



6CCS3PRJ Final Year
Implementing Procedural Content
Generation Algorithms in a Tile Map
RPG in the Godot Game Engine

Final Project Report

Author: Zishan Rahman

Supervisor: Senir Dinar

Student ID: 20071291

April 18, 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 18, 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	11
3.2	Implementations	19
4	Design & Specification	28
4.1	Performance	28
4.2	Layouts	29
5	Implementation	30
5.1	Lindenmayer System	30
5.2	Perlin/Simplex Noise	35
5.3	Poisson Disk Sampling	38
5.4	Voronoi Cells	40
6	Legal, Social, Ethical and Professional Issues	44
6.1	Using Other People's Resources	44
6.2	How I Will Release My Own Artefacts	46
7	Results/Evaluation	48
7.1	Software Testing	48
7.2	Comparing the Different Algorithms and Drawing Conclusions on Which Ones Are Best	48
8	Conclusion and Future Work	49
	Bibliography	55
A	Extra Information	56
A.1	Tables, proofs, graphs, test cases,	56

B	User Guide	57
B.1	Instructions	57
C	Source Code	58
C.1	Instructions	58
C.2	LSystemGrammarDemo	59
C.3	ProcGenRPG (L-System)	64
C.4	VoronoiCellsGD4	77
C.5	PoissonGD4	123
C.6	Noise Demo	170

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.[18]

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.[4]



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 (chapter 5), how the level layouts generated in each algorithm compare with each other in the Design & Specification section (chapter 4), and how each implementation compares overall (and also performance wise) in the Evaluation section (chapter 7). For this project, I chose to use the following 4 algorithms.

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

All of the above algorithms are “ontogenetic.” This contrasts with “teleological” procedural generation algorithms, and the difference between them is described in a 2008 article for video games magazine *Gamastura* by Mick West:

“Two competing methodologies in procedural content generation are teleological and ontogenetic. The teleological approach creates an accurate physical model of the environment and the process that creates the thing generated, and then simply runs the simulation, and the results should emerge as they do in nature.

The ontogenetic approach observes the end results of this process and then attempts to directly reproduce those results by ad hoc algorithms. Ontogenetic approaches are more com-

monly used in real-time applications such as games. (See "Shattering Reality,"[sic] Game Developer, August 2006.)"[55][54]

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.[33] 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. I go over how I got my implementation to work with complex multi-rule grammars in Chapters 3.2.2 and 5.1.

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 ϕ .

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:[56]

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

The first 3 iterations of string replacement with this one-rule grammar G are shown here:

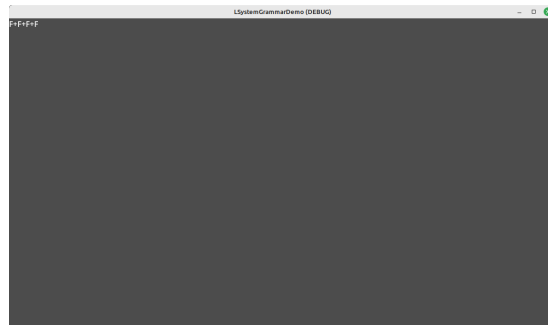


Figure 3.1: The axiom of the aforementioned simple L-System with just one rule. String size: 8.
Source: Own work.



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

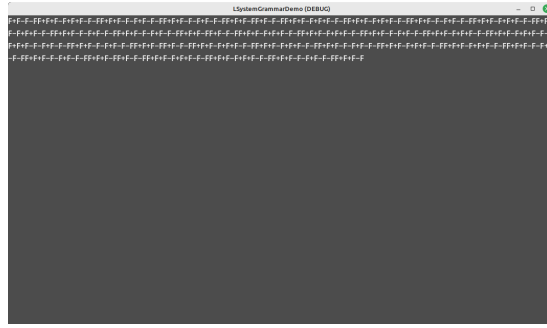


Figure 3.3: The second iteration of the aforementioned simple L-System with just one rule. String size: 475.
Source: Own work.

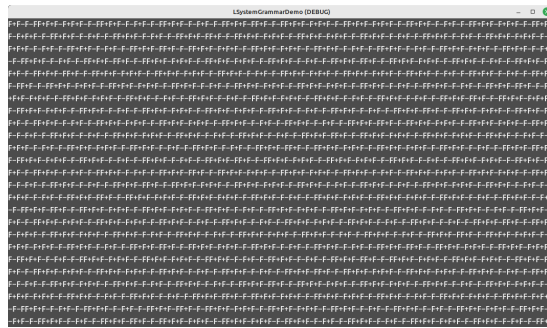


Figure 3.4: The third iteration of the aforementioned simple L-System with just one rule.
String size: 3803. The string is too large to show in the window, as you can see here.
Source: Own work.

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

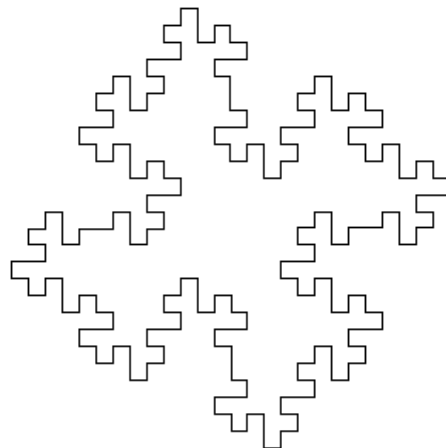


Figure 3.5: A lattice generated with the example grammar on a custom-written Classic Mac OS application specifically written for working with L-Systems.[2]

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[42], 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.[10] Since both tangents between a curve must be collinear[10], 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.[10] This results in noise images of higher and more appealing visual quality as, to quote a response from Stack Exchange by Hernan J. González[25], "it cuts low frequencies and emphasizes frequencies around and above the grid spacing."

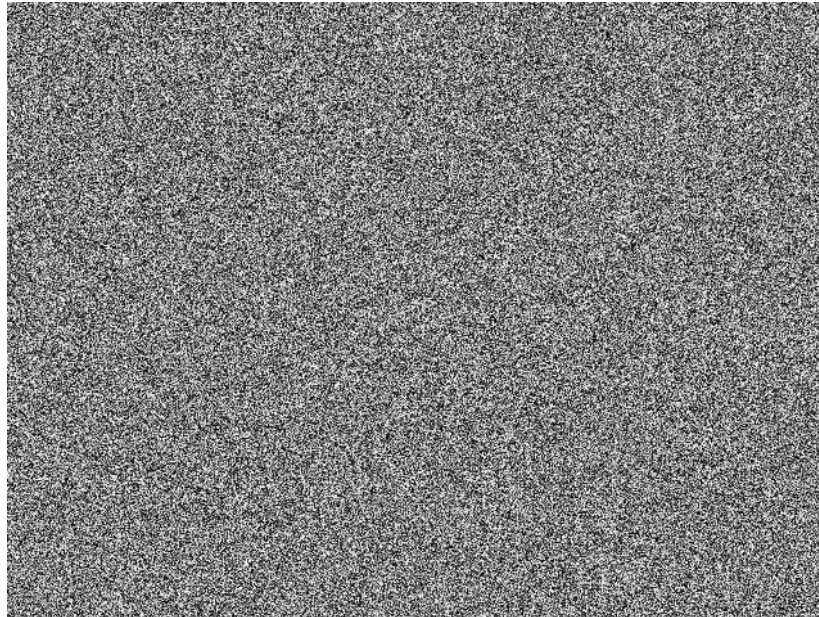


Figure 3.8: A white noise picture generated with Robson's white noise image generator.[48]
Settings: 640 squares horizontally, 480 squares vertically, size of squares 1, colours greyscale, bias none.

Perlin Noise



Value Noise

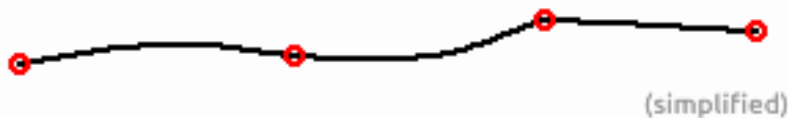


Figure 3.9: A comparison between the kinds of curves produced by Value noise interpolation and Perlin (and other Gradient) noise interpolation.[10]

Two particularly well-known Gradient noise algorithms that are commonly used for procedurally generating levels are the already mentioned Perlin Noise and Simplex Noise, both designed by American Computer Science professor Kenneth H. Perlin, with the former being an improvement on the former. Perlin Noise also takes a lattice of randomly assigned gradients, but the algorithm interpolates the dot products of those points instead of just their neighbouring values.[34] Simplex noise, meanwhile, tries to reduce the grid artifacts caused by the original algorithm, and has the added benefit of scaling better to larger dimensions.[13] Perlin filed a patent on his work in 2002 that was granted in 2005[43], which prompted the creation of the OpenSimplex noise algorithm[28][44][27] for free use; the patent has since expired in 2022, allowing free use to both Perlin and the original Simplex noise.[43]

Godot 3 previously featured an `OpenSimplexNoise` class[26][17] for generating noise textures, which used the OpenSimplex algorithm. In addition to using a “simplectic honeycomb” for its lattices[27], this algorithm also (to quote Michael Powell) “expands the range of the gradients a bit, so they can extend a little bit into neighboring cells. This theoretically makes the noise a little bit smoother, but it also means that extra cells need to be checked.”[44] Godot 4, on the other hand, allows us to use the *original* Simplex noise algorithm, as well as Perlin noise, 2 types of Value noise and a variation of Simplex noise that produces smoother, high quality noise images with an additional performance cost, and it allows us to control which algorithm we use for noise generation using the “noise_type” property and “NoiseType” enumeration in

the “FastNoiseLite” class that is now used for noise.[34]

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[16] (the implementation I made for this project does **not** guarantee maximality, however).

An implementation of Poisson disk sampling was originally developed in 1991 by Don P. Mitchell[37] as a replacement for inefficient Monte Carlo “dart-throwing” algorithms.[46] Mitchell’s algorithm ran in $\mathcal{O}(n^2)$ time, whereas Robert Brinson’s 2007 improved algorithm for Poisson disk sampling[3] ran in $\mathcal{O}(n)$. Subsequent quality and speed improvements to Brinson’s algorithm were published in 2019[47], 2021[46] and 2022[49]. The implementation made for this project, as well as the Unity project I based it on, were both based on Brinson’s 2007 $\mathcal{O}(n)$ algorithm.[32][31]

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

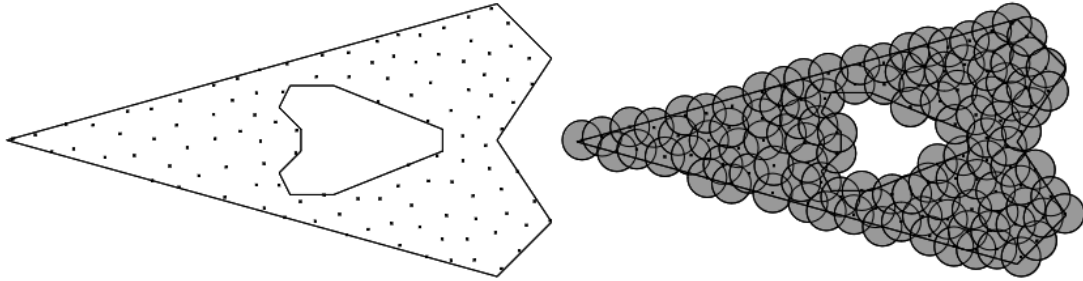


Figure 3.10: A diagram of a maximal Poisson disk distribution done on a concave plane, with the right side denoting maximality through the grey disks overlapping but not any points overlapping.[16]

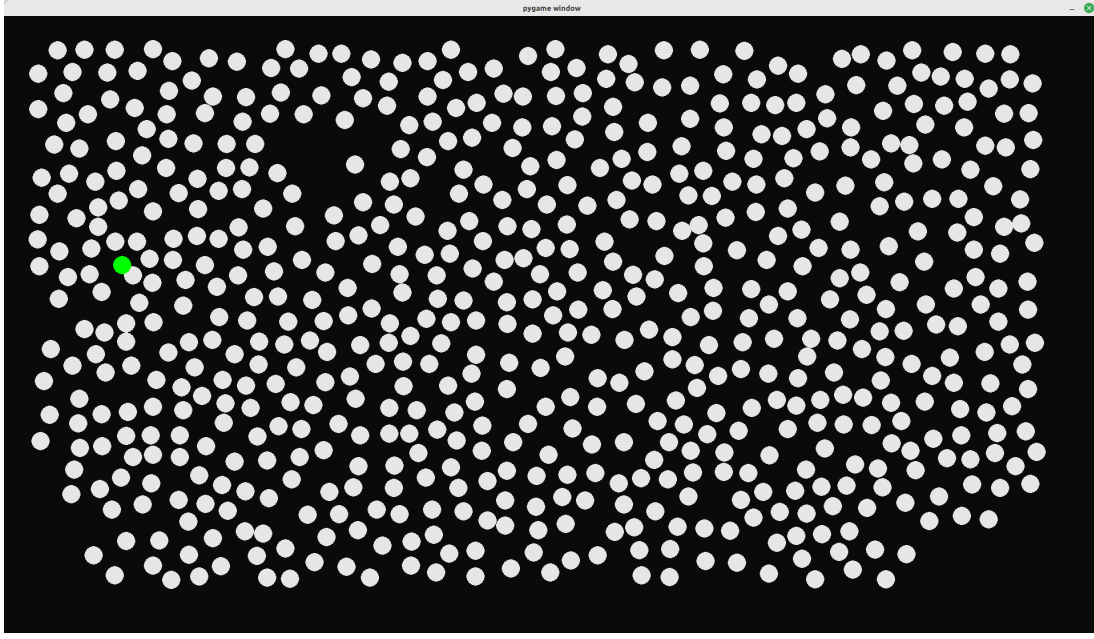


Figure 3.11: An implementation of Poisson disk sampling made in Pygame.[1] The screenshot was taken *after* all of the samples were taken.

3.1.4 Voronoï Cells

Named after the Ukrainian mathematician Georgy Voronoy, Voronoï cells work by taking a map of points, and randomly selecting a group of points. Within that selected group, cells are formed by calculating, in each point of the grid, the closest of the selected points to it. That is, each cell represents the group of points that are the closest to that random point (including that point in the group as well).[15] The final arrangement of cells represents a Voronoï Diagram or Voronoï 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, with straighter diagonals and cells shaped like irregular polygons, the geometry of which is more “blocky” and resembles taxicabs (hence its alternate name “Taxicab Geometry”). Two visual comparisons of the kinds of Voronoï cells generated with either distance calculation

are shown in Figures 3.12 and 3.13.

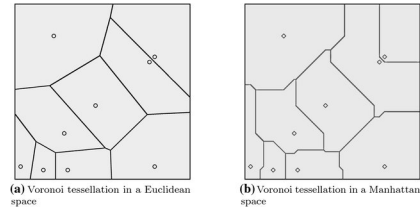


Figure 3.12: A visual comparison of the kinds of Voronoi cells generated with the Euclidean and Manhattan distance.[52]

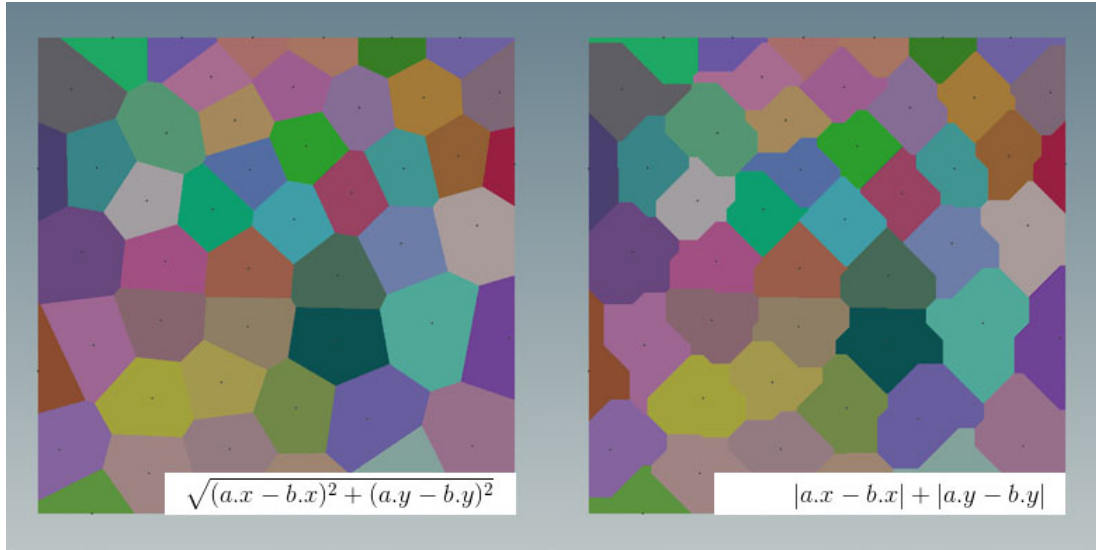


Figure 3.13: Another example of the differences between a Voronoi tessellation with distance between points calculated with either the Euclidean distance or the Manhattan distance.[53]

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 Commonalities Between Implementations

To implement the same scenario, aforementioned in the background of this report, across all 4 algorithm implementations, I had to include some of the same code and functions, as well as the same tile set shown in Figure 3.14.

From this tileset, which contains 1078 tiles, my code uses 27 building tiles, 13 tiles for trees and other fauna, 1 tile for the player character and one of 4 tiles for the ring. The relevant coordinates of the tiles for buildings, trees and the ring are each stored in constant arrays in

the script, while the player tile's coordinates are just stored in a local constant (not an array, since there is no need for one).

To handle player placement and subsequent movement, I have several functions. Godot's built in "physics_process" function handles events that happen in real-time, and is commonly used, like in this context, for player movement. In it, I first store the current player's cell, "player_movement_cell", in "previous_cell", then I initialise a "direction" based on which input movement was pressed ("Vector2i.LEFT" when "ui_left" was pressed, and so on). Then I add the player's current cell with the direction to calculate the potential "new_movement_cell". If this cell is within the bounds of the environment, as well as either a tree or empty space (or the ring), it moves there, and the previous cell gets erased. If the player ends up moving into the cell where the ring is, the player wins the game, and all movement is paused while a winner's dialog popup shows up. The player moves **very** quickly in our games, and I have yet to figure out how to slow down this movement while also not making movement so slow that the games drags; the player will not want to have to continually press down an arrow key to move to 1 cell in a map of 2880 cells. Since the performance of the algorithms are more important in this project, however, I decided to leave the very fast player movement as is.

I have written "place_player" and "place_ring" functions that handle the random generation of the player's and ring's initial starting positions. Both use the "_get_random_placement_cell" helper function to retrieve a new cell, and both use a while loop to make sure the randomly generated cell isn't already occupied. In both functions the placement cells are assigned and calculated **before** the while loop, so that their placements do not default to just (0, 0) in the beginning.

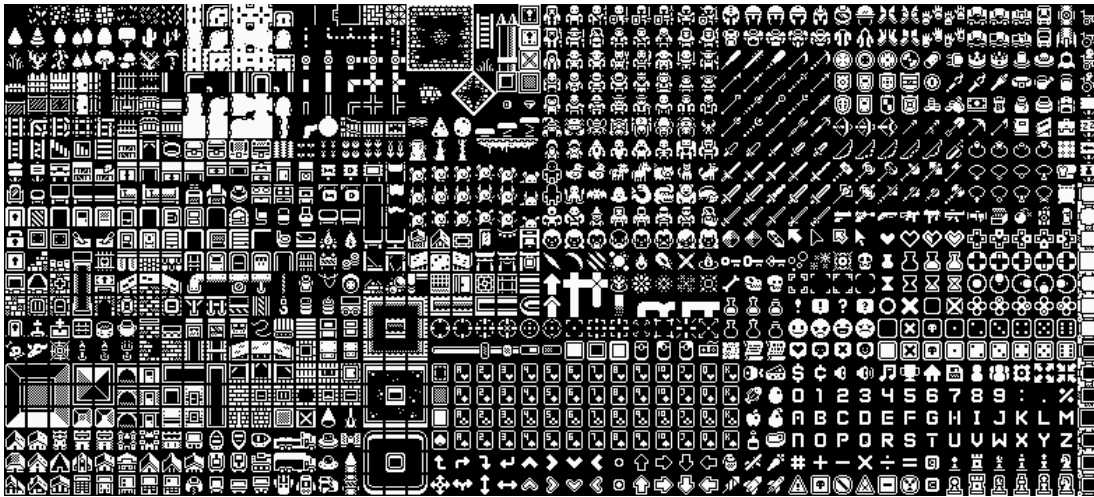


Figure 3.14: The tileset used for all 4 implementations of my scenario with PCG algorithms.[29] Of all the 1078 tiles, of size 16x16, in this tileset, only 45 of them get referenced in my code.

Across all implementations, there are two local variables, “x_tile_range” and “y_tile_range”. Both of these calculate the dimensions of our tile map by taking the display window’s respective x and y dimensions from the project’s settings (1152x640) and divides them by the respective x and y dimensions of the cell size (16x16). “x_tile_range” should resolve to 72 upon runtime, and “y_tile_range” should equal 40, giving us our 72x40 tile map that gives us a total of 2880 cells to work with in our games.

Finally, there are two dialog popups added to each scene tree, one for describing the game’s story (“AcceptDialog”, of type “AcceptDialog”) and another for when the game ends after the player has collected the ring (“WinDialog”, of type “ConfirmationDialog”). For “AcceptDialog” the “confirmed” and “canceled” signals are both connected to the function “_on_AcceptDialog_closed”, which hides the popup and unpauses the game. For “WinDialog”, on the other hand, “confirmed” is connected to “_on_WinDialog_confirmed” and “canceled” is connected to “_on_WinDialog_canceled”. “_on_WinDialog_confirmed” is meant to generate a new level layout, while “_on_WinDialog_canceled” is meant to close the game, both when the cancel button (labelled “Get Me Out of Here”) is clicked and when the cross on the top-right corner of the popup is clicked. However, as of now, only the top-right corner of both popups does what it is supposed to; clicking any of the other buttons from both popups, for some reason, does nothing at the moment, and I *did* make sure, in my code, that the signals were properly connected. However, the games themselves still run as they are supposed to, and the integration of the algorithms into the levels in the games are more important here, so since *they* still work, I decided to leave the popups, their behaviour and their code as they were. *If*

they are engine issues, regarding the buttons, they may hopefully get fixed in future versions of Godot.

3.2.2 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 Alexander Gillberg.[22] In the code from the Godot 3 project Gillberg made in that video[23][22], 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.[24] 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[40]:

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

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 19 milliseconds on average. This was the default grammar used by the L-System in the game. I also included 3 more grammars, one that generated more buildings (and impossible level layouts), another that generated more trees and another that generated more empty space. These can be easily selected with the “ruleset” export variable in the Godot editor. Further variance can be added with the addition of a randomly generated axiom, capped at a maximum height or smaller (minimum 1). If said option is enabled in the Godot editor, the default value in the export variable for setting this cap is 10, and since it is an export variable, it too can be adjusted in the editor as the developer sees fit.

3.2.3 Perlin/Simplex Noise

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

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

For the generation of the noise itself, I *could’ve* added a “Sprite2D” node to the scene tree, the root of which was my “TileMap”, and gave it a “NoiseTexture2D” texture and set its “noise” property to a newly-created “FastNoiseLite” instance, the latter of which contains the actual noise data. In the early stages of this implementation’s development, that’s what I did, and I created a script that solely set the seed of the “FastNoiseLite” resource to a random integer (using the “randi” method). However, for a more authentic result, and to forgo the

need of an additional node and noise texture that will not even be visible in the final product, I decided to create the noise for this algorithm implementation entirely programmatically. I stored the “FastNoiseLite” instance in its own class variable “noise”, and instantiated it with the “set_noise” method when starting the game (the “_ready” function automatically runs when the game starts).

