
1071    34:21/0 = 0
1072    35:21/0 = 0
1073    36:21/0 = 0
1074    37:21/0 = 0
1075    38:21/0 = 0
1076    39:21/0 = 0
1077    40:21/0 = 0
1078    41:21/0 = 0
1079    42:21/0 = 0
1080    43:21/0 = 0
1081    44:21/0 = 0
1082    45:21/0 = 0
1083    46:21/0 = 0
1084    47:21/0 = 0
1085    48:21/0 = 0
1086
1087    [sub_resource type="TileSet" id="TileSet_8pb5m"]
1088    sources/0 = SubResource("TileSetAtlasSource_j4usm")
1089
1090    [node name="TileMap" type="TileMap"]
1091    tile_set = SubResource("TileSet_8pb5m")
1092    format = 2

```

```
1093 script = ExtResource("2_iyhvf")
```

C.5.5 tile_map.gd

```
1  extends TileMap
2
3  var cell_points: Array[Vector2]
4  @export var point_radius: float = 1.0
5  @export var region_size: Vector2 = Vector2.ONE
6  @export var rejection_samples: int = 30
7
8  var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
9  var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
10
11 # Called when the node enters the scene tree for the first time.
12 func _ready():
13     randomize()
14     for x in range(-50, x_tile_range + 50):
15         for y in range(-50, y_tile_range + 50):
16             set_cell(0, Vector2(x, y), 0, Vector2(0, 0))
17     cell_points = generate_points(point_radius, region_size,
        rejection_samples)
18
19 func generate_points(radius: float, sample_region_size: Vector2,
    number_of_samples_before_rejection: int = 30) -> Array[Vector2
    ]:
20     var cell_size: float = radius / sqrt(2)
21     var grid: Array[Array] = []
22     var points: Array[Vector2] = []
23     var spawn_points: Array[Vector2] = []
```

```

24
25     spawn_points.append(sample_region_size/2)
26
27     while spawn_points.size() > 0:
28         var spawn_index: int = randi_range(0, spawn_points.size() -
29             1)
30         var spawn_centre: Vector2 = spawn_points[spawn_index]
31         var candidate_accepted: bool = false
32
33         for i in range(number_of_samples_before_rejection):
34             var angle: float = randf_range(0.0, 1.0) * TAU # TAU = PI
35             * 2
36             var direction: Vector2 = Vector2(sin(angle), cos(angle))
37             var candidate: Vector2 = spawn_centre + direction *
38                 randf_range(radius, 2 * radius)
39             if is_valid(candidate, sample_region_size, cell_size,
40                 radius, points, grid):
41                 points.append(candidate)
42                 spawn_points.append(candidate)
43                 grid[int(candidate.x/cell_size)][int(candidate.y/
44                     cell_size)] = len(points)
45                 candidate_accepted = true
46                 break
47
48         if not candidate_accepted:
49             spawn_points.remove_at(spawn_index)
50
51     return points
52
53 func is_valid(candidate: Vector2, sample_region_size: Vector2,
54     cell_size: float, radius: float, points: Array[Vector2], grid:
55     Array[Array]):
56     if candidate.x >= 0 and candidate.x < sample_region_size.x and

```

```

        candidate.y >= 0 and candidate.y < sample_region_size.y:
50     var cell_x: int = candidate.x / cell_size
51     var cell_y: int = candidate.y / cell_size
52     var search_start_x: int = max(0, cell_x - 2)
53     var search_end_x: int = min(cell_x + 2, x_tile_range - 1)
54     var search_start_y: int = max(0, cell_y - 2)
55     var search_end_y: int = min(cell_y + 2, y_tile_range - 1)
56     for x in range(search_start_x, search_end_x):
57         for y in range(search_start_y, search_end_y):
58             var point_index: int = grid[x][y]
59             if point_index != -1:
60                 var distance: float = (candidate - points[
                    point_index]).length()
61                 if distance < radius:
62                     return false
63         return true
64     return false

```

C.5.6 icon.svg.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://uotfe6soknht"
6  path="res://.godot/imported/icon.svg-218
    a8f2b3041327d8a5756f3a245f83b.ctex"
7  metadata={
8  "vram_texture": false
9  }
10
11 [deps]

```

```

12
13   source_file="res://icon.svg"
14   dest_files=["res://.godot/imported/icon.svg-218
           a8f2b3041327d8a5756f3a245f83b.ctex"]
15
16   [params]
17
18   compress/mode=0
19   compress/high_quality=false
20   compress/lossy_quality=0.7
21   compress/hdr_compression=1
22   compress/normal_map=0
23   compress/channel_pack=0
24   mipmaps/generate=false
25   mipmaps/limit=-1
26   roughness/mode=0
27   roughness/src_normal=""
28   process/fix_alpha_border=true
29   process/premult_alpha=false
30   process/normal_map_invert_y=false
31   process/hdr_as_srgb=false
32   process/hdr_clamp_exposure=false
33   process/size_limit=0
34   detect_3d/compress_to=1
35   svg/scale=1.0
36   editor/scale_with_editor_scale=false
37   editor/convert_colors_with_editor_theme=false

```

C.5.7 monochrome_packed.png.import

```

1   [remap]
2

```

```

3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://c3bpsm4r8t504"
6  path="res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"
7  metadata={
8  "vram_texture": false
9  }
10
11  [deps]
12
13  source_file="res://monochrome_packed.png"
14  dest_files=["res://.godot/imported/monochrome_packed.png-6
    b9bd1c64dd50f72acd3afd14d1ac34f.ctex"]
15
16  [params]
17
18  compress/mode=0
19  compress/high_quality=false
20  compress/lossy_quality=0.7
21  compress/hdr_compression=1
22  compress/normal_map=0
23  compress/channel_pack=0
24  mipmaps/generate=false
25  mipmaps/limit=-1
26  roughness/mode=0
27  roughness/src_normal=""
28  process/fix_alpha_border=true
29  process/premult_alpha=false
30  process/normal_map_invert_y=false
31  process/hdr_as_srgb=false
32  process/hdr_clamp_exposure=false
33  process/size_limit=0

```

34 detect_3d/compress_to=1