Initially having done the noise integration with a sprite node and noise texture allowed me to experiment with some of the “FastNoiseLite” class’s properties before finally resorting to programmatic noise creation. An instance of this class, by default, uses the “Simplex Smooth” noise algorithm, a version of the Simplex algorithm that produces higher quality noise images at the expense of slower speed.[34] We can also use just “Simplex” noise for higher speed, as well as the original “Perlin” noise algorithm.[34] Godot also allows us to use two kinds of Value noise, as well as a “Cellular” type that combines algorithms like Worley Noise and Voronoï diagrams to create “regions of the same value.”[34] I had problems with the “Cellular” noise type when experimenting with it, for reasons I will get into later, but the other noise types I made readily accessible in an “export” variable in my script (that is, a variable that can be easily accessed in the Godot editor when the TileMap node is clicked on) when I removed the sprite node and decided to programmatically make the noise. When the “set_noise” function is called, the noise type is assigned through the “_get_noise_type” function, which returns an integer value depending on the type of noise selected, and the returned result is cast to “FastNoiseLite”’s “NoiseType” enumeration[34] before it gets assigned (this prevents an “INT_AS_ENUM_WITHOUT_CAST” warning from the Godot editor’s linter for GDScript[36]).

Furthermore, I have 3 other export variables in the TileMap script for this implementation that directly correlate to some of “FastNoiseLite”’s properties. The “noise_frequency” variable in the script correlates to the “frequency” property in “FastNoiseLite”, which, as both names suggest, sets the noise frequency; the higher the frequency, the rougher and more granular the noise[34], which is probably why it is set to 0.01 by default.[34] The “fractal_type” and “cellular_distance_type” in the script **directly** correspond to the “fractal_type” and “cellular_distance_function” properties respectively, to the point where both even use the relevant enumerations from “FastNoiseLite” directly (“FractalType” and “CellularDistanceFunction” respectively).[34] The relevant values are all assigned accordingly in “set_noise”.

In terms of determining whether or not to place buildings or trees (or nothing), I took inspiration from a YouTube tutorial by Gingerageous Games utilising Godot 3[20][21] (which

breaks in Godot 4). His tutorial used multiple “TileMap” nodes in a single scene tree with a “Node2D” root, and controlled each individual tile map, representing a specific part of the environment (such as grass and roads), and used a floating point “cap” to determine whether or not to place a tile in a cell based on the noise pixel retrieved at that cell’s coordinate.[20][21] Since I’m using just one tile map for everything (trees and buildings), I had to mitigate a conflict where the building cap was smaller than the tree cap. If that were the case then, since the tree cells get painted first in my implementation, no buildings would ever get painted. To mitigate this, I added an additional condition to my if-statement for painting building cells (in the same line, to prevent creating a nested if-statement), which would allow the algorithm to overwrite an already painted tree cell with a building cell subject to a randomly generated floating point number (between 0 and 1 inclusive) being below a pre-defined floating point number in the exported variable “building_overtakes_tree”. This would then allow there to be a controlled proportion of buildings compared to trees (the higher the proportion, the more buildings compared to trees), regardless of whether the building cap was lower than the tree cap or not, and the algorithm would still perform as normal should the reverse be the case.

3.2.4 Poisson Disk Sampling

The Poisson Disk Sampling implementation was based on a Unity tutorial by Sebastian Lague[31][32], in which he used his algorithm to draw points onto a grid. He based his algorithm on Bridson’s $\mathcal{O}(n)$ algorithm.[3] The way he wrote *his* implementation was such that the radius of the circle would be equal to the diagonal of each square in the grid by default (when the radius was 1.0), ensuring that no point ever lies within the radius of another.

My implementation of the Poisson Disk Sampling algorithm mostly took from him, with some changes. Lague did his implementation in the C# language and, while Godot 4 *does* have a separate version with C# and .NET support, I opted to use the standard GDScript distribution of Godot 4 with all of my implementations. This meant that I had to adapt the code to work with not just the tile map but also the way GDScript worked. For one thing, the “grid” array in the “generate_points” had to be manually initialised by inserting arrays into an empty array, the quantity determined by what would have been the outer length of the 2D array (and what basically *was* this in Lague’s implementation), that being the ceiling function of the x-dimension of the sample region size divided by the cell size. From there, in each of the nested arrays, the value 0 had to be programatically inserted to all of them, the quantity of the *zeroes* also being determined by what would have been the *inner* length of the 2D array

(and what basically *was* this in Lague’s implementation), that being the ceiling function of the y-dimension of the sample region size divided by the cell size.

Adapting Lague’s implementation from C# and Unity to GDScript and Godot involved some extensive research into Unity’s API. When calculating the angle in “GeneratePoints”/“generate_points”, for example, the equivalent of Unity’s “Random.value” in Godot is “randf” (which *has* no static class to be called from). Furthermore, GDScript has a “TAU” constant that does the “Mathf.PI * 2” calculation done in Lague’s Unity implementation. The “sqrMagnitude” method used in Lague’s “isValid” function becomes “length_squared” in my “is_valid” method. When implementing “isValid” in GDScript I also had to make sure the inner and outer dimensions of the grid could be adequately accessed. I go over how I did that in the “Implementation” section of this report (see chapter 7).

3.2.5 Voronoï Cells

I based my implementation of this algorithm on some JavaScript code posted by an anonymous contributor to the Procedural Content Generation Wiki on the Wikidot platform in 2017, in which a brute-force implementation of the algorithm was implemented.[15] An auxiliary function in the JavaScript code, “randRange”, was taken out of *this* implementation, since Godot has a built-in “randi_range” function that serves the exact same purpose.[35] As I got further and further with my implementation of Voronoï diagrams in Godot, I realised the way the algorithm inherently worked meant that the level layouts it designed would be wholly unique, especially compared to the other three algorithms for which I made implementations of my scenario.

For example, unlike the other implementations, the algorithm ensure that all cells of the tilemap were **always** covered (to start with, in our game’s context), whereas the other implementations always left some cells unpainted. The nature of Voronoï tessellations also meant that groups of trees and buildings were bunched together, with no guarantee that they would ever form coherent connections that would make sense in a level of our scenario. This meant that the ring and player placements had to be altered so that, instead of being placed in non-existent empty cells, they would replace the cell of a tree.

Even *with* that, there would be no guarantee that a player would be able to complete a level successfully. For example, if a player and ring were spawned in different Voronoï cells of trees, and both of those cells were separated by cells of buildings such that they could not ever be feasibly reached, the game would be impossible to finish. Therefore, a new input event was created, “reset_position”, which can be triggered by pressing *either* the G key on a standard

computer keyboard *or* the right-click mouse button. Triggering the event respawns the player character in a different position, which could be occupied by either a tree or the ring, ensuring that the ring can still be collected and, therefore, game can still be won. The code for when this event is triggered is essentially a rehash of the code for “place_player”, except that the new cell **can** be the cell occupying the ring, and also the previous cell’s contents will be deleted (as the player is no longer at that position).

While the differences are drastic and very noticeable, I nonetheless kept working on this implementation and included it in my project. I believe that the fact that I was able to work through it and implement a working version of my scenario with it (albeit with some changes) adds further strength to my claims that Godot can work well with procedural generation algorithms, even ones where use in the context of a tile map RPG would be rarer, as well as proving my strengths as a games programmer in making tile maps work with PCG algorithms.

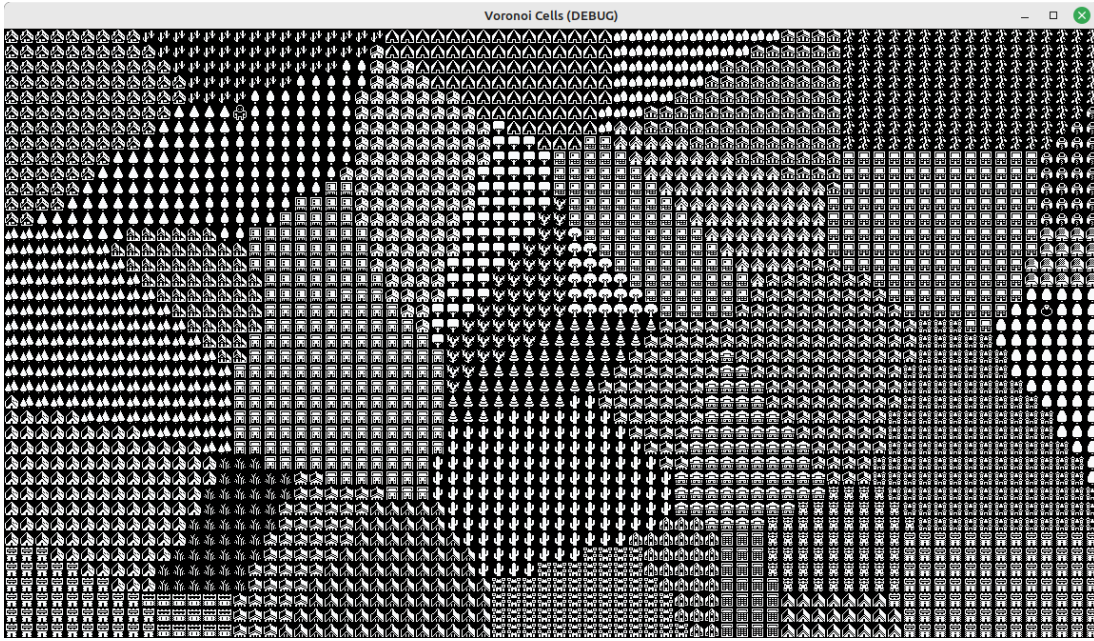


Figure 3.15: An example of the Voronoi cells implementation of the scenario in action.

Chapter 4

Design & Specification

Here, I will provide an abstract level of how I compared the performance of each content generation algorithm and how I made sure each implementation could produce as similar/like-for-like results as possible (and where they *couldn't* do so). I delve into more specifics and quantified data in Chapter 7.

4.1 Performance

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

While my Noise implementation was slower than my L-System implementation by a magnitude,

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

Voronoi Cells took the longest to compute on average. Computations with the Euclidean distance measurement took longer than those measured with the Manhattan distance.

4.2 Layouts

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

While the noise implementations varied greatly depending on what settings were used, and the way the implementation was designed allowed for very many possibilities as to how the noise would turn out (and how it would affect the final level), the results that I found produced the most similar results to that of the Poisson Disk Sampling implementation had the following configurations:

- Noise Type (“noise_type”): Simplex Smooth
- Fractal Type (“fractal_type”): Fractal None
- Cellular Distance Type/Function (“cellular_distance_type”): Distance Euclidean
- Noise Frequency (“noise_frequency”): 0.894
- Tree Cap (“tree_cap”): -0.048
- Building Cap (“building_cap”): -0.252
- Building Overtakes Tree (“building_overtakes_tree”): 0.12

Chapter 5

Implementation

Here I will go a degree deeper as to how I made each algorithm work. Where possible, I plan to use code snippets from the work I have done to justify how and why things were implemented the way they were.

5.1 Lindenmayer System

To implement our basic grammar in Godot (see chapter 3.1.1), we can take each rule and replace each string in accordance to our one rule, using the replace method, as demonstrated in Figure 5.1.

For handling more than one rule, we can instead use a new string buffer variable where, for each character in our string, we can attain a new string and append it to our string buffer. The resulting string is then returned and interpreted. This can be represented in Godot as demonstrated in Figure 5.2, which uses two functions to perform string replacement. The first function “get_new_replacement” performs the character replacement according to the L-System’s grammar rules, while the second function “replace_string” uses a string builder variable to allow for replacement of characters without directly affecting the original string and causing unwanted side effects (see chapter 3.2.2 and also the tutorial in the following citations[22][23][24], from which I took major inspiration for my L-System implementation). The “get_new_replacement” function was eventually used in my final implementation, whereas the code in the “replace_string” function was adapted into my implementation’s “parse” function, shown in Figure 5.5, in which some export variables were accounted for and the number of iterations was controlled through a while loop rather than a for loop, the while

condition being that the string size was smaller than the total number of cells. The string that “parse” returns chops off any excess characters.

This can *then* be used to handle more complex grammars that can handle more than one rule in which characters in strings are replaced by other strings of variable length, as seen in the example in Chapter 3.1.1.

With a constant use of the same grammar rules and axiom, one issue that arose in the development of levels in our scenario, with L-Systems, is the lack of variance in the kinds of tiles placed, and it was hard to figure out how to deviate even slightly from the eventually recognisable patterns the algorithm and default grammar (see chapter 3.2.2) would create. I thus developed a mitigation for this by allowing a choice between the provided axiom or a random one (the user/developer can also change the axiom in Godot’s editor). This is down to the exported variables in the L-System node’s script that I implemented (the L-System node’s parent is the “TileMap” root node). The relevant ones from the “l_system.gd” script file are shown in Figure 5.3. If “use_random_axiom” is set to true (and it is by default), then it takes a maximum character limit “upper_limit” (again an export variable configurable in the Godot editor itself) and then it generates a string of a length **up to** that limit- that is, the length of the returned string between 1 and the limit, both inclusive- using the alphabet that all of the provided grammars adhere to (“O” for blank space, “W” for trees and “B” for buildings). The axiom is then assigned to the “string” script variable in the “paint” method by calling the “paint” method and assigning the return type to it.

For the sake of creativity and additional experimentation, I created additional grammars that can be configured with the “ruleset” variable. The “Default” grammar is as described in 3.2.2, and the additional grammars generate higher proportions of either buildings, trees or empty space depending on which one is chosen (with the “more buildings” grammar producing levels that are impossible to finish in our scenario). The grammars themselves are detailed in Figure 5.4, and the method used to return the specifically set grammar in “parse” is shown in Figure 5.6.

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

Figure 5.1: A line of code that demonstrates directly replacing characters in a string according to our L-System grammar’s rules.

```

1  func get_new_replacement(character: String) -> String:
2      for rule in rules:
3          if rule["from"] == character:
4              return rule["to"]
5      return character
6
7  func replace_string(string: String) -> String:
8      var new_string = ""
9      for character in string:
10         new_string += get_new_replacement(character)
11     return new_string

```

Figure 5.2: Two GDScript functions for replacing characters in an L-System grammar with more than one rule. The first function was used in the final L-System implementation of the scenario. The second function was adapted into the “parse” function of my implementation. Both functions are in the `l_system.gd` script file.

```

1  @export var use_random_axiom: bool = true
2  ## Defines how many characters a random axiom can have MAXIMUM.
   Only used when use_random_axiom is true.
3  @export var upper_limit: int = 10

```

Figure 5.3: The `use_random_axiom` and `upper_limit` variables used when allowing a random axiom for an L-System grammar and then setting a maximum character limit for it.

$$O \rightarrow BWOB$$

$$W \rightarrow WBOBO$$

$$B \rightarrow BB$$

$$O \rightarrow WWO$$

$$W \rightarrow WBWO$$

$$B \rightarrow BWWO$$

$$O \rightarrow OOBWO$$

$$W \rightarrow OB$$

$$B \rightarrow OW$$

Figure 5.4: The other 3 grammars I created for my L-System implementation, aside from the default one covered in chapter 3.2.2. More buildings, more trees and more space respectively.

```

1  func _size() -> int:
2      return tile_map.x_tile_range * tile_map.y_tile_range
3
4  func rand_axiom() -> String:
5      var string_buffer: String = ""
6      var limit: int = randi_range(1, upper_limit)
7      for i in range(limit):
8          string_buffer += ["O", "W", "B"].pick_random()
9      return string_buffer
10
11 func parse() -> String:
12     if use_random_axiom:
13         axiom = rand_axiom()
14         string = axiom
15     if not use_custom_ruleset or ruleset != "Default":
16         rules = _get_ruleset()
17     var size: int = _size()
18     while len(string) <= size:
19         var new_string = ""
20         for character in string:
21             new_string += get_new_replacement(character)
22         string = new_string
23     string = string.substr(0, size)
24     return string

```

Figure 5.5: The parse function in `l_system.gd`, which takes the `rand_axiom` and `_get_ruleset` functions (if needed; the latter is shown in Figure 5.6), then gets the string size through `_size`. Then, the string replacements described in 5.2 are carried out in the while loop, before the string is then returned (albeit without any excess characters that will not be used in the paint method to paint the cells in the tile map).

```

1  func _get_ruleset() -> Array[Dictionary]:
2      match ruleset:
3          "More Buildings (IMPOSSIBLE)": return MORE_BUILDINGS
4          "More Trees": return MORE_TREES
5          "More Space": return MORE_SPACE
6          _: return DEFAULT

```

Figure 5.6: The `_get_ruleset` function used in the `parse` method in figure 5.5.

5.2 Perlin/Simplex Noise

While the “noise_type” export variable is a selection of strings that are eventually taken from to return an enumeration for “NoiseType”, a member of the “FastNoiseLite” class[34], the “fractal_type” and “cellular_distance_type” export variables more directly correspond to the relevant enumerations from “FastNoiseLite”, as described earlier in chapter 3.2.3 and shown in Figure 5.7. The *other* export variable is “noise_frequency”, which corresponds to the “frequency” property in “FastNoiseLite”. This is again shown in Figure 5.7.

Initiating the noise variable is done in the “set_noise” function, in which, after instantiating the “FastNoiseLite” class, the relevant export variables are then taken in and assigned their values accordingly, with the noise type handled through the “_get_noise_type” function, which returns an integer that is then cast to the “NoiseType” enumeration. By contrast, all the other attribute assignments- “frequency”, “fractal_type” and “cellular_distance_function”- were far more simple, since I could just use the values of “noise_frequency”, “fractal_type” (*in the script’s export variable, **not** the noise instance*) and “cellular_distance_type”, and assign them accordingly. Both functions are shown in Figure 5.8.

In both Figures 5.7 and 5.8, I had a commented out export variable, “octaves”. I wanted to see if changing the number of octaves in our noise image (or, rather, “layers”) would have an effect on the level layouts designed (the default is 5[34]). Unfortunately, in my experiments, I found that it did not have any noticeable effect, so I decided to leave the default number of octaves as is.

Painting the tiles in our tile map involves a “paint_points” method in which we iterate through every single cell of our tile map. For each cell in that map, we then check first if it can

paint a tree. If the noise value retrieved at the given pixel from the noise image is smaller than the maximum limit we set for trees, “tree_cap”, then we paint a tree tile at that cell. Then, we check for buildings. As I described in chapter 3.2.3, I had to implement additional Boolean checks for when “tree_cap” was smaller than “building_cap”, so that buildings could still get placed. First, we check that “building_cap” is smaller than *or equal to* “tree cap” **and**, if so, whether or not we overwrite the tree tile with a building tile in the same place, subject to a random float between 0 and 1 falling below the probability value set in “building_overtakes_tree”. The *alternative*, for when “building_cap” is smaller than “tree_cap”, is to check whether the current “noise_point” is smaller than the “building_cap” we set. In both cases, of course, that cell cannot already be occupied by something else. If those conditions are true, we can then paint a building tile. The code behind this part of the algorithm is shown in Figure 5.9.

```

1  var noise: FastNoiseLite
2  @export_enum("Perlin", "Simplex", "Simplex Smooth", "Value", "Value
    Cubic") var noise_type: String = "Simplex Smooth"
3  @export var fractal_type: FastNoiseLite.FractalType
4  @export var cellular_distance_type: FastNoiseLite.
    CellularDistanceFunction
5  #@export_range(1, 10, 1) var octaves: int = 5
6  @export_range(0.0, 1.0) var noise_frequency: float = 0.894

```

Figure 5.7: The noise, noise_type, fractal_type, cellular_distance_type and noise_frequency script variables in tile_map.gd. The latter three are export variables, and the latter two of *those* are assigned to the enumerations FractalType and CellularDistanceFunction respectively (both are members of the FastNoiseLite class).[34] noise_type, meanwhile, uses an external function called to assign properties to a newly created FastNoiseLite instance which is assigned to the noise script variable. noise_frequency here corresponds to the frequency attribute in FastNoiseLite. Notice the commented out octaves variable. Changing the number of octaves used had no effect on the level layouts produced, so I left the default number of octaves (5) as is.


```

1  func _get_noise_type() -> int:
2      match noise_type:
3          "Perlin": return 3
4          "Simplex": return 0
5          "Value": return 5
6          "Value Cubic": return 4
7          _: return 1 # Return Simplex Smooth by default
8
9  func set_noise() -> void:
10     noise = FastNoiseLite.new()
11     noise.frequency = noise_frequency
12     noise.noise_type = _get_noise_type() as FastNoiseLite.NoiseType
13     noise.fractal_type = fractal_type
14     noise.cellular_distance_function = cellular_distance_type
15     # noise.fractal_octaves = octaves
16     noise.seed = randi()

```

Figure 5.8: The `set_noise` method in the `tile_map.gd` script, which uses the earlier defined `_get_noise_type` method to assign the noise type. The noise type returned is then cast from an integer to an enumeration of type `NoiseType` (from `FastNoiseLite`). The `fractal_octaves` line is commented out because, when experimenting with octaves, I found no real effects they could have on the level layouts generated, so I left the default number of octaves (5) as is.

```

1  func paint_tiles() -> void:
2      for x in range(x_tile_range):
3          for y in range(y_tile_range):
4              var noise_point: float = noise.get_noise_2d(x * tile_set.
                  tile_size.x, y * tile_set.tile_size.y)
5              if noise_point < tree_cap and not get_used_cells(0).has(
                  Vector2i(x, y)):
6                  set_cell(0, Vector2i(x, y), 0, trees.pick_random())
7              if ((building_cap <= tree_cap and randf() <
                  building_overtakes_tree) or (building_cap > tree_cap
                  and noise_point < building_cap)) and not
                  get_used_cells(0).has(Vector2i(x, y)):
8                  set_cell(0, Vector2i(x, y), 0, buildings.pick_random())

```

Figure 5.9: The `paint_tiles` method in the `tile_map.gd` script iterates through the tile map and gets each noise pixel from the relevant part of the noise image. It first tries to paint a tree tile there, subject to the `noise_point` value being below the limit set for trees. Then, it decides whether or not to paint a building there. The conditions for painting building tiles are as described in chapter 3.2.3 and further elaborated on earlier in *this* chapter, 5.2.

5.3 Poisson Disk Sampling

To be able to access the inner and outer grid sizes in my implementation of this algorithm, since GDScript does not have a concept of different “Array”s and lists, I stored the lengths of the inner and outer grid in local variables in the “`generate_points`” function. Those local variables, “`grid_x_axis_size`” and “`grid_y_axis_size`” as shown in Figures 5.10 and 5.11, essentially store the same grid size values as in Lague’s implementation, right down to performing the same division in a ceiling function, to the inner grid and the outer grid respectively. Since these dimensions would also be needed for “`is_valid`”, instead of creating 2 more script variables, I instead took them in as 2 additional method parameters, as shown in Figures 5.12 and 5.13, and used them accordingly when calculating the maximum and minimum bounds for searching the nearest points of the cell, as shown in 5.14. Doing it this way ensured that the computation of this algorithm would stay efficient and not stall with an adequate (not too high) number of rejection samples.

```

1  var grid_x_axis_size: int = ceili(sample_region_size.x/cell_size)
2  var grid_y_axis_size: int = ceili(sample_region_size.y/cell_size)

```

Figure 5.10: The lines used to determine the inner and outer dimensions of the grid array.

```

1  for i in range(grid_x_axis_size):
2      grid.append([])
3      for j in range(grid_y_axis_size):
4          grid[i].append(0)

```

Figure 5.11: The nested for-loop that initialises the grid array.
First, each inner array is initialised and inserted, then a number of zeroes, determined by the grid's y-dimension, are inserted.

```

1  if is_valid(candidate, sample_region_size, cell_size, radius,
    points, grid, grid_x_axis_size, grid_y_axis_size):

```

Figure 5.12: The line that uses the grid's x and y dimensions as parameters. This calls the `is_valid` method using those additional parameters (see Figure 5.13).

```

1  func is_valid(candidate: Vector2, sample_region_size: Vector2,
    cell_size: float, radius: float, points: Array[Vector2], grid:
    Array[Array], grid_x_axis_size: int, grid_y_axis_size: int) ->
    bool

```

Figure 5.13: The function `is_valid`, which takes in 2 additional parameters denoting the x and y dimensions of the grid array used in `generate_points`.

```

1  var search_end_x: int = min(cell_x + 2, grid_x_axis_size - 1)
2  var search_end_y: int = min(cell_y + 2, grid_y_axis_size - 1)

```

Figure 5.14: The relevant lines of code in `is_valid` that reference the grid’s x and y dimensions, stored in additional variables as aforementioned.

5.4 Voronoï Cells

The original JavaScript implementation, as mentioned before, had a “randRange” function that I took out, but there was also an additional “mapSize” parameter in “definePoints” that, in *my* “define_points” function, didn’t really need, since I made sure the map’s dimensions were readily accessible via the “x_tile_range” and “y_tile_range” script variable. I therefore took out the second parameter in “define_points”, as shown in Figure 5.15, and substituted it with “x_tile_range” and “y_tile_range” accordingly, as shown in Figure 5.17.

The type of each Voronoï cell was determined by taking, and then deleting, a value from the “types” array. Said array is local to that function, and it is initialised by duplicating the “trees” array, then appending it with the “buildings” array, making sure the same type cannot be used for a Voronoï cell twice. Duplicating the array before merging it essentially makes sure that the *original* “trees” array is not affected by deletions performed on the “types” array. This computation is shown in Figure 5.16, and the deletion operation is shown in Figure 5.18.

Another addition to my implementation of the algorithm was the choice of using either the Euclidean distance or Manhattan distance for calculating the distance between points that would form cells. This was done with a function “calculate_points_delta”, as shown in Figure 5.23 and used in Figure 5.22, which is called on the calculation of “delta” in “define_points”. The function takes the contents of the exported variable “distance”, as well as the current “x” and “y” coordinates and point ID “p” during the current points delta calculation. It then checks if the String value in “distance” denotes either the Euclidean distance or the Manhattan distance, then it finally returns the appropriate calculation. Using the Manhattan distance instead of the Euclidean distance does indeed yield a considerable performance increase (as well as creating fewer Voronoï cells by using a smaller “random_starting_points” value), which I touch on in the Evaluation chapter.

Furthermore, my implementation would often paint cells in the tile map that were out of

bounds, so to mitigate this when painting them, I created an additional function “_is_in_bounds”, as shown in Figure 5.21 and used in Figure 5.20, for checking whether a painted cell (that is, the coordinates of the current point **plus** the delta/difference between it and the closest of the randomly selected starting points to it) is within the boundaries of the tile map. If it is not, then it does not get painted, though it is not deleted from the point’s citizens array either.

```
1 func define_points(num_points: int) -> void:
```

Figure 5.15: The define_points function header, with no argument for the map’s size. The num_points value that gets taken in during runtime is determined by the script’s export variable random_starting_points.

```
1 var types: Array[Vector2i] = trees.duplicate()
2 types.append_array(buildings)
```

Figure 5.16: The types array being initialised in define_points, with its values taken from the trees and buildings arrays, such that no type can be used for a cell twice, while also making sure that the original trees and buildings arrays are not affected by the deletions on types.

```
1 var x: int = randi_range(0, x_tile_range)
2 var y: int = randi_range(0, y_tile_range)
```

Figure 5.17: Godot’s built-in randi_range function being used in place of a self-defined one in define_points.

```
1 var type: Vector2i = types.pick_random()
2 types.erase(type)
```

Figure 5.18: The types of each Voronoï cell being picked and the erased in define_points.

```

1  const EUCLIDEAN: String = "Euclidean distance"
2  const MANHATTAN: String = "Manhattan distance"
3  @export_enum(EUCLIDEAN, MANHATTAN) var distance: String = MANHATTAN

```

Figure 5.19: The applicable values of the exported variable distance.

```

1  if _is_in_bounds(point["x"], citizen["dx"], point["y"], citizen["dy
    "]):
2      set_cell(0, Vector2(point["x"] + citizen["dx"], point["y"] +
        citizen["dy"]), 0, point["type"])

```

Figure 5.20: The appropriate block of code in the paint_points function that checks to see if a point would be in bounds or out of bounds before painting it in its relevant tile map cell. It does **not** delete the point if it lies out of bounds.

```

1  func _is_in_bounds(x: int, dx: int, y: int, dy: int) -> bool:
2      return x + dx >= 0 and x + dx < x_tile_range and y + dy >= 0 and
        y + dy < y_tile_range

```

Figure 5.21: The _is_in_bounds function called in the code snippet in Figure 5.20.

```
1  var delta: float = calculate_points_delta(x, y, p)
```

Figure 5.23: The calling of `calculate_points_delta` from Figure 5.22 in `define_points`, using the current `x` and `y` coordinates and point ID `p` in the iteration when grouping tile map cells together to form Voronoï cells from the randomly selected starting points.

```
1  func _squared(x: int) -> int:
2      return x ** 2
3
4  func calculate_points_delta(x: int, y: int, p: int) -> float:
5      if distance == EUCLIDEAN:
6          return sqrt(_squared(points[p]["x"] - x) + _squared(points[p]
7              ["y"] - y))
8      return abs(points[p]["x"] - x) + abs(points[p]["y"] - y)
```

Figure 5.22: The `calculate_points_delta` function being called in Figure 5.23. `_squared` is a self-defined helper function that is only used for the Euclidean distance calculation; it does as it says (it squares the number taken into it and returns the result).

Chapter 6

Legal, Social, Ethical and Professional Issues

Throughout the course of this project, I made sure I abode by the principles set out in the Code of Conduct & Code of Good Practice issued by the British Computer Society[51], acting with integrity, honesty and transparency in the way I handled potential licensing issues with my work and the other work I used as both reference and inspiration. Throughout my report and project video, I have thoroughly discussed and elaborated on the way external code, articles and other references were used in both my writing and my software artefacts. I have also denoted references and inspirations for code in my artefacts via the inclusion of comments in script files. To further ensure full transparency in my research, all of my citations, even remotely tangential ones, are cited in my bibliography as appropriate. In this chapter, the ways in which I dealt with the licenses of code references in my implementations are detailed in the following section 6.1, while the details on how I plan to release my *own* code and report, both for public access, are detailed in section 6.2.

6.1 Using Other People's Resources

As I continually worked on my project, I made sure the resources I worked with were freely available to use in an academic context like this.

For example, the Unity tutorial I used as an inspiration of my Godot Poisson Disk Sampling implementation[31] has its project files under the MIT License[32], a permissive open-source license which means it can be freely used and adapted with, even commercially.[41] This meant

that I could base my Godot implementation on his Unity implementation without fear of any legal implications. Nonetheless, to act with integrity, I have denoted properly, in this report and in code comments, that I have taken from and adapted his work, citing it accordingly in my bibliography as well.

As aforementioned in chapter 3.2.5, the JavaScript code example I used from the Procedural Content Generation wiki[15], for my Voronoi Cells implementation, was submitted by an anonymous Wikidot contributor in 2017. Like most if not all of the Wiki's contents, it is licensed under the Creative Commons Attribution-ShareAlike 3.0 License (all contents of the wiki follow this license unless otherwise specified); that is, the article and its contents (including the JavaScript code example) can be freely used and adapted, subject to the condition that the original source is attributed **and** that any transformed work, *like my implementation*, **must** be published under the same or a compatible license.[6] Since there are no listed compatible source code licenses I can use in lieu of this license[8], I must therefore abide by the license contents of the original article in my source code, since my implementation and the original JavaScript code are similar to a noticeable, although not entirely like for like, degree. I go over how I will release all my project artefacts to the public, as well as the \LaTeX source code and \BIBTeX citations of this report, in the following section 6.2.

Projects I have used as references on a smaller scale were also accounted for. While the Godot 3 TileMap noise tutorial I referenced in chapter 3.2.3 is up on GitHub, it has no readily attached license to it.[21] However, I did not take or adapt any substantial code from it and the scripting APIs for Godot 3 and Godot 4 are vastly different, *especially* in the context of tile maps, so I fully believe that my implemented code will not pose an issue, and I plan to post this, and all my self-produced artefacts, on GitHub under conditions that I will explain in the following section 6.2.

To produce the screenshot in Figure 3.6, I forked an existing Godot 3 project on GitHub[23], taken from a YouTube video tutorial on how to use an L-System to draw line graphics in Godot[22], converted it to Godot 4 and added an additional set of rules to it based on the example lattice grammar featured in chapter 3.1.1.[24] While I did the conversion and added my own code contributions to the fork, I do not regard this as a substantial part of my project, and thus have not included it in my source code listings in chapter C. The person behind the code has previously denoted appreciation of other people's forks of his code, so I do not believe the lack of readily available code license in his original repository is a substantial issue here. Nonetheless, since small parts of his code have been adapted to work with my L-System

implementation, I have emailed him directly for his permission to do both that and add the permissive MIT license to my fork, and I got his permission from a private email conversation we had on Tuesday 18th April 2023:

“Well, haven’t done anything with the channel in years, everything is more or less up for grabs. So you have my permission to use anything and/or everything however you like. You can add the MIT license to your fork.

You can refer to this conversation if needed!”

I have let him know that I am citing his work in both my dissertation and the released artefact. I have already added the license to my fork.[24]

Any usage of external screenshots in Figures throughout this document are properly cited and linked to in my bibliography. Screenshots that were produced by me are clearly denoted as such in Figure captions. Any usage of code snippets in Figures were written by me, and any external references used as bases for these snippets were clearly and properly cited as such.

Some of the self-produced screenshots, specifically the ones in Figures 3.1, 3.2, 3.3 and 3.4, are taken from a Godot project I created when I was still learning about how L-Systems worked for this dissertation. I decided to include it in my source code listings in chapter C, primarily because much of the code in there (specifically in the script file DemoNode.gd) is used in the final project, but also because the commit history shows the process I initially went through in building an L-System that could handle multiple grammar rules, as detailed in chapters 3.1.1, 3.2.2 and 5.1. Do note, however, that it is not as important to the project’s motives as my main 4 algorithm implementations.

6.2 How I Will Release My Own Artefacts

I have planned for my source code to both the dissertation and artefacts to be released on GitHub for public access. In order to do so, I have to assign all my repositories licenses so that my usage intentions and any repository usage conditions are clearly defined.

I have chosen to use the Creative Commons Attribution-Sharealike 4.0 license, as described in the previous section, for both my report and all my artefacts. The aforementioned license allows for commercial and non-commercial usage on the condition that (1) the concerned product is properly attributed to when used and (2) any adaptations of and modifications to this work are released under the same license (or a compatible one, or a later revision of it).[7] The concerned product can be used *verbatim* (i.e. as is) without having to share their work under this license, but when it is adapted upon, *then* the share-alike conditions apply.

Although it is not widely considered good practice to apply Creative Commons licenses to code[9], I still feel that CC-BY-SA-4.0 is the best license for my project overall. As well as resolving any complications with my Voronoi cells implementation, again as discussed in the previous section (CC-BY-SA-4.0 is compatible with the CC-BY-SA-3.0 license used in the original JavaScript code, as CC-BY-SA-3.0 allows for licensing newly adapted works under later versions of the license[8]), it also ensures that my original Godot implementations are still available for public viewing and modification while also ensuring others can still modify it and everyone else who *isn't* modifying it has the freedom to view these modifications. In that regard, it *is* similar to copyleft licenses such as the GPL and LGPL, though, as of the time of this publication, only the former is listed as compatible with CC-BY-SA-4.0.[8]

In an academic context, it ensures that the work I have put in behind this report, as well as the implementations, are still available when the code is taken and then built upon by someone else (and that these modifications are also made available for others to see how my work was built upon). Applying the license to this dissertation, as well as my artefacts, allows me to ensure that all aspects of my work are viewable by everyone, and any improvements made to this work, by me or anyone else, are also viewable and publicly recognised.

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

Due to the nature of the project (being several implementations of a computer game), the testing behind this project has solely revolved around trial-and-error, messing around with the exported variables in the Godot editor to see how things worked and what configurations worked best for our scenario. This involved taking many screenshots of generated levels and examining things by eye, seeing how layouts compared across implementations.

7.2 Comparing the Different Algorithms and Drawing Conclusions on Which Ones Are Best

Chapter 8

Conclusion and Future Work

To conclude, I have gained a wealth of knowledge about the way some of the most popular procedural content generation algorithms work, and how they are typically integrated into working games. I also learnt how I could leverage the features of the Godot game engine for some of them; for example, the “FastNoiseLite” class allows me to generate noise textures in Value, Perlin and even Simplex noise and then modify them accordingly with additional frequency settings, fractal types and cellular distance functions. By implementing them in my own 2D tiled RPG scenario, I was able to get 4 procedural generation algorithms well-integrated into working games, proving Godot’s technical proficiency in making these kinds of games work, and proving my own abilities as a games programmer. I was also able to compare the implementations of my algorithms in such a way that the differences, in terms of both performance times and the kinds of levels they produced, could very easily be discerned. The motives of this project can be pushed still further by measuring and comparing the performances of these algorithms in Big-O notation, including even more ontogenic algorithms such as Worley Noise, the Diamond-Square algorithm, Markov Chains and Cellular Automata, as well as teleological algorithms such as the Rain Drop algorithm and Reaction-Diffusion systems, using a larger tile map on all of these algorithms and even using a different, more intensive scenario entirely, such as a 3D walking simulator/open-world game. With procedural generation for level design, the possibilities are practically endless.

References

- [1] Joseph Bakulikira. Poisson disc sampling in pygame. <https://github.com/Josephbakulikira/Poisson-Disc-Sampling-with-pygame>. [Online; accessed 15-April-2023].
- [2] Paul Bourke. L-system user notes. <http://paulbourke.net/fractals/lsys>, July 1991.
- [3] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, page 22–es, New York, NY, USA, 2007. Association for Computing Machinery.
- [4] Mark Brown. How (and why) spelunky makes its own levels. <https://youtu.be/Uqk5Zf0tw3o>, April 2016. [Online-accessed 10-April-2023].
- [5] Chris Bradfield and contributors. Exploring procedural generation algorithms in godot. https://github.com/kidscancode/godot3_procgen_demos.
- [6] Creative Commons. Attribution-sharealike 3.0 unported (cc by-sa 3.0). <https://creativecommons.org/licenses/by-sa/3.0/>. [Online, Accessed 18-April-2023].
- [7] Creative Commons. Attribution-sharealike 4.0 international (cc by-sa 4.0). <https://creativecommons.org/licenses/by-sa/4.0/>. [Online, Accessed 18-April-2023].
- [8] Creative Commons. Compatible licenses. <https://creativecommons.org/share-your-work/licensing-considerations/compatible-licenses>. [Online, Accessed 18-April-2023].
- [9] Creative Commons. Frequently asked questions — can i apply a creative commons license to software? <https://creativecommons.org/faq/#can-i-apply-a-creative-commons-license-to-software>. [Online, Accessed 18-April-2023].

- [10] Timm (Stack Exchange Contributor). “benefit of perlin noise over value noise” - stack exchange answer. <https://computergraphics.stackexchange.com/a/3609>, June 2016. [Online, Accessed 16-April-2023].
- [11] Various contributors. Ontogenetic - procedural content generation wiki. <http://pcg.wikidot.com/pcg-algorithm:ontogenetic>.
- [12] Various contributors. Perlin noise - procedural content generation wiki. <http://pcg.wikidot.com/pcg-algorithm:perlin-noise>.
- [13] Various contributors. Simplex noise - procedural content generation wiki. <http://pcg.wikidot.com/pcg-algorithm:simplex-noise>.
- [14] Various contributors. Teleological - procedural content generation wiki. <http://pcg.wikidot.com/pcg-algorithm:teleological>.
- [15] Various contributors. Voronoi diagram - procedural content generation wiki. <http://pcg.wikidot.com/pcg-algorithm:voronoi-diagram>.
- [16] 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.
- [17] Joan Fons. Simplex noise lands in godot 3.1. <https://godotengine.org/article/simplex-noise-lands-godot-31>, September 2018.
- [18] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1:27, 10 2017.
- [19] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1(4):27, 2017.
- [20] Gingerageous Games. Autotiles opensimplex noise procedural generation godot 3.1 tutorial. <https://youtu.be/SBDs8hbs43w>, August 2019. [Online, Accessed 11-March-2023].
- [21] Gingerageous Games. Opensimplexnoisetilemaptutorial. <https://github.com/gingerageous/OpenSimplexNoiseTilemapTutorial>, March 2020. [Online, Accessed 11-March-2023].
- [22] Alexander Gillberg. Code that: L-system. <https://youtu.be/eY9XkJERiG0>, 2020.

- [23] Alexander Gillberg and contributors. An implementation of the rewriting system: Lindenmayer system. <https://github.com/codatproduction/Godots-L-System>, April 2020. [Online; accessed 10-April-2023].
- [24] Alexander Gillberg, Zishan Rahman, and contributors. An implementation of the rewriting system: Lindenmayer system. <https://github.com/Zishan-Rahman/Godots-L-System>, March 2023. [Online; accessed 10-April-2023, License MIT].
- [25] Hernan J. González. “why is gradient noise better quality than value noise?” - stack exchange answer. <https://math.stackexchange.com/a/184153>, August 2012. [Online, Accessed 16-April-2023, Stack Exchange Username “leonbloy”].
- [26] Ariel Manzur Juan Linietsky and the Godot community. Opensimplexnoise — godot engine (3.6) documentation in english. https://docs.godotengine.org/en/3.6/classes/class_opensimplexnoise.html. [Online, Accessed 17-April-2023, License CC-BY-3.0].
- [27] KDotJPG. Noise! <https://web.archive.org/web/20160529190516/http://uniblock.tumblr.com/post/97868843242/noise>, September 2014. [Online, Accessed 17-April-2023].
- [28] KdotJPG and contributors. Opensimplexnoise. <https://github.com/KdotJPG/OpenSimplex2>, 2014-present. [Online, Accessed 17-April-2023].
- [29] Kenney. 1-bit pack (version 1.2). <https://kenney.nl/assets/1-bit-pack>, November 2021. [Online, Accessed 23-March-2023, License CC0-1.0].
- [30] Marek Kopel and Grzegorz Maciejewski. Comparison of procedural noise-based environment generation methods. In Ngoc Thanh Nguyen, Bao Hung Hoang, Cong Phap Huynh, Dosam Hwang, Bogdan Trawiński, and Gottfried Vossen, editors, *Computational Collective Intelligence*, pages 878–887, Cham, 2020. Springer International Publishing.
- [31] Sebastian Lague. [unity] procedural object placement (e01: poisson disc sampling). <https://youtu.be/7WcmxyF07o>, November 2018. [Online, Accessed 11-April-2023].
- [32] Sebastian Lague. Poisson-disc-sampling. <https://github.com/SebLague/Poisson-Disc-Sampling>, November 2018-2020. [Online, Accessed 11-April-2023, License MIT].

- [33] 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.
- [34] Juan Linietsky, Ariel Manzur, and the Godot community. Fastnoiselite — godot engine (stable) documentation in english. https://docs.godotengine.org/en/4.0/classes/class_fastnoiselite.html.
- [35] Juan Linietsky, Ariel Manzur, and the Godot community. Gdscript — godot engine (4.0) documentation in english. https://docs.godotengine.org/en/4.0/classes/class_gdscript.html.
- [36] Juan Linietsky, Ariel Manzur, and the Godot community. Projectsettings — godot engine (stable) documentation in english. https://docs.godotengine.org/en/4.0/classes/class_projectsettings.html#properties. [Online, Accessed 16-April-2023, GDScript Warnings Classed as Properties, License CC-BY-3.0].
- [37] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Comput. Graph.*, 25(4):157–164, jul 1991.
- [38] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, page 157–164, New York, NY, USA, 1991. Association for Computing Machinery.
- [39] Nathan Lovato and contributors. Godot procedural generation. <https://github.com/GDQuest/godot-procedural-generation>.
- [40] Gabriela Ochoa. An introduction to lindenmayer systems: D0l-system. https://www1.biologie.uni-hamburg.de/b-online/e28_3/lsys.html#D0L-system, February 1998. [Online; accessed 11-April-2023].
- [41] Massachusetts Institute of Technology et al. The mit license - open source initiative. <https://opensource.org/license/mit/>. [Online, Accessed 18-April-2023].
- [42] Guilherme Oliveira. “intro to procedural levels in godot 3.1: Perlin noise tutorial”. <https://youtu.be/m6mu4uPGrMk>, November 2018.
- [43] Kenneth H. Perlin. Standard for perlin noise. <https://patents.google.com/patent/US6867776>, January 2002. [Online patent document, Accessed 16-April-2023].

- [44] Michael Powell. Spirit of iron: Simplectic noise. <https://web.archive.org/web/20160404092918/http://www.spiritofiron.com/2015/01/simplectic-noise.html>, January 2015. [Online, Accessed 17-April-2023].
- [45] Adarsh Pyarelal, Aditya Banerjee, and Kobus Barnard. Modular procedural generation for voxel maps. In Nikolos Gurney and Gita Sukthankar, editors, *Computational Theory of Mind for Human-Machine Teams*, pages 85–101, Cham, 2022. Springer Nature Switzerland.
- [46] Philippe Rivière and Jacob Rus. Poisson disk sampling functions. <https://observablehq.com/@fil/poisson-distribution-generators>, June 2022. [Online, Accessed 17-April-2023].
- [47] Martin Roberts. Maximal poisson disk sampling: an improved version of bridson’s algorithm. <http://extremelearning.com.au/an-improved-version-of-bridsons-algorithm-n-for-poisson-disc-sampling/>, November 2019. [Online, Accessed 17-April-2023].
- [48] Robson. An interactive webpage for generating images that simulate white noise. <https://robson.plus/white-noise-image-generator/>, April 2021. [Online, Accessed 16-April-2023].
- [49] Jacob Rus. A fork of roberts’s tweak to bridson’s algorithm for poisson disc sampling. <https://observablehq.com/@jruss/bridson-fork/2>, June 2022. [Online, Accessed 17-April-2023].
- [50] Tim Smith. Infiniteness and boundedness in 0l, dt0l, and t0l systems. In Adrian-Horia Dediu, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, pages 535–546, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [51] British Computer Society. Code of conduct for bcs members, version 8. <https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf>, June 2022. [Online, Accessed 18-April-2023].
- [52] Corina Ströbner. Criteria for naturalness in conceptual spaces. *Synthese*, 200, 04 2022.
- [53] Matt Tillman. Manhattan voronoi approximation. <http://www.pixelninja.design/manhattan-voronoi-approximation/>, April 2017.

- [54] Mick West. The inner product: A shattered reality. https://ubm-twvideo01.s3.amazonaws.com/o1/vault/GD_Mag_Archives/GDM_August_2006.pdf, August 2006. [Online, Accessed 17-April-2023, Pages 34-38].
- [55] Mick West. Random scattering: Creating realistic landscapes. http://web.archive.org/web/20080826103432/http://www.gamasutra.com/view/feature/1648/random_scattering_creating_.php?page=2, August 2008.
- [56] Wikipedia contributors. L-system — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=L-system&oldid=1124510226>, 2022. [Online; accessed 23-February-2023].
- [57] Wikipedia contributors. Opensimplex noise — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=OpenSimplex_noise&oldid=1102898483, 2022. [Online; accessed 16-April-2023].
- [58] Wikipedia contributors. Procedural generation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Procedural_generation&oldid=1143473446, 2023. [Online; accessed 10-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="TextLabel" type="Label" parent="."]
12 offset_right = 1152.0
13 offset_bottom = 23.0
14 autowrap_mode = 3
15
16 [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://www1.biologie.uni-hamburg.de/b-online/
   e28_3/lsys.html#D0L-system
6
7  @export_enum("basic", "choices", "deterministic") var choices:
   String = "choices"
8
9  @export var axiom: String
10
11 @onready var string: String
12
13 @onready var timer = $Timer
14
15 @onready var label = $TextLabel
16
17 @onready var rules: Array[Dictionary]
18
19
20 func set_values():
21     if choices == "basic":
22         rules = [
23             {
24                 "from": "F",
25                 "to": "F+F"
26             }
27         ]
28         axiom = "F+"
29     elif choices == "choices":
30         rules = [
31             {
32                 "from": "F",
33                 "to": "F+---FFFF+F+-FF"
34             }
35         ]
```

```

30     axiom = "F+F+F+F"
31     elif choices == "deterministic":
32         rules = [
33             {
34                 "from": "a",
35                 "to": "ab"
36             },
37             {
38                 "from": "b",
39                 "to": "a"
40             }
41         ]
42         axiom = "b"
43
44     func _ready():
45         set_values()
46         string = axiom
47         label.size.x = get_viewport().size.x
48         label.text = string
49         print(len(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="LSystem 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
22  [rendering]
23
24  environment/defaults/default_clear_color=Color(0, 0, 0, 1)

```

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 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 PLAYER_SPRITE: Vector2i = Vector2i(24, 7)
9  var player_placement_cell: Vector2i
10 const rings: Array[Vector2i] = [
11     Vector2i(43, 6),
12     Vector2i(44, 6),
13     Vector2i(45, 6),
14     Vector2i(46, 6)
15 ]
16 var ring_placement_cell: Vector2i
17
18 # Called when the node enters the scene tree for the first time.
19 func _ready() -> void:
20     randomize()
21     var start_time: float = Time.get_ticks_msec()
22     l_system.paint()
23     place_player()
24     place_ring()
25     var new_time: float = Time.get_ticks_msec() - start_time
26     print("Time taken: " + str(new_time) + "ms")
27     $AcceptDialog.dialog_text = "You're a hollow Golem who seeks the
        ultimate treasure; a ring that's got something on top of it
        . It's somewhere in this large village and barely visible to
```

```

        your naked eyes, which took us " + str(new_time) + "
milliseconds to generate (" + str(new_time / 1000.0) + "
seconds), but you'll stop at nothing to get what you want.
You can chow down every tree and fauna that stands in your
way of the ring, but your Achilles heel is any bricks and
mortar, which WILL make you stop at your tracks. Since it's
easy to get lost in here, we'll tell you that you're in
position " + str(player_placement_cell) + " in this big
village of size " + str(Vector2i(x_tile_range, y_tile_range)
) + ". The ring is in position " + str(ring_placement_cell)
+ ", but it is YOUR job to traverse the village, chow down
the trees and get it for yourself, so are you ready to
attain the treasure that is rightfully yours?!"

28     $AcceptDialog.visible = true
29     $AcceptDialog.confirmed.connect(_on_AcceptDialog_closed)
30     $AcceptDialog.canceled.connect(_on_AcceptDialog_closed)
31     $WinDialog.confirmed.connect(_on_WinDialog_confirmed)
32     $WinDialog.canceled.connect(_on_WinDialog_canceled)
33     get_tree().paused = true
34
35     func _on_WinDialog_confirmed() -> void:
36         get_tree().reload_current_scene()
37
38     func _on_WinDialog_canceled() -> void:
39         get_tree().quit()
40
41     func _on_AcceptDialog_closed() -> void:
42         $AcceptDialog.visible = false
43         get_tree().paused = false
44
45     func _get_random_placement_cell() -> Vector2i:
46         return Vector2i(randi() % x_tile_range, randi() % y_tile_range)
47

```

```

48  func place_player() -> void:
49      player_placement_cell = _get_random_placement_cell()
50      while get_used_cells(0).has(player_placement_cell):
51          player_placement_cell = _get_random_placement_cell()
52      set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
53
54  func place_ring() -> void:
55      ring_placement_cell = _get_random_placement_cell()
56      while get_used_cells(0).has(ring_placement_cell):
57          ring_placement_cell = _get_random_placement_cell()
58      set_cell(0, ring_placement_cell, 0, rings.pick_random())
59
60  func _is_not_out_of_bounds(cell: Vector2i) -> bool:
61      return cell.x >= 0 and cell.x < x_tile_range and cell.y >= 0 and
        cell.y < y_tile_range
62
63  func _physics_process(_delta: float) -> void:
64      var previous_cell: Vector2i = player_placement_cell
65      var direction: Vector2i = Vector2i.ZERO
66      if Input.is_action_pressed("ui_up"): direction = Vector2i.UP
67      elif Input.is_action_pressed("ui_down"): direction = Vector2i.
        DOWN
68      elif Input.is_action_pressed("ui_left"): direction = Vector2i.
        LEFT
69      elif Input.is_action_pressed("ui_right"): direction = Vector2i.
        RIGHT
70      var new_placement_cell: Vector2i = player_placement_cell +
        direction
71      if (not get_used_cells(0).has(new_placement_cell) or l_system.
        trees.has(get_cell_atlas_coords(0, new_placement_cell)) or
        new_placement_cell == ring_placement_cell) and
        _is_not_out_of_bounds(new_placement_cell):
72      player_placement_cell = new_placement_cell

```



```

73         set_cell(0, previous_cell, 0) # deletes contents of previous
           cell (atlas_coords = Vector2i(-1, -1))
74         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
75         if player_placement_cell == ring_placement_cell:
76             $WinDialog.visible = true
77             get_tree().paused = true
78
79     # ALGORITHM AND CUSTOM EXPORT VARIABLES ARE IN LSYSTEM NODE

```

C.3.6 tile_map.tscn

```

1  [gd_scene load_steps=6 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  [ext_resource type="PackedScene" uid="uid://cau5jgogdnf53" path="
    res://accept_dialog.tscn" id="4_060oh"]
7  [ext_resource type="PackedScene" uid="uid://b5q8ovcigrvyr" path="
    res://win_dialog.tscn" id="5_3s48a"]
8
9  [node name="TileMap" type="TileMap"]
10 tile_set = ExtResource("1_l3nwg")
11 format = 2
12 layer_0/name = "Things"
13 script = ExtResource("2_wrxl8")
14
15 [node name="LSystem" parent="." instance=ExtResource("3_ktw1n")]
16
17 [node name="AcceptDialog" parent="." instance=ExtResource("4_060oh")

```

```

    )]
18
19 [node name="WinDialog" parent="." instance=ExtResource("5_3s48a")]

```

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 PLAYER_SPRITE: Vector2i = Vector2i(24, 7)
9  var player_placement_cell: Vector2i
10 const rings: Array[Vector2i] = [
11     Vector2i(43, 6),
12     Vector2i(44, 6),
13     Vector2i(45, 6),
14     Vector2i(46, 6)
15 ]
16 var ring_placement_cell: Vector2i
17
18 # Called when the node enters the scene tree for the first time.
19 func _ready() -> void:
20     randomize()
21     var start_time: float = Time.get_ticks_msec()
22     l_system.paint()
23     place_player()
24     place_ring()

```

```

25     var new_time: float = Time.get_ticks_msec() - start_time
26     print("Time taken: " + str(new_time) + "ms")
27     $AcceptDialog.dialog_text = "You're a hollow Golem who seeks the
        ultimate treasure; a ring that's got something on top of it
        . It's somewhere in this large village and barely visible to
        your naked eyes, which took us " + str(new_time) + "
        milliseconds to generate (" + str(new_time / 1000.0) + "
        seconds), but you'll stop at nothing to get what you want.
        You can chow down every tree and fauna that stands in your
        way of the ring, but your Achilles heel is any bricks and
        mortar, which WILL make you stop at your tracks. Since it's
        easy to get lost in here, we'll tell you that you're in
        position " + str(player_placement_cell) + " in this big
        village of size " + str(Vector2i(x_tile_range, y_tile_range)
        ) + ". The ring is in position " + str(ring_placement_cell)
        + ", but it is YOUR job to traverse the village, chow down
        the trees and get it for yourself, so are you ready to
        attain the treasure that is rightfully yours?!"

28     $AcceptDialog.visible = true
29     $AcceptDialog.confirmed.connect(_on_AcceptDialog_closed)
30     $AcceptDialog.canceled.connect(_on_AcceptDialog_closed)
31     $WinDialog.confirmed.connect(_on_WinDialog_confirmed)
32     $WinDialog.canceled.connect(_on_WinDialog_canceled)
33     get_tree().paused = true
34
35     func _on_WinDialog_confirmed() -> void:
36         get_tree().reload_current_scene()
37
38     func _on_WinDialog_canceled() -> void:
39         get_tree().quit()
40
41     func _on_AcceptDialog_closed() -> void:
42         $AcceptDialog.visible = false

```

```

43     get_tree().paused = false
44
45     func _get_random_placement_cell() -> Vector2i:
46         return Vector2i(randi() % x_tile_range, randi() % y_tile_range)
47
48     func place_player() -> void:
49         player_placement_cell = _get_random_placement_cell()
50         while get_used_cells(0).has(player_placement_cell):
51             player_placement_cell = _get_random_placement_cell()
52         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
53
54     func place_ring() -> void:
55         ring_placement_cell = _get_random_placement_cell()
56         while get_used_cells(0).has(ring_placement_cell):
57             ring_placement_cell = _get_random_placement_cell()
58         set_cell(0, ring_placement_cell, 0, rings.pick_random())
59
60     func _is_not_out_of_bounds(cell: Vector2i) -> bool:
61         return cell.x >= 0 and cell.x < x_tile_range and cell.y >= 0 and
           cell.y < y_tile_range
62
63     func _physics_process(_delta: float) -> void:
64         var previous_cell: Vector2i = player_placement_cell
65         var direction: Vector2i = Vector2i.ZERO
66         if Input.is_action_pressed("ui_up"): direction = Vector2i.UP
67         elif Input.is_action_pressed("ui_down"): direction = Vector2i.
           DOWN
68         elif Input.is_action_pressed("ui_left"): direction = Vector2i.
           LEFT
69         elif Input.is_action_pressed("ui_right"): direction = Vector2i.
           RIGHT
70         var new_placement_cell: Vector2i = player_placement_cell +
           direction

```

```

71     if (not get_used_cells(0).has(new_placement_cell) or l_system.
        trees.has(get_cell_atlas_coords(0, new_placement_cell)) or
        new_placement_cell == ring_placement_cell) and
        _is_not_out_of_bounds(new_placement_cell):
72         player_placement_cell = new_placement_cell
73         set_cell(0, previous_cell, 0) # deletes contents of previous
            cell (atlas_coords = Vector2i(-1, -1))
74         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
75         if player_placement_cell == ring_placement_cell:
76             $WinDialog.visible = true
77             get_tree().paused = true
78
79     # ALGORITHM AND CUSTOM EXPORT VARIABLES ARE IN LSYSTEM NODE

```

C.3.8 accept_dialog.tscn

```

1     [gd_scene format=3 uid="uid://cau5jgogdnf53"]
2
3     [node name="AcceptDialog" type="AcceptDialog"]
4     title = "Tree-Munching Time!"
5     position = Vector2i(326, 100)
6     size = Vector2i(500, 421)
7     mouse_passthrough = true
8     ok_button_text = "Bring it on!"
9     dialog_text = "You're a hollow Golem who seeks the ultimate
        treasure; a ring that's got something on top of it. It's
        somewhere in this large village and barely visible to your
        naked eyes, but you'll stop at nothing to get what you want.
        You can chow down every tree and fauna that stands in your way
        of the ring, but your Achilles heel is any bricks and mortar,
        which will make you stop at your tracks. Are you ready to
        attain your treasure?w Golem in a black-and-white world, in

```

```

        search for your most desired treasure. It's a ring with
        something on top of it. And you'll stop at nothing to get what
        you want. You can chow down every tree and fauna that stands in
        your way of the ring, but your Achilles heel is any bricks and
        mortar, which will make you stop at your tracks. Are you ready
        to attain the treasure that is rightfully yours?!"
10  dialog_autowrap = true

```

C.3.9 win_dialog.tscn

```

1  [gd_scene format=3 uid="uid://b5q8ovcigrvyr"]
2
3  [node name="WinDialog" type="ConfirmationDialog"]
4  title = "You Found the Treasure!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Get Me a New Village"
9  dialog_text = "You found your treasure! Well done, you!"
10
11  Would you like to travel to a new village in the hopes of finding
        another ring? Or would you like to take your treasure home now?
        "
12  dialog_autowrap = true
13  cancel_button_text = "Get Me Out of Here"

```

C.3.10 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.3.11 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
21
22 [input]
23
24 reset_position={
25 "deadzone": 0.5,
26 "events": [Object(InputEventKey,"resource_local_to_scene":false,"
27             resource_name":"","device":-1,"window_id":0,"alt_pressed":false,
28             "shift_pressed":false,"ctrl_pressed":false,"meta_pressed":
29             false,"pressed":false,"keycode":71,"physical_keycode":0,"
30             key_label":0,"unicode":103,"echo":false,"script":null)
31 , Object(InputEventMouseButton,"resource_local_to_scene":false,"
32           resource_name":"","device":-1,"window_id":0,"alt_pressed":false,
33           "shift_pressed":false,"ctrl_pressed":false,"meta_pressed":
34           false,"button_mask":2,"position":Vector2(75, 12),"
35           global_position":Vector2(78, 44),"factor":1.0,"button_index
36           ":2,"pressed":true,"double_click":false,"script":null)
37 ]

```

```

29     }
30
31     [rendering]
32
33     environment/defaults/default_clear_color=Color(0, 0, 0, 1)

```

C.4.4 tile_map.tscn

```

1     [gd_scene load_steps=7 format=3 uid="uid://d6lxn73sfbsrt"]
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     [ext_resource type="PackedScene" path="res://accept_dialog.tscn" id
        ="3_y08lj"]
6     [ext_resource type="PackedScene" path="res://win_dialog.tscn" id="4
        _fkys0"]
7
8     [sub_resource type="TileSetAtlasSource" id="
        TileSetAtlasSource_6h0bd"]
9     texture = ExtResource("1_o183d")
10    0:0/0 = 0
11    1:0/0 = 0
12    2:0/0 = 0
13    3:0/0 = 0
14    4:0/0 = 0
15    5:0/0 = 0
16    6:0/0 = 0
17    7:0/0 = 0
18    8:0/0 = 0
19    9:0/0 = 0
20    10:0/0 = 0

```

21 11:0/0 = 0
22 12:0/0 = 0
23 13:0/0 = 0
24 14:0/0 = 0
25 15:0/0 = 0
26 16:0/0 = 0
27 17:0/0 = 0
28 18:0/0 = 0
29 19:0/0 = 0
30 20:0/0 = 0
31 21:0/0 = 0
32 22:0/0 = 0
33 23:0/0 = 0
34 24:0/0 = 0
35 25:0/0 = 0
36 26:0/0 = 0
37 27:0/0 = 0
38 28:0/0 = 0
39 29:0/0 = 0
40 30:0/0 = 0
41 31:0/0 = 0
42 32:0/0 = 0
43 33:0/0 = 0
44 34:0/0 = 0
45 35:0/0 = 0
46 36:0/0 = 0
47 37:0/0 = 0
48 38:0/0 = 0
49 39:0/0 = 0
50 40:0/0 = 0
51 41:0/0 = 0
52 42:0/0 = 0
53 43:0/0 = 0

54 44:0/0 = 0
55 45:0/0 = 0
56 46:0/0 = 0
57 47:0/0 = 0
58 48:0/0 = 0
59 0:1/0 = 0
60 1:1/0 = 0
61 2:1/0 = 0
62 3:1/0 = 0
63 4:1/0 = 0
64 5:1/0 = 0
65 6:1/0 = 0
66 7:1/0 = 0
67 8:1/0 = 0
68 9:1/0 = 0
69 10:1/0 = 0
70 11:1/0 = 0
71 12:1/0 = 0
72 13:1/0 = 0
73 14:1/0 = 0
74 15:1/0 = 0
75 16:1/0 = 0
76 17:1/0 = 0
77 18:1/0 = 0
78 19:1/0 = 0
79 20:1/0 = 0
80 21:1/0 = 0
81 22:1/0 = 0
82 23:1/0 = 0
83 24:1/0 = 0
84 25:1/0 = 0
85 26:1/0 = 0
86 27:1/0 = 0

87 28:1/0 = 0
88 29:1/0 = 0
89 30:1/0 = 0
90 31:1/0 = 0
91 32:1/0 = 0
92 33:1/0 = 0
93 34:1/0 = 0
94 35:1/0 = 0
95 36:1/0 = 0
96 37:1/0 = 0
97 38:1/0 = 0
98 39:1/0 = 0
99 40:1/0 = 0
100 41:1/0 = 0
101 42:1/0 = 0
102 43:1/0 = 0
103 44:1/0 = 0
104 45:1/0 = 0
105 46:1/0 = 0
106 47:1/0 = 0
107 48:1/0 = 0
108 0:2/0 = 0
109 1:2/0 = 0
110 2:2/0 = 0
111 3:2/0 = 0
112 4:2/0 = 0
113 5:2/0 = 0
114 6:2/0 = 0
115 7:2/0 = 0
116 8:2/0 = 0
117 9:2/0 = 0
118 10:2/0 = 0
119 11:2/0 = 0

120	$12:2/0 = 0$
121	$13:2/0 = 0$
122	$14:2/0 = 0$
123	$15:2/0 = 0$
124	$16:2/0 = 0$
125	$17:2/0 = 0$
126	$18:2/0 = 0$
127	$19:2/0 = 0$
128	$20:2/0 = 0$
129	$21:2/0 = 0$
130	$22:2/0 = 0$
131	$23:2/0 = 0$
132	$24:2/0 = 0$
133	$25:2/0 = 0$
134	$26:2/0 = 0$
135	$27:2/0 = 0$
136	$28:2/0 = 0$
137	$29:2/0 = 0$
138	$30:2/0 = 0$
139	$31:2/0 = 0$
140	$32:2/0 = 0$
141	$33:2/0 = 0$
142	$34:2/0 = 0$
143	$35:2/0 = 0$
144	$36:2/0 = 0$
145	$37:2/0 = 0$
146	$38:2/0 = 0$
147	$39:2/0 = 0$
148	$40:2/0 = 0$
149	$41:2/0 = 0$
150	$42:2/0 = 0$
151	$43:2/0 = 0$
152	$44:2/0 = 0$

153	$45:2/0 = 0$
154	$46:2/0 = 0$
155	$47:2/0 = 0$
156	$48:2/0 = 0$
157	$0:3/0 = 0$
158	$1:3/0 = 0$
159	$2:3/0 = 0$
160	$3:3/0 = 0$
161	$4:3/0 = 0$
162	$5:3/0 = 0$
163	$6:3/0 = 0$
164	$7:3/0 = 0$
165	$8:3/0 = 0$
166	$9:3/0 = 0$
167	$10:3/0 = 0$
168	$11:3/0 = 0$
169	$12:3/0 = 0$
170	$13:3/0 = 0$
171	$14:3/0 = 0$
172	$15:3/0 = 0$
173	$16:3/0 = 0$
174	$17:3/0 = 0$
175	$18:3/0 = 0$
176	$19:3/0 = 0$
177	$20:3/0 = 0$
178	$21:3/0 = 0$
179	$22:3/0 = 0$
180	$23:3/0 = 0$
181	$24:3/0 = 0$
182	$25:3/0 = 0$
183	$26:3/0 = 0$
184	$27:3/0 = 0$
185	$28:3/0 = 0$

186 29:3/0 = 0
187 30:3/0 = 0
188 31:3/0 = 0
189 32:3/0 = 0
190 33:3/0 = 0
191 34:3/0 = 0
192 35:3/0 = 0
193 36:3/0 = 0
194 37:3/0 = 0
195 38:3/0 = 0
196 39:3/0 = 0
197 40:3/0 = 0
198 41:3/0 = 0
199 42:3/0 = 0
200 43:3/0 = 0
201 44:3/0 = 0
202 45:3/0 = 0
203 46:3/0 = 0
204 47:3/0 = 0
205 48:3/0 = 0
206 0:4/0 = 0
207 1:4/0 = 0
208 2:4/0 = 0
209 3:4/0 = 0
210 4:4/0 = 0
211 5:4/0 = 0
212 6:4/0 = 0
213 7:4/0 = 0
214 8:4/0 = 0
215 9:4/0 = 0
216 10:4/0 = 0
217 11:4/0 = 0
218 12:4/0 = 0

219	$13:4/0 = 0$
220	$14:4/0 = 0$
221	$15:4/0 = 0$
222	$16:4/0 = 0$
223	$17:4/0 = 0$
224	$18:4/0 = 0$
225	$19:4/0 = 0$
226	$20:4/0 = 0$
227	$21:4/0 = 0$
228	$22:4/0 = 0$
229	$23:4/0 = 0$
230	$24:4/0 = 0$
231	$25:4/0 = 0$
232	$26:4/0 = 0$
233	$27:4/0 = 0$
234	$28:4/0 = 0$
235	$29:4/0 = 0$
236	$30:4/0 = 0$
237	$31:4/0 = 0$
238	$32:4/0 = 0$
239	$33:4/0 = 0$
240	$34:4/0 = 0$
241	$35:4/0 = 0$
242	$36:4/0 = 0$
243	$37:4/0 = 0$
244	$38:4/0 = 0$
245	$39:4/0 = 0$
246	$40:4/0 = 0$
247	$41:4/0 = 0$
248	$42:4/0 = 0$
249	$43:4/0 = 0$
250	$44:4/0 = 0$
251	$45:4/0 = 0$

252	$46:4/0 = 0$
253	$47:4/0 = 0$
254	$48:4/0 = 0$
255	$0:5/0 = 0$
256	$1:5/0 = 0$
257	$2:5/0 = 0$
258	$3:5/0 = 0$
259	$4:5/0 = 0$
260	$5:5/0 = 0$
261	$6:5/0 = 0$
262	$7:5/0 = 0$
263	$8:5/0 = 0$
264	$9:5/0 = 0$
265	$10:5/0 = 0$
266	$11:5/0 = 0$
267	$12:5/0 = 0$
268	$13:5/0 = 0$
269	$14:5/0 = 0$
270	$15:5/0 = 0$
271	$16:5/0 = 0$
272	$17:5/0 = 0$
273	$18:5/0 = 0$
274	$19:5/0 = 0$
275	$20:5/0 = 0$
276	$21:5/0 = 0$
277	$22:5/0 = 0$
278	$23:5/0 = 0$
279	$24:5/0 = 0$
280	$25:5/0 = 0$
281	$26:5/0 = 0$
282	$27:5/0 = 0$
283	$28:5/0 = 0$
284	$29:5/0 = 0$

285 30:5/0 = 0
286 31:5/0 = 0
287 32:5/0 = 0
288 33:5/0 = 0
289 34:5/0 = 0
290 35:5/0 = 0
291 36:5/0 = 0
292 37:5/0 = 0
293 38:5/0 = 0
294 39:5/0 = 0
295 40:5/0 = 0
296 41:5/0 = 0
297 42:5/0 = 0
298 43:5/0 = 0
299 44:5/0 = 0
300 45:5/0 = 0
301 46:5/0 = 0
302 47:5/0 = 0
303 48:5/0 = 0
304 0:6/0 = 0
305 1:6/0 = 0
306 2:6/0 = 0
307 3:6/0 = 0
308 4:6/0 = 0
309 5:6/0 = 0
310 6:6/0 = 0
311 7:6/0 = 0
312 8:6/0 = 0
313 9:6/0 = 0
314 10:6/0 = 0
315 11:6/0 = 0
316 12:6/0 = 0
317 13:6/0 = 0

318 14:6/0 = 0
319 15:6/0 = 0
320 16:6/0 = 0
321 17:6/0 = 0
322 18:6/0 = 0
323 19:6/0 = 0
324 20:6/0 = 0
325 21:6/0 = 0
326 22:6/0 = 0
327 23:6/0 = 0
328 24:6/0 = 0
329 25:6/0 = 0
330 26:6/0 = 0
331 27:6/0 = 0
332 28:6/0 = 0
333 29:6/0 = 0
334 30:6/0 = 0
335 31:6/0 = 0
336 32:6/0 = 0
337 33:6/0 = 0
338 34:6/0 = 0
339 35:6/0 = 0
340 36:6/0 = 0
341 37:6/0 = 0
342 38:6/0 = 0
343 39:6/0 = 0
344 40:6/0 = 0
345 41:6/0 = 0
346 42:6/0 = 0
347 43:6/0 = 0
348 44:6/0 = 0
349 45:6/0 = 0
350 46:6/0 = 0

351 47:6/0 = 0
352 48:6/0 = 0
353 0:7/0 = 0
354 1:7/0 = 0
355 2:7/0 = 0
356 3:7/0 = 0
357 4:7/0 = 0
358 5:7/0 = 0
359 6:7/0 = 0
360 7:7/0 = 0
361 8:7/0 = 0
362 9:7/0 = 0
363 10:7/0 = 0
364 11:7/0 = 0
365 12:7/0 = 0
366 13:7/0 = 0
367 14:7/0 = 0
368 15:7/0 = 0
369 16:7/0 = 0
370 17:7/0 = 0
371 18:7/0 = 0
372 19:7/0 = 0
373 20:7/0 = 0
374 21:7/0 = 0
375 22:7/0 = 0
376 23:7/0 = 0
377 24:7/0 = 0
378 25:7/0 = 0
379 26:7/0 = 0
380 27:7/0 = 0
381 28:7/0 = 0
382 29:7/0 = 0
383 30:7/0 = 0

384 $31:7/0 = 0$
385 $32:7/0 = 0$
386 $33:7/0 = 0$
387 $34:7/0 = 0$
388 $35:7/0 = 0$
389 $36:7/0 = 0$
390 $37:7/0 = 0$
391 $38:7/0 = 0$
392 $39:7/0 = 0$
393 $40:7/0 = 0$
394 $41:7/0 = 0$
395 $42:7/0 = 0$
396 $43:7/0 = 0$
397 $44:7/0 = 0$
398 $45:7/0 = 0$
399 $46:7/0 = 0$
400 $47:7/0 = 0$
401 $48:7/0 = 0$
402 $0:8/0 = 0$
403 $1:8/0 = 0$
404 $2:8/0 = 0$
405 $3:8/0 = 0$
406 $4:8/0 = 0$
407 $5:8/0 = 0$
408 $6:8/0 = 0$
409 $7:8/0 = 0$
410 $8:8/0 = 0$
411 $9:8/0 = 0$
412 $10:8/0 = 0$
413 $11:8/0 = 0$
414 $12:8/0 = 0$
415 $13:8/0 = 0$
416 $14:8/0 = 0$

417 15:8/0 = 0
418 16:8/0 = 0
419 17:8/0 = 0
420 18:8/0 = 0
421 19:8/0 = 0
422 20:8/0 = 0
423 21:8/0 = 0
424 22:8/0 = 0
425 23:8/0 = 0
426 24:8/0 = 0
427 25:8/0 = 0
428 26:8/0 = 0
429 27:8/0 = 0
430 28:8/0 = 0
431 29:8/0 = 0
432 30:8/0 = 0
433 31:8/0 = 0
434 32:8/0 = 0
435 33:8/0 = 0
436 34:8/0 = 0
437 35:8/0 = 0
438 36:8/0 = 0
439 37:8/0 = 0
440 38:8/0 = 0
441 39:8/0 = 0
442 40:8/0 = 0
443 41:8/0 = 0
444 42:8/0 = 0
445 43:8/0 = 0
446 44:8/0 = 0
447 45:8/0 = 0
448 46:8/0 = 0
449 47:8/0 = 0

450 $48:8/0 = 0$
451 $0:9/0 = 0$
452 $1:9/0 = 0$
453 $2:9/0 = 0$
454 $3:9/0 = 0$
455 $4:9/0 = 0$
456 $5:9/0 = 0$
457 $6:9/0 = 0$
458 $7:9/0 = 0$
459 $8:9/0 = 0$
460 $9:9/0 = 0$
461 $10:9/0 = 0$
462 $11:9/0 = 0$
463 $12:9/0 = 0$
464 $13:9/0 = 0$
465 $14:9/0 = 0$
466 $15:9/0 = 0$
467 $16:9/0 = 0$
468 $17:9/0 = 0$
469 $18:9/0 = 0$
470 $19:9/0 = 0$
471 $20:9/0 = 0$
472 $21:9/0 = 0$
473 $22:9/0 = 0$
474 $23:9/0 = 0$
475 $24:9/0 = 0$
476 $25:9/0 = 0$
477 $26:9/0 = 0$
478 $27:9/0 = 0$
479 $28:9/0 = 0$
480 $29:9/0 = 0$
481 $30:9/0 = 0$
482 $31:9/0 = 0$

483 32:9/0 = 0
484 33:9/0 = 0
485 34:9/0 = 0
486 35:9/0 = 0
487 36:9/0 = 0
488 37:9/0 = 0
489 38:9/0 = 0
490 39:9/0 = 0
491 40:9/0 = 0
492 41:9/0 = 0
493 42:9/0 = 0
494 43:9/0 = 0
495 44:9/0 = 0
496 45:9/0 = 0
497 46:9/0 = 0
498 47:9/0 = 0
499 48:9/0 = 0
500 0:10/0 = 0
501 1:10/0 = 0
502 2:10/0 = 0
503 3:10/0 = 0
504 4:10/0 = 0
505 5:10/0 = 0
506 6:10/0 = 0
507 7:10/0 = 0
508 8:10/0 = 0
509 9:10/0 = 0
510 10:10/0 = 0
511 11:10/0 = 0
512 12:10/0 = 0
513 13:10/0 = 0
514 14:10/0 = 0
515 15:10/0 = 0

516 16:10/0 = 0
517 17:10/0 = 0
518 18:10/0 = 0
519 19:10/0 = 0
520 20:10/0 = 0
521 21:10/0 = 0
522 22:10/0 = 0
523 23:10/0 = 0
524 24:10/0 = 0
525 25:10/0 = 0
526 26:10/0 = 0
527 27:10/0 = 0
528 28:10/0 = 0
529 29:10/0 = 0
530 30:10/0 = 0
531 31:10/0 = 0
532 32:10/0 = 0
533 33:10/0 = 0
534 34:10/0 = 0
535 35:10/0 = 0
536 36:10/0 = 0
537 37:10/0 = 0
538 38:10/0 = 0
539 39:10/0 = 0
540 40:10/0 = 0
541 41:10/0 = 0
542 42:10/0 = 0
543 43:10/0 = 0
544 44:10/0 = 0
545 45:10/0 = 0
546 46:10/0 = 0
547 47:10/0 = 0
548 48:10/0 = 0

549 0:11/0 = 0
550 1:11/0 = 0
551 2:11/0 = 0
552 3:11/0 = 0
553 4:11/0 = 0
554 5:11/0 = 0
555 6:11/0 = 0
556 7:11/0 = 0
557 8:11/0 = 0
558 9:11/0 = 0
559 10:11/0 = 0
560 11:11/0 = 0
561 12:11/0 = 0
562 13:11/0 = 0
563 14:11/0 = 0
564 15:11/0 = 0
565 16:11/0 = 0
566 17:11/0 = 0
567 18:11/0 = 0
568 19:11/0 = 0
569 20:11/0 = 0
570 21:11/0 = 0
571 22:11/0 = 0
572 23:11/0 = 0
573 24:11/0 = 0
574 25:11/0 = 0
575 26:11/0 = 0
576 27:11/0 = 0
577 28:11/0 = 0
578 29:11/0 = 0
579 30:11/0 = 0
580 31:11/0 = 0
581 32:11/0 = 0

582 33:11/0 = 0
583 34:11/0 = 0
584 35:11/0 = 0
585 36:11/0 = 0
586 37:11/0 = 0
587 38:11/0 = 0
588 39:11/0 = 0
589 40:11/0 = 0
590 41:11/0 = 0
591 42:11/0 = 0
592 43:11/0 = 0
593 44:11/0 = 0
594 45:11/0 = 0
595 46:11/0 = 0
596 47:11/0 = 0
597 48:11/0 = 0
598 0:12/0 = 0
599 1:12/0 = 0
600 2:12/0 = 0
601 3:12/0 = 0
602 4:12/0 = 0
603 5:12/0 = 0
604 6:12/0 = 0
605 7:12/0 = 0
606 8:12/0 = 0
607 9:12/0 = 0
608 10:12/0 = 0
609 11:12/0 = 0
610 12:12/0 = 0
611 13:12/0 = 0
612 14:12/0 = 0
613 15:12/0 = 0
614 16:12/0 = 0

615 17:12/0 = 0
616 18:12/0 = 0
617 19:12/0 = 0
618 20:12/0 = 0
619 21:12/0 = 0
620 22:12/0 = 0
621 23:12/0 = 0
622 24:12/0 = 0
623 25:12/0 = 0
624 26:12/0 = 0
625 27:12/0 = 0
626 28:12/0 = 0
627 29:12/0 = 0
628 30:12/0 = 0
629 31:12/0 = 0
630 32:12/0 = 0
631 33:12/0 = 0
632 34:12/0 = 0
633 35:12/0 = 0
634 36:12/0 = 0
635 37:12/0 = 0
636 38:12/0 = 0
637 39:12/0 = 0
638 40:12/0 = 0
639 41:12/0 = 0
640 42:12/0 = 0
641 43:12/0 = 0
642 44:12/0 = 0
643 45:12/0 = 0
644 46:12/0 = 0
645 47:12/0 = 0
646 48:12/0 = 0
647 0:13/0 = 0

648 1:13/0 = 0
649 2:13/0 = 0
650 3:13/0 = 0
651 4:13/0 = 0
652 5:13/0 = 0
653 6:13/0 = 0
654 7:13/0 = 0
655 8:13/0 = 0
656 9:13/0 = 0
657 10:13/0 = 0
658 11:13/0 = 0
659 12:13/0 = 0
660 13:13/0 = 0
661 14:13/0 = 0
662 15:13/0 = 0
663 16:13/0 = 0
664 17:13/0 = 0
665 18:13/0 = 0
666 19:13/0 = 0
667 20:13/0 = 0
668 21:13/0 = 0
669 22:13/0 = 0
670 23:13/0 = 0
671 24:13/0 = 0
672 25:13/0 = 0
673 26:13/0 = 0
674 27:13/0 = 0
675 28:13/0 = 0
676 29:13/0 = 0
677 30:13/0 = 0
678 31:13/0 = 0
679 32:13/0 = 0
680 33:13/0 = 0

681 34:13/0 = 0
682 35:13/0 = 0
683 36:13/0 = 0
684 37:13/0 = 0
685 38:13/0 = 0
686 39:13/0 = 0
687 40:13/0 = 0
688 41:13/0 = 0
689 42:13/0 = 0
690 43:13/0 = 0
691 44:13/0 = 0
692 45:13/0 = 0
693 46:13/0 = 0
694 47:13/0 = 0
695 48:13/0 = 0
696 0:14/0 = 0
697 1:14/0 = 0
698 2:14/0 = 0
699 3:14/0 = 0
700 4:14/0 = 0
701 5:14/0 = 0
702 6:14/0 = 0
703 7:14/0 = 0
704 8:14/0 = 0
705 9:14/0 = 0
706 10:14/0 = 0
707 11:14/0 = 0
708 12:14/0 = 0
709 13:14/0 = 0
710 14:14/0 = 0
711 15:14/0 = 0
712 16:14/0 = 0
713 17:14/0 = 0

714	$18:14/0 = 0$
715	$19:14/0 = 0$
716	$20:14/0 = 0$
717	$21:14/0 = 0$
718	$22:14/0 = 0$
719	$23:14/0 = 0$
720	$24:14/0 = 0$
721	$25:14/0 = 0$
722	$26:14/0 = 0$
723	$27:14/0 = 0$
724	$28:14/0 = 0$
725	$29:14/0 = 0$
726	$30:14/0 = 0$
727	$31:14/0 = 0$
728	$32:14/0 = 0$
729	$33:14/0 = 0$
730	$34:14/0 = 0$
731	$35:14/0 = 0$
732	$36:14/0 = 0$
733	$37:14/0 = 0$
734	$38:14/0 = 0$
735	$39:14/0 = 0$
736	$40:14/0 = 0$
737	$41:14/0 = 0$
738	$42:14/0 = 0$
739	$43:14/0 = 0$
740	$44:14/0 = 0$
741	$45:14/0 = 0$
742	$46:14/0 = 0$
743	$47:14/0 = 0$
744	$48:14/0 = 0$
745	$0:15/0 = 0$
746	$1:15/0 = 0$

747 2:15/0 = 0
748 3:15/0 = 0
749 4:15/0 = 0
750 5:15/0 = 0
751 6:15/0 = 0
752 7:15/0 = 0
753 8:15/0 = 0
754 9:15/0 = 0
755 10:15/0 = 0
756 11:15/0 = 0
757 12:15/0 = 0
758 13:15/0 = 0
759 14:15/0 = 0
760 15:15/0 = 0
761 16:15/0 = 0
762 17:15/0 = 0
763 18:15/0 = 0
764 19:15/0 = 0
765 20:15/0 = 0
766 21:15/0 = 0
767 22:15/0 = 0
768 23:15/0 = 0
769 24:15/0 = 0
770 25:15/0 = 0
771 26:15/0 = 0
772 27:15/0 = 0
773 28:15/0 = 0
774 29:15/0 = 0
775 30:15/0 = 0
776 31:15/0 = 0
777 32:15/0 = 0
778 33:15/0 = 0
779 34:15/0 = 0

780	$35:15/0 = 0$
781	$36:15/0 = 0$
782	$37:15/0 = 0$
783	$38:15/0 = 0$
784	$39:15/0 = 0$
785	$40:15/0 = 0$
786	$41:15/0 = 0$
787	$42:15/0 = 0$
788	$43:15/0 = 0$
789	$44:15/0 = 0$
790	$45:15/0 = 0$
791	$46:15/0 = 0$
792	$47:15/0 = 0$
793	$48:15/0 = 0$
794	$0:16/0 = 0$
795	$1:16/0 = 0$
796	$2:16/0 = 0$
797	$3:16/0 = 0$
798	$4:16/0 = 0$
799	$5:16/0 = 0$
800	$6:16/0 = 0$
801	$7:16/0 = 0$
802	$8:16/0 = 0$
803	$9:16/0 = 0$
804	$10:16/0 = 0$
805	$11:16/0 = 0$
806	$12:16/0 = 0$
807	$13:16/0 = 0$
808	$14:16/0 = 0$
809	$15:16/0 = 0$
810	$16:16/0 = 0$
811	$17:16/0 = 0$
812	$18:16/0 = 0$

813 19:16/0 = 0
814 20:16/0 = 0
815 21:16/0 = 0
816 22:16/0 = 0
817 23:16/0 = 0
818 24:16/0 = 0
819 25:16/0 = 0
820 26:16/0 = 0
821 27:16/0 = 0
822 28:16/0 = 0
823 29:16/0 = 0
824 30:16/0 = 0
825 31:16/0 = 0
826 32:16/0 = 0
827 33:16/0 = 0
828 34:16/0 = 0
829 35:16/0 = 0
830 36:16/0 = 0
831 37:16/0 = 0
832 38:16/0 = 0
833 39:16/0 = 0
834 40:16/0 = 0
835 41:16/0 = 0
836 42:16/0 = 0
837 43:16/0 = 0
838 44:16/0 = 0
839 45:16/0 = 0
840 46:16/0 = 0
841 47:16/0 = 0
842 48:16/0 = 0
843 0:17/0 = 0
844 1:17/0 = 0
845 2:17/0 = 0

846 3:17/0 = 0
847 4:17/0 = 0
848 5:17/0 = 0
849 6:17/0 = 0
850 7:17/0 = 0
851 8:17/0 = 0
852 9:17/0 = 0
853 10:17/0 = 0
854 11:17/0 = 0
855 12:17/0 = 0
856 13:17/0 = 0
857 14:17/0 = 0
858 15:17/0 = 0
859 16:17/0 = 0
860 17:17/0 = 0
861 18:17/0 = 0
862 19:17/0 = 0
863 20:17/0 = 0
864 21:17/0 = 0
865 22:17/0 = 0
866 23:17/0 = 0
867 24:17/0 = 0
868 25:17/0 = 0
869 26:17/0 = 0
870 27:17/0 = 0
871 28:17/0 = 0
872 29:17/0 = 0
873 30:17/0 = 0
874 31:17/0 = 0
875 32:17/0 = 0
876 33:17/0 = 0
877 34:17/0 = 0
878 35:17/0 = 0

879 36:17/0 = 0
880 37:17/0 = 0
881 38:17/0 = 0
882 39:17/0 = 0
883 40:17/0 = 0
884 41:17/0 = 0
885 42:17/0 = 0
886 43:17/0 = 0
887 44:17/0 = 0
888 45:17/0 = 0
889 46:17/0 = 0
890 47:17/0 = 0
891 48:17/0 = 0
892 0:18/0 = 0
893 1:18/0 = 0
894 2:18/0 = 0
895 3:18/0 = 0
896 4:18/0 = 0
897 5:18/0 = 0
898 6:18/0 = 0
899 7:18/0 = 0
900 8:18/0 = 0
901 9:18/0 = 0
902 10:18/0 = 0
903 11:18/0 = 0
904 12:18/0 = 0
905 13:18/0 = 0
906 14:18/0 = 0
907 15:18/0 = 0
908 16:18/0 = 0
909 17:18/0 = 0
910 18:18/0 = 0
911 19:18/0 = 0

912 20:18/0 = 0
913 21:18/0 = 0
914 22:18/0 = 0
915 23:18/0 = 0
916 24:18/0 = 0
917 25:18/0 = 0
918 26:18/0 = 0
919 27:18/0 = 0
920 28:18/0 = 0
921 29:18/0 = 0
922 30:18/0 = 0
923 31:18/0 = 0
924 32:18/0 = 0
925 33:18/0 = 0
926 34:18/0 = 0
927 35:18/0 = 0
928 36:18/0 = 0
929 37:18/0 = 0
930 38:18/0 = 0
931 39:18/0 = 0
932 40:18/0 = 0
933 41:18/0 = 0
934 42:18/0 = 0
935 43:18/0 = 0
936 44:18/0 = 0
937 45:18/0 = 0
938 46:18/0 = 0
939 47:18/0 = 0
940 48:18/0 = 0
941 0:19/0 = 0
942 1:19/0 = 0
943 2:19/0 = 0
944 3:19/0 = 0

945 4:19/0 = 0
946 5:19/0 = 0
947 6:19/0 = 0
948 7:19/0 = 0
949 8:19/0 = 0
950 9:19/0 = 0
951 10:19/0 = 0
952 11:19/0 = 0
953 12:19/0 = 0
954 13:19/0 = 0
955 14:19/0 = 0
956 15:19/0 = 0
957 16:19/0 = 0
958 17:19/0 = 0
959 18:19/0 = 0
960 19:19/0 = 0
961 20:19/0 = 0
962 21:19/0 = 0
963 22:19/0 = 0
964 23:19/0 = 0
965 24:19/0 = 0
966 25:19/0 = 0
967 26:19/0 = 0
968 27:19/0 = 0
969 28:19/0 = 0
970 29:19/0 = 0
971 30:19/0 = 0
972 31:19/0 = 0
973 32:19/0 = 0
974 33:19/0 = 0
975 34:19/0 = 0
976 35:19/0 = 0
977 36:19/0 = 0

978 37:19/0 = 0
979 38:19/0 = 0
980 39:19/0 = 0
981 40:19/0 = 0
982 41:19/0 = 0
983 42:19/0 = 0
984 43:19/0 = 0
985 44:19/0 = 0
986 45:19/0 = 0
987 46:19/0 = 0
988 47:19/0 = 0
989 48:19/0 = 0
990 0:20/0 = 0
991 1:20/0 = 0
992 2:20/0 = 0
993 3:20/0 = 0
994 4:20/0 = 0
995 5:20/0 = 0
996 6:20/0 = 0
997 7:20/0 = 0
998 8:20/0 = 0
999 9:20/0 = 0
1000 10:20/0 = 0
1001 11:20/0 = 0
1002 12:20/0 = 0
1003 13:20/0 = 0
1004 14:20/0 = 0
1005 15:20/0 = 0
1006 16:20/0 = 0
1007 17:20/0 = 0
1008 18:20/0 = 0
1009 19:20/0 = 0
1010 20:20/0 = 0

1011	$21:20/0 = 0$
1012	$22:20/0 = 0$
1013	$23:20/0 = 0$
1014	$24:20/0 = 0$
1015	$25:20/0 = 0$
1016	$26:20/0 = 0$
1017	$27:20/0 = 0$
1018	$28:20/0 = 0$
1019	$29:20/0 = 0$
1020	$30:20/0 = 0$
1021	$31:20/0 = 0$
1022	$32:20/0 = 0$
1023	$33:20/0 = 0$
1024	$34:20/0 = 0$
1025	$35:20/0 = 0$
1026	$36:20/0 = 0$
1027	$37:20/0 = 0$
1028	$38:20/0 = 0$
1029	$39:20/0 = 0$
1030	$40:20/0 = 0$
1031	$41:20/0 = 0$
1032	$42:20/0 = 0$
1033	$43:20/0 = 0$
1034	$44:20/0 = 0$
1035	$45:20/0 = 0$
1036	$46:20/0 = 0$
1037	$47:20/0 = 0$
1038	$48:20/0 = 0$
1039	$0:21/0 = 0$
1040	$1:21/0 = 0$
1041	$2:21/0 = 0$
1042	$3:21/0 = 0$
1043	$4:21/0 = 0$

1044	$5:21/0 = 0$
1045	$6:21/0 = 0$
1046	$7:21/0 = 0$
1047	$8:21/0 = 0$
1048	$9:21/0 = 0$
1049	$10:21/0 = 0$
1050	$11:21/0 = 0$
1051	$12:21/0 = 0$
1052	$13:21/0 = 0$
1053	$14:21/0 = 0$
1054	$15:21/0 = 0$
1055	$16:21/0 = 0$
1056	$17:21/0 = 0$
1057	$18:21/0 = 0$
1058	$19:21/0 = 0$
1059	$20:21/0 = 0$
1060	$21:21/0 = 0$
1061	$22:21/0 = 0$
1062	$23:21/0 = 0$
1063	$24:21/0 = 0$
1064	$25:21/0 = 0$
1065	$26:21/0 = 0$
1066	$27:21/0 = 0$
1067	$28:21/0 = 0$
1068	$29:21/0 = 0$
1069	$30:21/0 = 0$
1070	$31:21/0 = 0$
1071	$32:21/0 = 0$
1072	$33:21/0 = 0$
1073	$34:21/0 = 0$
1074	$35:21/0 = 0$
1075	$36:21/0 = 0$
1076	$37:21/0 = 0$

```

1077 38:21/0 = 0
1078 39:21/0 = 0
1079 40:21/0 = 0
1080 41:21/0 = 0
1081 42:21/0 = 0
1082 43:21/0 = 0
1083 44:21/0 = 0
1084 45:21/0 = 0
1085 46:21/0 = 0
1086 47:21/0 = 0
1087 48:21/0 = 0
1088
1089 [sub_resource type="TileSet" id="TileSet_3drs5"]
1090 sources/0 = SubResource("TileSetAtlasSource_6h0bd")
1091
1092 [node name="TileMap" type="TileMap"]
1093 tile_set = SubResource("TileSet_3drs5")
1094 format = 2
1095 script = ExtResource("2_lf4lw")
1096
1097 [node name="AcceptDialog" parent="." instance=ExtResource("3_y08lj"
    )]
1098
1099 [node name="WinDialog" parent="." instance=ExtResource("4_fkys0")]

```

C.4.5 tile_map.gd

```

1 extends TileMap
2
3 const buildings: Array[Vector2i] = [
4     Vector2i(0, 19),
5     Vector2i(1, 19),

```

```

6      Vector2i(2, 19),
7      Vector2i(3, 19),
8      Vector2i(4, 19),
9      Vector2i(5, 19),
10     Vector2i(6, 19),
11     Vector2i(7, 19),
12     Vector2i(8, 20),
13     Vector2i(0, 20),
14     Vector2i(1, 20),
15     Vector2i(2, 20),
16     Vector2i(3, 20),
17     Vector2i(4, 20),
18     Vector2i(5, 20),
19     Vector2i(6, 20),
20     Vector2i(7, 20),
21     Vector2i(8, 20),
22     Vector2i(0, 21),
23     Vector2i(1, 21),
24     Vector2i(2, 21),
25     Vector2i(3, 21),
26     Vector2i(4, 21),
27     Vector2i(5, 21),
28     Vector2i(6, 21),
29     Vector2i(7, 21),
30     Vector2i(8, 21)
31 ]
32 const trees: Array[Vector2i] = [
33     Vector2i(0,1),
34     Vector2i(1,1),
35     Vector2i(2,1),
36     Vector2i(3,1),
37     Vector2i(4,1),
38     Vector2i(5,1),

```

```

39     Vector2i(6,1),
40     Vector2i(7,1),
41     Vector2i(0,2),
42     Vector2i(1,2),
43     Vector2i(2,2),
44     Vector2i(3,2),
45     Vector2i(4,2)
46 ]
47 const PLAYER_SPRITE: Vector2i = Vector2i(24, 7)
48 var player_placement_cell: Vector2i
49 const rings: Array[Vector2i] = [
50     Vector2i(43, 6),
51     Vector2i(44, 6),
52     Vector2i(45, 6),
53     Vector2i(46, 6)
54 ]
55 var ring_placement_cell: Vector2i
56
57 var points: Array[Dictionary] = []
58 const EUCLIDEAN: String = "Euclidean distance"
59 const MANHATTAN: String = "Manhattan distance"
60 @export_enum(EUCLIDEAN, MANHATTAN) var distance: String = MANHATTAN
61 @export_range(10, 40, 1) var random_starting_points: int = 20
62 var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
63 var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
64
65 # Called when the node enters the scene tree for the first time.
66 func _ready() -> void:
67     randomize()
68     var start_time: float = Time.get_ticks_msec()
69     define_points(random_starting_points)

```

```

70     paint_points()
71     place_player()
72     place_ring()
73     var new_time: float = Time.get_ticks_msec() - start_time
74     print("Time taken: " + str(new_time) + "ms")
75     $AcceptDialog.dialog_text = "You're a hollow Golem who seeks the
        ultimate treasure; a ring that's got something on top of it
        . It's somewhere in this large village and barely visible to
        your naked eyes, which took us " + str(new_time) + "
        milliseconds to generate (" + str(new_time / 1000.0) + "
        seconds), but you'll stop at nothing to get what you want.
        You can chow down every tree and fauna that stands in your
        way of the ring, but your Achilles heel is any bricks and
        mortar, which WILL make you stop at your tracks. Since it's
        easy to get lost in here, we'll tell you that you're in
        position " + str(player_placement_cell) + " in this big
        village of size " + str(Vector2i(x_tile_range, y_tile_range)
        ) + ". It's also easy to get stuck here, so either press the
        G key or right click to spawn somewhere else where there is
        fauna (or even the ring!!), because this game actually
        WANTS you to win it. Ultimately, though, it is YOUR job to
        find the ring, so are you ready to attain the treasure that
        is rightfully yours?!"
76     $AcceptDialog.visible = true
77     $AcceptDialog.confirmed.connect(_on_AcceptDialog_closed)
78     $AcceptDialog.canceled.connect(_on_AcceptDialog_closed)
79     $WinDialog.confirmed.connect(_on_WinDialog_confirmed)
80     $WinDialog.canceled.connect(_on_WinDialog_canceled)
81     get_tree().paused = true
82
83     func _on_WinDialog_confirmed() -> void:
84         get_tree().reload_current_scene()
85

```

```

86  func _on_WinDialog_canceled() -> void:
87      get_tree().quit()
88
89  func _on_AcceptDialog_closed() -> void:
90      $AcceptDialog.visible = false
91      get_tree().paused = false
92
93  func _get_random_placement_cell() -> Vector2i:
94      return Vector2i(randi() % x_tile_range, randi() % y_tile_range)
95
96  func place_player() -> void:
97      player_placement_cell = _get_random_placement_cell()
98      while buildings.has(get_cell_atlas_coords(0,
          player_placement_cell)) or player_placement_cell ==
          ring_placement_cell:
99          player_placement_cell = _get_random_placement_cell()
100      set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
101
102  func place_ring() -> void:
103      ring_placement_cell = _get_random_placement_cell()
104      while buildings.has(get_cell_atlas_coords(0, ring_placement_cell
          )) or ring_placement_cell == player_placement_cell:
105          ring_placement_cell = _get_random_placement_cell()
106      set_cell(0, ring_placement_cell, 0, rings.pick_random())
107
108  func _is_not_out_of_bounds(cell: Vector2i) -> bool:
109      return cell.x >= 0 and cell.x < x_tile_range and cell.y >= 0 and
          cell.y < y_tile_range
110
111  func _physics_process(_delta) -> void:
112      var previous_cell: Vector2i = player_placement_cell
113      var direction: Vector2i = Vector2i.ZERO
114      if Input.is_action_pressed("ui_up"): direction = Vector2i.UP

```



```

115     elif Input.is_action_pressed("ui_down"): direction = Vector2i.
        DOWN
116     elif Input.is_action_pressed("ui_left"): direction = Vector2i.
        LEFT
117     elif Input.is_action_pressed("ui_right"): direction = Vector2i.
        RIGHT
118     elif Input.is_action_just_pressed("reset_position"): # Respawn
        player in a different part of the map
119         player_placement_cell = _get_random_placement_cell()
120         while buildings.has(get_cell_atlas_coords(0,
            player_placement_cell)): # This time, since we're not
            STARTING the game, we don't care whether or not the
            player magically lands on the ring
121             player_placement_cell = _get_random_placement_cell()
122         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
123         set_cell(0, previous_cell, 0) # replace the previous sprite
124         return
125     var new_placement_cell: Vector2i = player_placement_cell +
        direction
126     if (not get_used_cells(0).has(new_placement_cell) or trees.has(
        get_cell_atlas_coords(0, new_placement_cell)) or
        new_placement_cell == ring_placement_cell) and
        _is_not_out_of_bounds(new_placement_cell):
127         player_placement_cell = new_placement_cell
128         set_cell(0, previous_cell, 0) # deletes contents of previous
            cell (atlas_coords = Vector2i(-1, -1))
129         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
130         if player_placement_cell == ring_placement_cell:
131             $WinDialog.visible = true
132             get_tree().paused = true
133
134     # ALGORITHM BEGINS HERE
135

```

```

136 func paint_points() -> void:
137     for point in points:
138         set_cell(0, Vector2(point["x"], point["y"]), 0, point["type"]
139             ])
140         for citizen in point["citizens"]:
141             if _is_in_bounds(point["x"], citizen["dx"], point["y"],
142                 citizen["dy"]):
143                 set_cell(0, Vector2(point["x"] + citizen["dx"], point["
144                     y"] + citizen["dy"]), 0, point["type"])
145
146 func _is_in_bounds(x: int, dx: int, y: int, dy: int) -> bool:
147     return x + dx >= 0 and x + dx < x_tile_range and y + dy >= 0 and
148         y + dy < y_tile_range
149
150 func _squared(x: int) -> int:
151     return x ** 2
152
153 func calculate_points_delta(x: int, y: int, p: int) -> float:
154     if distance == EUCLIDEAN:
155         return sqrt(_squared(points[p]["x"] - x) + _squared(points[p]
156             ]["y"] - y))
157     return abs(points[p]["x"] - x) + abs(points[p]["y"] - y)
158
159 func define_points(num_points: int) -> void:
160     var types: Array[Vector2i] = trees.duplicate()
161     types.append_array(buildings)
162     for i in range(num_points):
163         var x: int = randi_range(0, x_tile_range)
164         var y: int = randi_range(0, y_tile_range)
165         var type: Vector2i = types.pick_random()
166         types.erase(type)
167         points.append(
168             {

```

```

164         "type": type,
165         "x": x,
166         "y": y,
167         "citizens": []
168     }
169 )
170 for x in range(x_tile_range):
171     for y in range(y_tile_range):
172         var lowest_delta: Dictionary = {
173             "point_id": 0,
174             "delta": x_tile_range * y_tile_range
175         }
176         for p in range(len(points)):
177             var delta: float = calculate_points_delta(x, y, p)
178             if delta < lowest_delta["delta"]:
179                 lowest_delta = {
180                     "point_id": p,
181                     "delta": delta
182                 }
183             var active_point: Dictionary = points[lowest_delta["
184                 point_id"]]
185             var dx: int = x - active_point["x"]
186             var dy: int = y - active_point["y"]
187             active_point["citizens"].append(
188                 {
189                     "dx": dx,
190                     "dy": dy
191                 }

```

C.4.6 accept_dialog.tscn

```

1  [gd_scene format=3 uid="uid://cau5jgogdnf53"]
2
3  [node name="AcceptDialog" type="AcceptDialog"]
4  title = "Tree-Munching Time!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Bring it on!"
9  dialog_text = "You're a hollow Golem who seeks the ultimate
    treasure; a ring that's got something on top of it. It's
    somewhere in this large village and barely visible to your
    naked eyes, but you'll stop at nothing to get what you want.
    You can chow down every tree and fauna that stands in your way
    of the ring, but your Achilles heel is any bricks and mortar,
    which will make you stop at your tracks. Are you ready to
    attain your treasure?w Golem in a black-and-white world, in
    search for your most desired treasure. It's a ring with
    something on top of it. And you'll stop at nothing to get what
    you want. You can chow down every tree and fauna that stands in
    your way of the ring, but your Achilles heel is any bricks and
    mortar, which will make you stop at your tracks. Are you ready
    to attain the treasure that is rightfully yours?!"
10 dialog_autowrap = true

```

C.4.7 win_dialog.tscn

```

1  [gd_scene format=3 uid="uid://b5q8ovcigrvyr"]
2
3  [node name="WinDialog" type="ConfirmationDialog"]
4  title = "You Found the Treasure!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)

```

```

7  mouse_passthrough = true
8  ok_button_text = "Get Me a New Village"
9  dialog_text = "You found your treasure! Well done, you!"
10
11  Would you like to travel to a new village in the hopes of finding
    another ring? Or would you like to take your treasure home now?
    "
12  dialog_autowrap = true
13  cancel_button_text = "Get Me Out of Here"

```

C.4.8 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.4.9 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="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
21
22 [input]
```



```

23
24  reset_position={
25  "deadzone": 0.5,
26  "events": [Object(InputEventKey,"resource_local_to_scene":false,"
                resource_name":"","device":-1,"window_id":0,"alt_pressed":false,
                ,"shift_pressed":false,"ctrl_pressed":false,"meta_pressed":
                false,"pressed":false,"keycode":71,"physical_keycode":0,"
                key_label":0,"unicode":103,"echo":false,"script":null)
27  , Object(InputEventMouseButton,"resource_local_to_scene":false,"
                resource_name":"","device":-1,"window_id":0,"alt_pressed":false,
                ,"shift_pressed":false,"ctrl_pressed":false,"meta_pressed":
                false,"button_mask":2,"position":Vector2(75, 12),"
                global_position":Vector2(78, 44),"factor":1.0,"button_index
                ":2,"pressed":true,"double_click":false,"script":null)
28  ]
29  }
30
31  [rendering]
32
33  environment/defaults/default_clear_color=Color(0, 0, 0, 1)

```

C.5.4 tile_map.tscn

```

1  [gd_scene load_steps=7 format=3 uid="uid://d6lxxnr5bdh1w"]
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  [ext_resource type="PackedScene" path="res://accept_dialog.tscn" id
        ="3_y08lj"]
6  [ext_resource type="PackedScene" path="res://win_dialog.tscn" id="4
        _fkys0"]

```

```

7
8   [sub_resource type="TileSetAtlasSource" id="
      TileSetAtlasSource_6h0bd"]
9   texture = ExtResource("1_o183d")
10  0:0/0 = 0
11  1:0/0 = 0
12  2:0/0 = 0
13  3:0/0 = 0
14  4:0/0 = 0
15  5:0/0 = 0
16  6:0/0 = 0
17  7:0/0 = 0
18  8:0/0 = 0
19  9:0/0 = 0
20  10:0/0 = 0
21  11:0/0 = 0
22  12:0/0 = 0
23  13:0/0 = 0
24  14:0/0 = 0
25  15:0/0 = 0
26  16:0/0 = 0
27  17:0/0 = 0
28  18:0/0 = 0
29  19:0/0 = 0
30  20:0/0 = 0
31  21:0/0 = 0
32  22:0/0 = 0
33  23:0/0 = 0
34  24:0/0 = 0
35  25:0/0 = 0
36  26:0/0 = 0
37  27:0/0 = 0
38  28:0/0 = 0

```

39 29:0/0 = 0
40 30:0/0 = 0
41 31:0/0 = 0
42 32:0/0 = 0
43 33:0/0 = 0
44 34:0/0 = 0
45 35:0/0 = 0
46 36:0/0 = 0
47 37:0/0 = 0
48 38:0/0 = 0
49 39:0/0 = 0
50 40:0/0 = 0
51 41:0/0 = 0
52 42:0/0 = 0
53 43:0/0 = 0
54 44:0/0 = 0
55 45:0/0 = 0
56 46:0/0 = 0
57 47:0/0 = 0
58 48:0/0 = 0
59 0:1/0 = 0
60 1:1/0 = 0
61 2:1/0 = 0
62 3:1/0 = 0
63 4:1/0 = 0
64 5:1/0 = 0
65 6:1/0 = 0
66 7:1/0 = 0
67 8:1/0 = 0
68 9:1/0 = 0
69 10:1/0 = 0
70 11:1/0 = 0
71 12:1/0 = 0

72 13:1/0 = 0
73 14:1/0 = 0
74 15:1/0 = 0
75 16:1/0 = 0
76 17:1/0 = 0
77 18:1/0 = 0
78 19:1/0 = 0
79 20:1/0 = 0
80 21:1/0 = 0
81 22:1/0 = 0
82 23:1/0 = 0
83 24:1/0 = 0
84 25:1/0 = 0
85 26:1/0 = 0
86 27:1/0 = 0
87 28:1/0 = 0
88 29:1/0 = 0
89 30:1/0 = 0
90 31:1/0 = 0
91 32:1/0 = 0
92 33:1/0 = 0
93 34:1/0 = 0
94 35:1/0 = 0
95 36:1/0 = 0
96 37:1/0 = 0
97 38:1/0 = 0
98 39:1/0 = 0
99 40:1/0 = 0
100 41:1/0 = 0
101 42:1/0 = 0
102 43:1/0 = 0
103 44:1/0 = 0
104 45:1/0 = 0

105 46:1/0 = 0
106 47:1/0 = 0
107 48:1/0 = 0
108 0:2/0 = 0
109 1:2/0 = 0
110 2:2/0 = 0
111 3:2/0 = 0
112 4:2/0 = 0
113 5:2/0 = 0
114 6:2/0 = 0
115 7:2/0 = 0
116 8:2/0 = 0
117 9:2/0 = 0
118 10:2/0 = 0
119 11:2/0 = 0
120 12:2/0 = 0
121 13:2/0 = 0
122 14:2/0 = 0
123 15:2/0 = 0
124 16:2/0 = 0
125 17:2/0 = 0
126 18:2/0 = 0
127 19:2/0 = 0
128 20:2/0 = 0
129 21:2/0 = 0
130 22:2/0 = 0
131 23:2/0 = 0
132 24:2/0 = 0
133 25:2/0 = 0
134 26:2/0 = 0
135 27:2/0 = 0
136 28:2/0 = 0
137 29:2/0 = 0

138 30:2/0 = 0
139 31:2/0 = 0
140 32:2/0 = 0
141 33:2/0 = 0
142 34:2/0 = 0
143 35:2/0 = 0
144 36:2/0 = 0
145 37:2/0 = 0
146 38:2/0 = 0
147 39:2/0 = 0
148 40:2/0 = 0
149 41:2/0 = 0
150 42:2/0 = 0
151 43:2/0 = 0
152 44:2/0 = 0
153 45:2/0 = 0
154 46:2/0 = 0
155 47:2/0 = 0
156 48:2/0 = 0
157 0:3/0 = 0
158 1:3/0 = 0
159 2:3/0 = 0
160 3:3/0 = 0
161 4:3/0 = 0
162 5:3/0 = 0
163 6:3/0 = 0
164 7:3/0 = 0
165 8:3/0 = 0
166 9:3/0 = 0
167 10:3/0 = 0
168 11:3/0 = 0
169 12:3/0 = 0
170 13:3/0 = 0

171	$14:3/0 = 0$
172	$15:3/0 = 0$
173	$16:3/0 = 0$
174	$17:3/0 = 0$
175	$18:3/0 = 0$
176	$19:3/0 = 0$
177	$20:3/0 = 0$
178	$21:3/0 = 0$
179	$22:3/0 = 0$
180	$23:3/0 = 0$
181	$24:3/0 = 0$
182	$25:3/0 = 0$
183	$26:3/0 = 0$
184	$27:3/0 = 0$
185	$28:3/0 = 0$
186	$29:3/0 = 0$
187	$30:3/0 = 0$
188	$31:3/0 = 0$
189	$32:3/0 = 0$
190	$33:3/0 = 0$
191	$34:3/0 = 0$
192	$35:3/0 = 0$
193	$36:3/0 = 0$
194	$37:3/0 = 0$
195	$38:3/0 = 0$
196	$39:3/0 = 0$
197	$40:3/0 = 0$
198	$41:3/0 = 0$
199	$42:3/0 = 0$
200	$43:3/0 = 0$
201	$44:3/0 = 0$
202	$45:3/0 = 0$
203	$46:3/0 = 0$

204 $47:3/0 = 0$
205 $48:3/0 = 0$
206 $0:4/0 = 0$
207 $1:4/0 = 0$
208 $2:4/0 = 0$
209 $3:4/0 = 0$
210 $4:4/0 = 0$
211 $5:4/0 = 0$
212 $6:4/0 = 0$
213 $7:4/0 = 0$
214 $8:4/0 = 0$
215 $9:4/0 = 0$
216 $10:4/0 = 0$
217 $11:4/0 = 0$
218 $12:4/0 = 0$
219 $13:4/0 = 0$
220 $14:4/0 = 0$
221 $15:4/0 = 0$
222 $16:4/0 = 0$
223 $17:4/0 = 0$
224 $18:4/0 = 0$
225 $19:4/0 = 0$
226 $20:4/0 = 0$
227 $21:4/0 = 0$
228 $22:4/0 = 0$
229 $23:4/0 = 0$
230 $24:4/0 = 0$
231 $25:4/0 = 0$
232 $26:4/0 = 0$
233 $27:4/0 = 0$
234 $28:4/0 = 0$
235 $29:4/0 = 0$
236 $30:4/0 = 0$

237 31:4/0 = 0
 238 32:4/0 = 0
 239 33:4/0 = 0
 240 34:4/0 = 0
 241 35:4/0 = 0
 242 36:4/0 = 0
 243 37:4/0 = 0
 244 38:4/0 = 0
 245 39:4/0 = 0
 246 40:4/0 = 0
 247 41:4/0 = 0
 248 42:4/0 = 0
 249 43:4/0 = 0
 250 44:4/0 = 0
 251 45:4/0 = 0
 252 46:4/0 = 0
 253 47:4/0 = 0
 254 48:4/0 = 0
 255 0:5/0 = 0
 256 1:5/0 = 0
 257 2:5/0 = 0
 258 3:5/0 = 0
 259 4:5/0 = 0
 260 5:5/0 = 0
 261 6:5/0 = 0
 262 7:5/0 = 0
 263 8:5/0 = 0
 264 9:5/0 = 0
 265 10:5/0 = 0
 266 11:5/0 = 0
 267 12:5/0 = 0
 268 13:5/0 = 0
 269 14:5/0 = 0

270	$15:5/0 = 0$
271	$16:5/0 = 0$
272	$17:5/0 = 0$
273	$18:5/0 = 0$
274	$19:5/0 = 0$
275	$20:5/0 = 0$
276	$21:5/0 = 0$
277	$22:5/0 = 0$
278	$23:5/0 = 0$
279	$24:5/0 = 0$
280	$25:5/0 = 0$
281	$26:5/0 = 0$
282	$27:5/0 = 0$
283	$28:5/0 = 0$
284	$29:5/0 = 0$
285	$30:5/0 = 0$
286	$31:5/0 = 0$
287	$32:5/0 = 0$
288	$33:5/0 = 0$
289	$34:5/0 = 0$
290	$35:5/0 = 0$
291	$36:5/0 = 0$
292	$37:5/0 = 0$
293	$38:5/0 = 0$
294	$39:5/0 = 0$
295	$40:5/0 = 0$
296	$41:5/0 = 0$
297	$42:5/0 = 0$
298	$43:5/0 = 0$
299	$44:5/0 = 0$
300	$45:5/0 = 0$
301	$46:5/0 = 0$
302	$47:5/0 = 0$

303 $48:5/0 = 0$
304 $0:6/0 = 0$
305 $1:6/0 = 0$
306 $2:6/0 = 0$
307 $3:6/0 = 0$
308 $4:6/0 = 0$
309 $5:6/0 = 0$
310 $6:6/0 = 0$
311 $7:6/0 = 0$
312 $8:6/0 = 0$
313 $9:6/0 = 0$
314 $10:6/0 = 0$
315 $11:6/0 = 0$
316 $12:6/0 = 0$
317 $13:6/0 = 0$
318 $14:6/0 = 0$
319 $15:6/0 = 0$
320 $16:6/0 = 0$
321 $17:6/0 = 0$
322 $18:6/0 = 0$
323 $19:6/0 = 0$
324 $20:6/0 = 0$
325 $21:6/0 = 0$
326 $22:6/0 = 0$
327 $23:6/0 = 0$
328 $24:6/0 = 0$
329 $25:6/0 = 0$
330 $26:6/0 = 0$
331 $27:6/0 = 0$
332 $28:6/0 = 0$
333 $29:6/0 = 0$
334 $30:6/0 = 0$
335 $31:6/0 = 0$

336 32:6/0 = 0
337 33:6/0 = 0
338 34:6/0 = 0
339 35:6/0 = 0
340 36:6/0 = 0
341 37:6/0 = 0
342 38:6/0 = 0
343 39:6/0 = 0
344 40:6/0 = 0
345 41:6/0 = 0
346 42:6/0 = 0
347 43:6/0 = 0
348 44:6/0 = 0
349 45:6/0 = 0
350 46:6/0 = 0
351 47:6/0 = 0
352 48:6/0 = 0
353 0:7/0 = 0
354 1:7/0 = 0
355 2:7/0 = 0
356 3:7/0 = 0
357 4:7/0 = 0
358 5:7/0 = 0
359 6:7/0 = 0
360 7:7/0 = 0
361 8:7/0 = 0
362 9:7/0 = 0
363 10:7/0 = 0
364 11:7/0 = 0
365 12:7/0 = 0
366 13:7/0 = 0
367 14:7/0 = 0
368 15:7/0 = 0

369 16:7/0 = 0
370 17:7/0 = 0
371 18:7/0 = 0
372 19:7/0 = 0
373 20:7/0 = 0
374 21:7/0 = 0
375 22:7/0 = 0
376 23:7/0 = 0
377 24:7/0 = 0
378 25:7/0 = 0
379 26:7/0 = 0
380 27:7/0 = 0
381 28:7/0 = 0
382 29:7/0 = 0
383 30:7/0 = 0
384 31:7/0 = 0
385 32:7/0 = 0
386 33:7/0 = 0
387 34:7/0 = 0
388 35:7/0 = 0
389 36:7/0 = 0
390 37:7/0 = 0
391 38:7/0 = 0
392 39:7/0 = 0
393 40:7/0 = 0
394 41:7/0 = 0
395 42:7/0 = 0
396 43:7/0 = 0
397 44:7/0 = 0
398 45:7/0 = 0
399 46:7/0 = 0
400 47:7/0 = 0
401 48:7/0 = 0

402 $0:8/0 = 0$
403 $1:8/0 = 0$
404 $2:8/0 = 0$
405 $3:8/0 = 0$
406 $4:8/0 = 0$
407 $5:8/0 = 0$
408 $6:8/0 = 0$
409 $7:8/0 = 0$
410 $8:8/0 = 0$
411 $9:8/0 = 0$
412 $10:8/0 = 0$
413 $11:8/0 = 0$
414 $12:8/0 = 0$
415 $13:8/0 = 0$
416 $14:8/0 = 0$
417 $15:8/0 = 0$
418 $16:8/0 = 0$
419 $17:8/0 = 0$
420 $18:8/0 = 0$
421 $19:8/0 = 0$
422 $20:8/0 = 0$
423 $21:8/0 = 0$
424 $22:8/0 = 0$
425 $23:8/0 = 0$
426 $24:8/0 = 0$
427 $25:8/0 = 0$
428 $26:8/0 = 0$
429 $27:8/0 = 0$
430 $28:8/0 = 0$
431 $29:8/0 = 0$
432 $30:8/0 = 0$
433 $31:8/0 = 0$
434 $32:8/0 = 0$

435 $33:8/0 = 0$
436 $34:8/0 = 0$
437 $35:8/0 = 0$
438 $36:8/0 = 0$
439 $37:8/0 = 0$
440 $38:8/0 = 0$
441 $39:8/0 = 0$
442 $40:8/0 = 0$
443 $41:8/0 = 0$
444 $42:8/0 = 0$
445 $43:8/0 = 0$
446 $44:8/0 = 0$
447 $45:8/0 = 0$
448 $46:8/0 = 0$
449 $47:8/0 = 0$
450 $48:8/0 = 0$
451 $0:9/0 = 0$
452 $1:9/0 = 0$
453 $2:9/0 = 0$
454 $3:9/0 = 0$
455 $4:9/0 = 0$
456 $5:9/0 = 0$
457 $6:9/0 = 0$
458 $7:9/0 = 0$
459 $8:9/0 = 0$
460 $9:9/0 = 0$
461 $10:9/0 = 0$
462 $11:9/0 = 0$
463 $12:9/0 = 0$
464 $13:9/0 = 0$
465 $14:9/0 = 0$
466 $15:9/0 = 0$
467 $16:9/0 = 0$

468 17:9/0 = 0
469 18:9/0 = 0
470 19:9/0 = 0
471 20:9/0 = 0
472 21:9/0 = 0
473 22:9/0 = 0
474 23:9/0 = 0
475 24:9/0 = 0
476 25:9/0 = 0
477 26:9/0 = 0
478 27:9/0 = 0
479 28:9/0 = 0
480 29:9/0 = 0
481 30:9/0 = 0
482 31:9/0 = 0
483 32:9/0 = 0
484 33:9/0 = 0
485 34:9/0 = 0
486 35:9/0 = 0
487 36:9/0 = 0
488 37:9/0 = 0
489 38:9/0 = 0
490 39:9/0 = 0
491 40:9/0 = 0
492 41:9/0 = 0
493 42:9/0 = 0
494 43:9/0 = 0
495 44:9/0 = 0
496 45:9/0 = 0
497 46:9/0 = 0
498 47:9/0 = 0
499 48:9/0 = 0
500 0:10/0 = 0

501 1:10/0 = 0
502 2:10/0 = 0
503 3:10/0 = 0
504 4:10/0 = 0
505 5:10/0 = 0
506 6:10/0 = 0
507 7:10/0 = 0
508 8:10/0 = 0
509 9:10/0 = 0
510 10:10/0 = 0
511 11:10/0 = 0
512 12:10/0 = 0
513 13:10/0 = 0
514 14:10/0 = 0
515 15:10/0 = 0
516 16:10/0 = 0
517 17:10/0 = 0
518 18:10/0 = 0
519 19:10/0 = 0
520 20:10/0 = 0
521 21:10/0 = 0
522 22:10/0 = 0
523 23:10/0 = 0
524 24:10/0 = 0
525 25:10/0 = 0
526 26:10/0 = 0
527 27:10/0 = 0
528 28:10/0 = 0
529 29:10/0 = 0
530 30:10/0 = 0
531 31:10/0 = 0
532 32:10/0 = 0
533 33:10/0 = 0

534 34:10/0 = 0
535 35:10/0 = 0
536 36:10/0 = 0
537 37:10/0 = 0
538 38:10/0 = 0
539 39:10/0 = 0
540 40:10/0 = 0
541 41:10/0 = 0
542 42:10/0 = 0
543 43:10/0 = 0
544 44:10/0 = 0
545 45:10/0 = 0
546 46:10/0 = 0
547 47:10/0 = 0
548 48:10/0 = 0
549 0:11/0 = 0
550 1:11/0 = 0
551 2:11/0 = 0
552 3:11/0 = 0
553 4:11/0 = 0
554 5:11/0 = 0
555 6:11/0 = 0
556 7:11/0 = 0
557 8:11/0 = 0
558 9:11/0 = 0
559 10:11/0 = 0
560 11:11/0 = 0
561 12:11/0 = 0
562 13:11/0 = 0
563 14:11/0 = 0
564 15:11/0 = 0
565 16:11/0 = 0
566 17:11/0 = 0

567 18:11/0 = 0
568 19:11/0 = 0
569 20:11/0 = 0
570 21:11/0 = 0
571 22:11/0 = 0
572 23:11/0 = 0
573 24:11/0 = 0
574 25:11/0 = 0
575 26:11/0 = 0
576 27:11/0 = 0
577 28:11/0 = 0
578 29:11/0 = 0
579 30:11/0 = 0
580 31:11/0 = 0
581 32:11/0 = 0
582 33:11/0 = 0
583 34:11/0 = 0
584 35:11/0 = 0
585 36:11/0 = 0
586 37:11/0 = 0
587 38:11/0 = 0
588 39:11/0 = 0
589 40:11/0 = 0
590 41:11/0 = 0
591 42:11/0 = 0
592 43:11/0 = 0
593 44:11/0 = 0
594 45:11/0 = 0
595 46:11/0 = 0
596 47:11/0 = 0
597 48:11/0 = 0
598 0:12/0 = 0
599 1:12/0 = 0

600 2:12/0 = 0
601 3:12/0 = 0
602 4:12/0 = 0
603 5:12/0 = 0
604 6:12/0 = 0
605 7:12/0 = 0
606 8:12/0 = 0
607 9:12/0 = 0
608 10:12/0 = 0
609 11:12/0 = 0
610 12:12/0 = 0
611 13:12/0 = 0
612 14:12/0 = 0
613 15:12/0 = 0
614 16:12/0 = 0
615 17:12/0 = 0
616 18:12/0 = 0
617 19:12/0 = 0
618 20:12/0 = 0
619 21:12/0 = 0
620 22:12/0 = 0
621 23:12/0 = 0
622 24:12/0 = 0
623 25:12/0 = 0
624 26:12/0 = 0
625 27:12/0 = 0
626 28:12/0 = 0
627 29:12/0 = 0
628 30:12/0 = 0
629 31:12/0 = 0
630 32:12/0 = 0
631 33:12/0 = 0
632 34:12/0 = 0

633 35:12/0 = 0
634 36:12/0 = 0
635 37:12/0 = 0
636 38:12/0 = 0
637 39:12/0 = 0
638 40:12/0 = 0
639 41:12/0 = 0
640 42:12/0 = 0
641 43:12/0 = 0
642 44:12/0 = 0
643 45:12/0 = 0
644 46:12/0 = 0
645 47:12/0 = 0
646 48:12/0 = 0
647 0:13/0 = 0
648 1:13/0 = 0
649 2:13/0 = 0
650 3:13/0 = 0
651 4:13/0 = 0
652 5:13/0 = 0
653 6:13/0 = 0
654 7:13/0 = 0
655 8:13/0 = 0
656 9:13/0 = 0
657 10:13/0 = 0
658 11:13/0 = 0
659 12:13/0 = 0
660 13:13/0 = 0
661 14:13/0 = 0
662 15:13/0 = 0
663 16:13/0 = 0
664 17:13/0 = 0
665 18:13/0 = 0

666 19:13/0 = 0
667 20:13/0 = 0
668 21:13/0 = 0
669 22:13/0 = 0
670 23:13/0 = 0
671 24:13/0 = 0
672 25:13/0 = 0
673 26:13/0 = 0
674 27:13/0 = 0
675 28:13/0 = 0
676 29:13/0 = 0
677 30:13/0 = 0
678 31:13/0 = 0
679 32:13/0 = 0
680 33:13/0 = 0
681 34:13/0 = 0
682 35:13/0 = 0
683 36:13/0 = 0
684 37:13/0 = 0
685 38:13/0 = 0
686 39:13/0 = 0
687 40:13/0 = 0
688 41:13/0 = 0
689 42:13/0 = 0
690 43:13/0 = 0
691 44:13/0 = 0
692 45:13/0 = 0
693 46:13/0 = 0
694 47:13/0 = 0
695 48:13/0 = 0
696 0:14/0 = 0
697 1:14/0 = 0
698 2:14/0 = 0

699	$3:14/0 = 0$
700	$4:14/0 = 0$
701	$5:14/0 = 0$
702	$6:14/0 = 0$
703	$7:14/0 = 0$
704	$8:14/0 = 0$
705	$9:14/0 = 0$
706	$10:14/0 = 0$
707	$11:14/0 = 0$
708	$12:14/0 = 0$
709	$13:14/0 = 0$
710	$14:14/0 = 0$
711	$15:14/0 = 0$
712	$16:14/0 = 0$
713	$17:14/0 = 0$
714	$18:14/0 = 0$
715	$19:14/0 = 0$
716	$20:14/0 = 0$
717	$21:14/0 = 0$
718	$22:14/0 = 0$
719	$23:14/0 = 0$
720	$24:14/0 = 0$
721	$25:14/0 = 0$
722	$26:14/0 = 0$
723	$27:14/0 = 0$
724	$28:14/0 = 0$
725	$29:14/0 = 0$
726	$30:14/0 = 0$
727	$31:14/0 = 0$
728	$32:14/0 = 0$
729	$33:14/0 = 0$
730	$34:14/0 = 0$
731	$35:14/0 = 0$

732	$36:14/0 = 0$
733	$37:14/0 = 0$
734	$38:14/0 = 0$
735	$39:14/0 = 0$
736	$40:14/0 = 0$
737	$41:14/0 = 0$
738	$42:14/0 = 0$
739	$43:14/0 = 0$
740	$44:14/0 = 0$
741	$45:14/0 = 0$
742	$46:14/0 = 0$
743	$47:14/0 = 0$
744	$48:14/0 = 0$
745	$0:15/0 = 0$
746	$1:15/0 = 0$
747	$2:15/0 = 0$
748	$3:15/0 = 0$
749	$4:15/0 = 0$
750	$5:15/0 = 0$
751	$6:15/0 = 0$
752	$7:15/0 = 0$
753	$8:15/0 = 0$
754	$9:15/0 = 0$
755	$10:15/0 = 0$
756	$11:15/0 = 0$
757	$12:15/0 = 0$
758	$13:15/0 = 0$
759	$14:15/0 = 0$
760	$15:15/0 = 0$
761	$16:15/0 = 0$
762	$17:15/0 = 0$
763	$18:15/0 = 0$
764	$19:15/0 = 0$

765 20:15/0 = 0
766 21:15/0 = 0
767 22:15/0 = 0
768 23:15/0 = 0
769 24:15/0 = 0
770 25:15/0 = 0
771 26:15/0 = 0
772 27:15/0 = 0
773 28:15/0 = 0
774 29:15/0 = 0
775 30:15/0 = 0
776 31:15/0 = 0
777 32:15/0 = 0
778 33:15/0 = 0
779 34:15/0 = 0
780 35:15/0 = 0
781 36:15/0 = 0
782 37:15/0 = 0
783 38:15/0 = 0
784 39:15/0 = 0
785 40:15/0 = 0
786 41:15/0 = 0
787 42:15/0 = 0
788 43:15/0 = 0
789 44:15/0 = 0
790 45:15/0 = 0
791 46:15/0 = 0
792 47:15/0 = 0
793 48:15/0 = 0
794 0:16/0 = 0
795 1:16/0 = 0
796 2:16/0 = 0
797 3:16/0 = 0

798	$4:16/0 = 0$
799	$5:16/0 = 0$
800	$6:16/0 = 0$
801	$7:16/0 = 0$
802	$8:16/0 = 0$
803	$9:16/0 = 0$
804	$10:16/0 = 0$
805	$11:16/0 = 0$
806	$12:16/0 = 0$
807	$13:16/0 = 0$
808	$14:16/0 = 0$
809	$15:16/0 = 0$
810	$16:16/0 = 0$
811	$17:16/0 = 0$
812	$18:16/0 = 0$
813	$19:16/0 = 0$
814	$20:16/0 = 0$
815	$21:16/0 = 0$
816	$22:16/0 = 0$
817	$23:16/0 = 0$
818	$24:16/0 = 0$
819	$25:16/0 = 0$
820	$26:16/0 = 0$
821	$27:16/0 = 0$
822	$28:16/0 = 0$
823	$29:16/0 = 0$
824	$30:16/0 = 0$
825	$31:16/0 = 0$
826	$32:16/0 = 0$
827	$33:16/0 = 0$
828	$34:16/0 = 0$
829	$35:16/0 = 0$
830	$36:16/0 = 0$

831 37:16/0 = 0
832 38:16/0 = 0
833 39:16/0 = 0
834 40:16/0 = 0
835 41:16/0 = 0
836 42:16/0 = 0
837 43:16/0 = 0
838 44:16/0 = 0
839 45:16/0 = 0
840 46:16/0 = 0
841 47:16/0 = 0
842 48:16/0 = 0
843 0:17/0 = 0
844 1:17/0 = 0
845 2:17/0 = 0
846 3:17/0 = 0
847 4:17/0 = 0
848 5:17/0 = 0
849 6:17/0 = 0
850 7:17/0 = 0
851 8:17/0 = 0
852 9:17/0 = 0
853 10:17/0 = 0
854 11:17/0 = 0
855 12:17/0 = 0
856 13:17/0 = 0
857 14:17/0 = 0
858 15:17/0 = 0
859 16:17/0 = 0
860 17:17/0 = 0
861 18:17/0 = 0
862 19:17/0 = 0
863 20:17/0 = 0

864 21:17/0 = 0
865 22:17/0 = 0
866 23:17/0 = 0
867 24:17/0 = 0
868 25:17/0 = 0
869 26:17/0 = 0
870 27:17/0 = 0
871 28:17/0 = 0
872 29:17/0 = 0
873 30:17/0 = 0
874 31:17/0 = 0
875 32:17/0 = 0
876 33:17/0 = 0
877 34:17/0 = 0
878 35:17/0 = 0
879 36:17/0 = 0
880 37:17/0 = 0
881 38:17/0 = 0
882 39:17/0 = 0
883 40:17/0 = 0
884 41:17/0 = 0
885 42:17/0 = 0
886 43:17/0 = 0
887 44:17/0 = 0
888 45:17/0 = 0
889 46:17/0 = 0
890 47:17/0 = 0
891 48:17/0 = 0
892 0:18/0 = 0
893 1:18/0 = 0
894 2:18/0 = 0
895 3:18/0 = 0
896 4:18/0 = 0

897 5:18/0 = 0
898 6:18/0 = 0
899 7:18/0 = 0
900 8:18/0 = 0
901 9:18/0 = 0
902 10:18/0 = 0
903 11:18/0 = 0
904 12:18/0 = 0
905 13:18/0 = 0
906 14:18/0 = 0
907 15:18/0 = 0
908 16:18/0 = 0
909 17:18/0 = 0
910 18:18/0 = 0
911 19:18/0 = 0
912 20:18/0 = 0
913 21:18/0 = 0
914 22:18/0 = 0
915 23:18/0 = 0
916 24:18/0 = 0
917 25:18/0 = 0
918 26:18/0 = 0
919 27:18/0 = 0
920 28:18/0 = 0
921 29:18/0 = 0
922 30:18/0 = 0
923 31:18/0 = 0
924 32:18/0 = 0
925 33:18/0 = 0
926 34:18/0 = 0
927 35:18/0 = 0
928 36:18/0 = 0
929 37:18/0 = 0

930 38:18/0 = 0
931 39:18/0 = 0
932 40:18/0 = 0
933 41:18/0 = 0
934 42:18/0 = 0
935 43:18/0 = 0
936 44:18/0 = 0
937 45:18/0 = 0
938 46:18/0 = 0
939 47:18/0 = 0
940 48:18/0 = 0
941 0:19/0 = 0
942 1:19/0 = 0
943 2:19/0 = 0
944 3:19/0 = 0
945 4:19/0 = 0
946 5:19/0 = 0
947 6:19/0 = 0
948 7:19/0 = 0
949 8:19/0 = 0
950 9:19/0 = 0
951 10:19/0 = 0
952 11:19/0 = 0
953 12:19/0 = 0
954 13:19/0 = 0
955 14:19/0 = 0
956 15:19/0 = 0
957 16:19/0 = 0
958 17:19/0 = 0
959 18:19/0 = 0
960 19:19/0 = 0
961 20:19/0 = 0
962 21:19/0 = 0

963 22:19/0 = 0
964 23:19/0 = 0
965 24:19/0 = 0
966 25:19/0 = 0
967 26:19/0 = 0
968 27:19/0 = 0
969 28:19/0 = 0
970 29:19/0 = 0
971 30:19/0 = 0
972 31:19/0 = 0
973 32:19/0 = 0
974 33:19/0 = 0
975 34:19/0 = 0
976 35:19/0 = 0
977 36:19/0 = 0
978 37:19/0 = 0
979 38:19/0 = 0
980 39:19/0 = 0
981 40:19/0 = 0
982 41:19/0 = 0
983 42:19/0 = 0
984 43:19/0 = 0
985 44:19/0 = 0
986 45:19/0 = 0
987 46:19/0 = 0
988 47:19/0 = 0
989 48:19/0 = 0
990 0:20/0 = 0
991 1:20/0 = 0
992 2:20/0 = 0
993 3:20/0 = 0
994 4:20/0 = 0
995 5:20/0 = 0

996	$6:20/0 = 0$
997	$7:20/0 = 0$
998	$8:20/0 = 0$
999	$9:20/0 = 0$
1000	$10:20/0 = 0$
1001	$11:20/0 = 0$
1002	$12:20/0 = 0$
1003	$13:20/0 = 0$
1004	$14:20/0 = 0$
1005	$15:20/0 = 0$
1006	$16:20/0 = 0$
1007	$17:20/0 = 0$
1008	$18:20/0 = 0$
1009	$19:20/0 = 0$
1010	$20:20/0 = 0$
1011	$21:20/0 = 0$
1012	$22:20/0 = 0$
1013	$23:20/0 = 0$
1014	$24:20/0 = 0$
1015	$25:20/0 = 0$
1016	$26:20/0 = 0$
1017	$27:20/0 = 0$
1018	$28:20/0 = 0$
1019	$29:20/0 = 0$
1020	$30:20/0 = 0$
1021	$31:20/0 = 0$
1022	$32:20/0 = 0$
1023	$33:20/0 = 0$
1024	$34:20/0 = 0$
1025	$35:20/0 = 0$
1026	$36:20/0 = 0$
1027	$37:20/0 = 0$
1028	$38:20/0 = 0$

1029	$39:20/0 = 0$
1030	$40:20/0 = 0$
1031	$41:20/0 = 0$
1032	$42:20/0 = 0$
1033	$43:20/0 = 0$
1034	$44:20/0 = 0$
1035	$45:20/0 = 0$
1036	$46:20/0 = 0$
1037	$47:20/0 = 0$
1038	$48:20/0 = 0$
1039	$0:21/0 = 0$
1040	$1:21/0 = 0$
1041	$2:21/0 = 0$
1042	$3:21/0 = 0$
1043	$4:21/0 = 0$
1044	$5:21/0 = 0$
1045	$6:21/0 = 0$
1046	$7:21/0 = 0$
1047	$8:21/0 = 0$
1048	$9:21/0 = 0$
1049	$10:21/0 = 0$
1050	$11:21/0 = 0$
1051	$12:21/0 = 0$
1052	$13:21/0 = 0$
1053	$14:21/0 = 0$
1054	$15:21/0 = 0$
1055	$16:21/0 = 0$
1056	$17:21/0 = 0$
1057	$18:21/0 = 0$
1058	$19:21/0 = 0$
1059	$20:21/0 = 0$
1060	$21:21/0 = 0$
1061	$22:21/0 = 0$

```

1062    23:21/0 = 0
1063    24:21/0 = 0
1064    25:21/0 = 0
1065    26:21/0 = 0
1066    27:21/0 = 0
1067    28:21/0 = 0
1068    29:21/0 = 0
1069    30:21/0 = 0
1070    31:21/0 = 0
1071    32:21/0 = 0
1072    33:21/0 = 0
1073    34:21/0 = 0
1074    35:21/0 = 0
1075    36:21/0 = 0
1076    37:21/0 = 0
1077    38:21/0 = 0
1078    39:21/0 = 0
1079    40:21/0 = 0
1080    41:21/0 = 0
1081    42:21/0 = 0
1082    43:21/0 = 0
1083    44:21/0 = 0
1084    45:21/0 = 0
1085    46:21/0 = 0
1086    47:21/0 = 0
1087    48:21/0 = 0
1088
1089    [sub_resource type="TileSet" id="TileSet_3drs5"]
1090    sources/0 = SubResource("TileSetAtlasSource_6h0bd")
1091
1092    [node name="TileMap" type="TileMap"]
1093    tile_set = SubResource("TileSet_3drs5")
1094    format = 2

```

```

1095  script = ExtResource("2_lf4lw")
1096
1097  [node name="AcceptDialog" parent="." instance=ExtResource("3_y08lj"
        )]
1098
1099  [node name="WinDialog" parent="." instance=ExtResource("4_fkys0")]

```

C.5.5 tile_map.gd

```

1  extends TileMap
2
3  const buildings: Array[Vector2i] = [
4      Vector2i(0, 19),
5      Vector2i(1, 19),
6      Vector2i(2, 19),
7      Vector2i(3, 19),
8      Vector2i(4, 19),
9      Vector2i(5, 19),
10     Vector2i(6, 19),
11     Vector2i(7, 19),
12     Vector2i(8, 20),
13     Vector2i(0, 20),
14     Vector2i(1, 20),
15     Vector2i(2, 20),
16     Vector2i(3, 20),
17     Vector2i(4, 20),
18     Vector2i(5, 20),
19     Vector2i(6, 20),
20     Vector2i(7, 20),
21     Vector2i(8, 20),
22     Vector2i(0, 21),
23     Vector2i(1, 21),

```

```

24     Vector2i(2, 21),
25     Vector2i(3, 21),
26     Vector2i(4, 21),
27     Vector2i(5, 21),
28     Vector2i(6, 21),
29     Vector2i(7, 21),
30     Vector2i(8, 21)
31 ]
32 const trees: Array[Vector2i] = [
33     Vector2i(0,1),
34     Vector2i(1,1),
35     Vector2i(2,1),
36     Vector2i(3,1),
37     Vector2i(4,1),
38     Vector2i(5,1),
39     Vector2i(6,1),
40     Vector2i(7,1),
41     Vector2i(0,2),
42     Vector2i(1,2),
43     Vector2i(2,2),
44     Vector2i(3,2),
45     Vector2i(4,2)
46 ]
47 const PLAYER_SPRITE: Vector2i = Vector2i(24, 7)
48 var player_placement_cell: Vector2i
49 const rings: Array[Vector2i] = [
50     Vector2i(43, 6),
51     Vector2i(44, 6),
52     Vector2i(45, 6),
53     Vector2i(46, 6)
54 ]
55 var ring_placement_cell: Vector2i
56

```

```

57  var points: Array[Dictionary] = []
58  const EUCLIDEAN: String = "Euclidean distance"
59  const MANHATTAN: String = "Manhattan distance"
60  @export_enum(EUCLIDEAN, MANHATTAN) var distance: String = MANHATTAN
61  @export_range(10, 40, 1) var random_starting_points: int = 20
62  var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
63  var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
64
65  # Called when the node enters the scene tree for the first time.
66  func _ready() -> void:
67      randomize()
68      var start_time: float = Time.get_ticks_msec()
69      define_points(random_starting_points)
70      paint_points()
71      place_player()
72      place_ring()
73      var new_time: float = Time.get_ticks_msec() - start_time
74      print("Time taken: " + str(new_time) + "ms")
75      $AcceptDialog.dialog_text = "You're a hollow Golem who seeks the
        ultimate treasure; a ring that's got something on top of it
        . It's somewhere in this large village and barely visible to
        your naked eyes, which took us " + str(new_time) + "
        milliseconds to generate (" + str(new_time / 1000.0) + "
        seconds), but you'll stop at nothing to get what you want.
        You can chow down every tree and fauna that stands in your
        way of the ring, but your Achilles heel is any bricks and
        mortar, which WILL make you stop at your tracks. Since it's
        easy to get lost in here, we'll tell you that you're in
        position " + str(player_placement_cell) + " in this big
        village of size " + str(Vector2i(x_tile_range, y_tile_range)
        ) + ". It's also easy to get stuck here, so either press the

```

```

    G key or right click to spawn somewhere else where there is
    fauna (or even the ring!!), because this game actually
    WANTS you to win it. Ultimately, though, it is YOUR job to
    find the ring, so are you ready to attain the treasure that
    is rightfully yours?!"

76     $AcceptDialog.visible = true
77     $AcceptDialog.confirmed.connect(_on_AcceptDialog_closed)
78     $AcceptDialog.canceled.connect(_on_AcceptDialog_closed)
79     $WinDialog.confirmed.connect(_on_WinDialog_confirmed)
80     $WinDialog.canceled.connect(_on_WinDialog_canceled)
81     get_tree().paused = true
82
83     func _on_WinDialog_confirmed() -> void:
84         get_tree().reload_current_scene()
85
86     func _on_WinDialog_canceled() -> void:
87         get_tree().quit()
88
89     func _on_AcceptDialog_closed() -> void:
90         $AcceptDialog.visible = false
91         get_tree().paused = false
92
93     func _get_random_placement_cell() -> Vector2i:
94         return Vector2i(randi() % x_tile_range, randi() % y_tile_range)
95
96     func place_player() -> void:
97         player_placement_cell = _get_random_placement_cell()
98         while buildings.has(get_cell_atlas_coords(0,
99             player_placement_cell)) or player_placement_cell ==
100             ring_placement_cell:
101             player_placement_cell = _get_random_placement_cell()
102         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)

```

```

102 func place_ring() -> void:
103     ring_placement_cell = _get_random_placement_cell()
104     while buildings.has(get_cell_atlas_coords(0, ring_placement_cell
        )) or ring_placement_cell == player_placement_cell:
105         ring_placement_cell = _get_random_placement_cell()
106     set_cell(0, ring_placement_cell, 0, rings.pick_random())
107
108 func _is_not_out_of_bounds(cell: Vector2i) -> bool:
109     return cell.x >= 0 and cell.x < x_tile_range and cell.y >= 0 and
        cell.y < y_tile_range
110
111 func _physics_process(_delta) -> void:
112     var previous_cell: Vector2i = player_placement_cell
113     var direction: Vector2i = Vector2i.ZERO
114     if Input.is_action_pressed("ui_up"): direction = Vector2i.UP
115     elif Input.is_action_pressed("ui_down"): direction = Vector2i.
        DOWN
116     elif Input.is_action_pressed("ui_left"): direction = Vector2i.
        LEFT
117     elif Input.is_action_pressed("ui_right"): direction = Vector2i.
        RIGHT
118     elif Input.is_action_just_pressed("reset_position"): # Respawn
        player in a different part of the map
119     player_placement_cell = _get_random_placement_cell()
120     while buildings.has(get_cell_atlas_coords(0,
        player_placement_cell)): # This time, since we're not
        STARTING the game, we don't care whether or not the
        player magically lands on the ring
121     player_placement_cell = _get_random_placement_cell()
122     set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
123     set_cell(0, previous_cell, 0) # replace the previous sprite
124     return
125     var new_placement_cell: Vector2i = player_placement_cell +

```

```

        direction
126     if (not get_used_cells(0).has(new_placement_cell) or trees.has(
            get_cell_atlas_coords(0, new_placement_cell)) or
            new_placement_cell == ring_placement_cell) and
            _is_not_out_of_bounds(new_placement_cell):
127         player_placement_cell = new_placement_cell
128         set_cell(0, previous_cell, 0) # deletes contents of previous
            cell (atlas_coords = Vector2i(-1, -1))
129         set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
130         if player_placement_cell == ring_placement_cell:
131             $WinDialog.visible = true
132             get_tree().paused = true
133
134     # ALGORITHM BEGINS HERE
135
136     func paint_points() -> void:
137         for point in points:
138             set_cell(0, Vector2(point["x"], point["y"]), 0, point["type"]
                ])
139             for citizen in point["citizens"]:
140                 if _is_in_bounds(point["x"], citizen["dx"], point["y"],
                    citizen["dy"]):
141                     set_cell(0, Vector2(point["x"] + citizen["dx"], point["
                        y"] + citizen["dy"]), 0, point["type"])
142
143     func _is_in_bounds(x: int, dx: int, y: int, dy: int) -> bool:
144         return x + dx >= 0 and x + dx < x_tile_range and y + dy >= 0 and
            y + dy < y_tile_range
145
146     func _squared(x: int) -> int:
147         return x ** 2
148
149     func calculate_points_delta(x: int, y: int, p: int) -> float:

```



```

150     if distance == EUCLIDEAN:
151         return sqrt(_squared(points[p]["x"] - x) + _squared(points[p]
152             ["y"] - y))
153     return abs(points[p]["x"] - x) + abs(points[p]["y"] - y)
154
155 func define_points(num_points: int) -> void:
156     var types: Array[Vector2i] = trees.duplicate()
157     types.append_array(buildings)
158     for i in range(num_points):
159         var x: int = randi_range(0, x_tile_range)
160         var y: int = randi_range(0, y_tile_range)
161         var type: Vector2i = types.pick_random()
162         types.erase(type)
163         points.append(
164             {
165                 "type": type,
166                 "x": x,
167                 "y": y,
168                 "citizens": []
169             }
170         )
171     for x in range(x_tile_range):
172         for y in range(y_tile_range):
173             var lowest_delta: Dictionary = {
174                 "point_id": 0,
175                 "delta": x_tile_range * y_tile_range
176             }
177             for p in range(len(points)):
178                 var delta: float = calculate_points_delta(x, y, p)
179                 if delta < lowest_delta["delta"]:
180                     lowest_delta = {
181                         "point_id": p,
182                         "delta": delta

```

```

182         }
183         var active_point: Dictionary = points[lowest_delta["
            point_id"]]
184         var dx: int = x - active_point["x"]
185         var dy: int = y - active_point["y"]
186         active_point["citizens"].append(
187             {
188                 "dx": dx,
189                 "dy": dy
190             }
191         )

```

C.5.6 accept_dialog.tscn

```

1  [gd_scene format=3 uid="uid://cau5jgogdnf53"]
2
3  [node name="AcceptDialog" type="AcceptDialog"]
4  title = "Tree-Munching Time!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Bring it on!"
9  dialog_text = "You're a hollow Golem who seeks the ultimate
    treasure; a ring that's got something on top of it. It's
    somewhere in this large village and barely visible to your
    naked eyes, but you'll stop at nothing to get what you want.
    You can chow down every tree and fauna that stands in your way
    of the ring, but your Achilles heel is any bricks and mortar,
    which will make you stop at your tracks. Are you ready to
    attain your treasure?w Golem in a black-and-white world, in
    search for your most desired treasure. It's a ring with
    something on top of it. And you'll stop at nothing to get what

```

```

    you want. You can chow down every tree and fauna that stands in
    your way of the ring, but your Achilles heel is any bricks and
    mortar, which will make you stop at your tracks. Are you ready
    to attain the treasure that is rightfully yours?!"
10  dialog_autowrap = true

```

C.5.7 win_dialog.tscn

```

1  [gd_scene format=3 uid="uid://b5q8ovcigrvyr"]
2
3  [node name="WinDialog" type="ConfirmationDialog"]
4  title = "You Found the Treasure!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Get Me a New Village"
9  dialog_text = "You found your treasure! Well done, you!"
10
11  Would you like to travel to a new village in the hopes of finding
    another ring? Or would you like to take your treasure home now?
    "
12  dialog_autowrap = true
13  cancel_button_text = "Get Me Out of Here"

```

C.5.8 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.5.9 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

```

C.6 Noise Demo

C.6.1 .gitattributes

```

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

```

C.6.2 .gitignore

```

1  # Godot 4+ specific ignores
2  .godot/

```

C.6.3 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="Noise Demo"
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  environment/defaults/default_clear_color=Color(0, 0, 0, 1)

```

C.6.4 tile_map.tscn

```

1  [gd_scene load_steps=7 format=3 uid="uid://d4jdcavluwx6s"]
2
3  [ext_resource type="Texture2D" uid="uid://m662wwd4prmn" path="res
    ://monochrome_packed.png" id="1_ld7xx"]
4  [ext_resource type="Script" path="res://tile_map.gd" id="2_o1dn1"]
5  [ext_resource type="PackedScene" uid="uid://cau5jgogdnf53" path="
    res://accept_dialog.tscn" id="3_e0ur6"]
6  [ext_resource type="PackedScene" uid="uid://b5q8ovcigrvyr" path="
    res://win_dialog.tscn" id="4_ecfaa"]
7
8  [sub_resource type="TileSetAtlasSource" id="

```

```

    TileSetAtlasSource_1e80b"]
9    texture = ExtResource("1_ld7xx")
10   0:0/0 = 0
11   1:0/0 = 0
12   2:0/0 = 0
13   3:0/0 = 0
14   4:0/0 = 0
15   5:0/0 = 0
16   6:0/0 = 0
17   7:0/0 = 0
18   8:0/0 = 0
19   9:0/0 = 0
20  10:0/0 = 0
21  11:0/0 = 0
22  12:0/0 = 0
23  13:0/0 = 0
24  14:0/0 = 0
25  15:0/0 = 0
26  16:0/0 = 0
27  17:0/0 = 0
28  18:0/0 = 0
29  19:0/0 = 0
30  20:0/0 = 0
31  21:0/0 = 0
32  22:0/0 = 0
33  23:0/0 = 0
34  24:0/0 = 0
35  25:0/0 = 0
36  26:0/0 = 0
37  27:0/0 = 0
38  28:0/0 = 0
39  29:0/0 = 0
40  30:0/0 = 0

```


41 31:0/0 = 0
42 32:0/0 = 0
43 33:0/0 = 0
44 34:0/0 = 0
45 35:0/0 = 0
46 36:0/0 = 0
47 37:0/0 = 0
48 38:0/0 = 0
49 39:0/0 = 0
50 40:0/0 = 0
51 41:0/0 = 0
52 42:0/0 = 0
53 43:0/0 = 0
54 44:0/0 = 0
55 45:0/0 = 0
56 46:0/0 = 0
57 47:0/0 = 0
58 48:0/0 = 0
59 0:1/0 = 0
60 1:1/0 = 0
61 2:1/0 = 0
62 3:1/0 = 0
63 4:1/0 = 0
64 5:1/0 = 0
65 6:1/0 = 0
66 7:1/0 = 0
67 8:1/0 = 0
68 9:1/0 = 0
69 10:1/0 = 0
70 11:1/0 = 0
71 12:1/0 = 0
72 13:1/0 = 0
73 14:1/0 = 0

74 15:1/0 = 0
75 16:1/0 = 0
76 17:1/0 = 0
77 18:1/0 = 0
78 19:1/0 = 0
79 20:1/0 = 0
80 21:1/0 = 0
81 22:1/0 = 0
82 23:1/0 = 0
83 24:1/0 = 0
84 25:1/0 = 0
85 26:1/0 = 0
86 27:1/0 = 0
87 28:1/0 = 0
88 29:1/0 = 0
89 30:1/0 = 0
90 31:1/0 = 0
91 32:1/0 = 0
92 33:1/0 = 0
93 34:1/0 = 0
94 35:1/0 = 0
95 36:1/0 = 0
96 37:1/0 = 0
97 38:1/0 = 0
98 39:1/0 = 0
99 40:1/0 = 0
100 41:1/0 = 0
101 42:1/0 = 0
102 43:1/0 = 0
103 44:1/0 = 0
104 45:1/0 = 0
105 46:1/0 = 0
106 47:1/0 = 0

107	$48:1/0 = 0$
108	$0:2/0 = 0$
109	$1:2/0 = 0$
110	$2:2/0 = 0$
111	$3:2/0 = 0$
112	$4:2/0 = 0$
113	$5:2/0 = 0$
114	$6:2/0 = 0$
115	$7:2/0 = 0$
116	$8:2/0 = 0$
117	$9:2/0 = 0$
118	$10:2/0 = 0$
119	$11:2/0 = 0$
120	$12:2/0 = 0$
121	$13:2/0 = 0$
122	$14:2/0 = 0$
123	$15:2/0 = 0$
124	$16:2/0 = 0$
125	$17:2/0 = 0$
126	$18:2/0 = 0$
127	$19:2/0 = 0$
128	$20:2/0 = 0$
129	$21:2/0 = 0$
130	$22:2/0 = 0$
131	$23:2/0 = 0$
132	$24:2/0 = 0$
133	$25:2/0 = 0$
134	$26:2/0 = 0$
135	$27:2/0 = 0$
136	$28:2/0 = 0$
137	$29:2/0 = 0$
138	$30:2/0 = 0$
139	$31:2/0 = 0$

140	$32:2/0 = 0$
141	$33:2/0 = 0$
142	$34:2/0 = 0$
143	$35:2/0 = 0$
144	$36:2/0 = 0$
145	$37:2/0 = 0$
146	$38:2/0 = 0$
147	$39:2/0 = 0$
148	$40:2/0 = 0$
149	$41:2/0 = 0$
150	$42:2/0 = 0$
151	$43:2/0 = 0$
152	$44:2/0 = 0$
153	$45:2/0 = 0$
154	$46:2/0 = 0$
155	$47:2/0 = 0$
156	$48:2/0 = 0$
157	$0:3/0 = 0$
158	$1:3/0 = 0$
159	$2:3/0 = 0$
160	$3:3/0 = 0$
161	$4:3/0 = 0$
162	$5:3/0 = 0$
163	$6:3/0 = 0$
164	$7:3/0 = 0$
165	$8:3/0 = 0$
166	$9:3/0 = 0$
167	$10:3/0 = 0$
168	$11:3/0 = 0$
169	$12:3/0 = 0$
170	$13:3/0 = 0$
171	$14:3/0 = 0$
172	$15:3/0 = 0$

173	$16:3/0 = 0$
174	$17:3/0 = 0$
175	$18:3/0 = 0$
176	$19:3/0 = 0$
177	$20:3/0 = 0$
178	$21:3/0 = 0$
179	$22:3/0 = 0$
180	$23:3/0 = 0$
181	$24:3/0 = 0$
182	$25:3/0 = 0$
183	$26:3/0 = 0$
184	$27:3/0 = 0$
185	$28:3/0 = 0$
186	$29:3/0 = 0$
187	$30:3/0 = 0$
188	$31:3/0 = 0$
189	$32:3/0 = 0$
190	$33:3/0 = 0$
191	$34:3/0 = 0$
192	$35:3/0 = 0$
193	$36:3/0 = 0$
194	$37:3/0 = 0$
195	$38:3/0 = 0$
196	$39:3/0 = 0$
197	$40:3/0 = 0$
198	$41:3/0 = 0$
199	$42:3/0 = 0$
200	$43:3/0 = 0$
201	$44:3/0 = 0$
202	$45:3/0 = 0$
203	$46:3/0 = 0$
204	$47:3/0 = 0$
205	$48:3/0 = 0$

206	$0:4/0 = 0$
207	$1:4/0 = 0$
208	$2:4/0 = 0$
209	$3:4/0 = 0$
210	$4:4/0 = 0$
211	$5:4/0 = 0$
212	$6:4/0 = 0$
213	$7:4/0 = 0$
214	$8:4/0 = 0$
215	$9:4/0 = 0$
216	$10:4/0 = 0$
217	$11:4/0 = 0$
218	$12:4/0 = 0$
219	$13:4/0 = 0$
220	$14:4/0 = 0$
221	$15:4/0 = 0$
222	$16:4/0 = 0$
223	$17:4/0 = 0$
224	$18:4/0 = 0$
225	$19:4/0 = 0$
226	$20:4/0 = 0$
227	$21:4/0 = 0$
228	$22:4/0 = 0$
229	$23:4/0 = 0$
230	$24:4/0 = 0$
231	$25:4/0 = 0$
232	$26:4/0 = 0$
233	$27:4/0 = 0$
234	$28:4/0 = 0$
235	$29:4/0 = 0$
236	$30:4/0 = 0$
237	$31:4/0 = 0$
238	$32:4/0 = 0$

239 33:4/0 = 0
240 34:4/0 = 0
241 35:4/0 = 0
242 36:4/0 = 0
243 37:4/0 = 0
244 38:4/0 = 0
245 39:4/0 = 0
246 40:4/0 = 0
247 41:4/0 = 0
248 42:4/0 = 0
249 43:4/0 = 0
250 44:4/0 = 0
251 45:4/0 = 0
252 46:4/0 = 0
253 47:4/0 = 0
254 48:4/0 = 0
255 0:5/0 = 0
256 1:5/0 = 0
257 2:5/0 = 0
258 3:5/0 = 0
259 4:5/0 = 0
260 5:5/0 = 0
261 6:5/0 = 0
262 7:5/0 = 0
263 8:5/0 = 0
264 9:5/0 = 0
265 10:5/0 = 0
266 11:5/0 = 0
267 12:5/0 = 0
268 13:5/0 = 0
269 14:5/0 = 0
270 15:5/0 = 0
271 16:5/0 = 0

272	$17:5/0 = 0$
273	$18:5/0 = 0$
274	$19:5/0 = 0$
275	$20:5/0 = 0$
276	$21:5/0 = 0$
277	$22:5/0 = 0$
278	$23:5/0 = 0$
279	$24:5/0 = 0$
280	$25:5/0 = 0$
281	$26:5/0 = 0$
282	$27:5/0 = 0$
283	$28:5/0 = 0$
284	$29:5/0 = 0$
285	$30:5/0 = 0$
286	$31:5/0 = 0$
287	$32:5/0 = 0$
288	$33:5/0 = 0$
289	$34:5/0 = 0$
290	$35:5/0 = 0$
291	$36:5/0 = 0$
292	$37:5/0 = 0$
293	$38:5/0 = 0$
294	$39:5/0 = 0$
295	$40:5/0 = 0$
296	$41:5/0 = 0$
297	$42:5/0 = 0$
298	$43:5/0 = 0$
299	$44:5/0 = 0$
300	$45:5/0 = 0$
301	$46:5/0 = 0$
302	$47:5/0 = 0$
303	$48:5/0 = 0$
304	$0:6/0 = 0$

305 1:6/0 = 0
306 2:6/0 = 0
307 3:6/0 = 0
308 4:6/0 = 0
309 5:6/0 = 0
310 6:6/0 = 0
311 7:6/0 = 0
312 8:6/0 = 0
313 9:6/0 = 0
314 10:6/0 = 0
315 11:6/0 = 0
316 12:6/0 = 0
317 13:6/0 = 0
318 14:6/0 = 0
319 15:6/0 = 0
320 16:6/0 = 0
321 17:6/0 = 0
322 18:6/0 = 0
323 19:6/0 = 0
324 20:6/0 = 0
325 21:6/0 = 0
326 22:6/0 = 0
327 23:6/0 = 0
328 24:6/0 = 0
329 25:6/0 = 0
330 26:6/0 = 0
331 27:6/0 = 0
332 28:6/0 = 0
333 29:6/0 = 0
334 30:6/0 = 0
335 31:6/0 = 0
336 32:6/0 = 0
337 33:6/0 = 0

338 34:6/0 = 0
339 35:6/0 = 0
340 36:6/0 = 0
341 37:6/0 = 0
342 38:6/0 = 0
343 39:6/0 = 0
344 40:6/0 = 0
345 41:6/0 = 0
346 42:6/0 = 0
347 43:6/0 = 0
348 44:6/0 = 0
349 45:6/0 = 0
350 46:6/0 = 0
351 47:6/0 = 0
352 48:6/0 = 0
353 0:7/0 = 0
354 1:7/0 = 0
355 2:7/0 = 0
356 3:7/0 = 0
357 4:7/0 = 0
358 5:7/0 = 0
359 6:7/0 = 0
360 7:7/0 = 0
361 8:7/0 = 0
362 9:7/0 = 0
363 10:7/0 = 0
364 11:7/0 = 0
365 12:7/0 = 0
366 13:7/0 = 0
367 14:7/0 = 0
368 15:7/0 = 0
369 16:7/0 = 0
370 17:7/0 = 0

371 18:7/0 = 0
372 19:7/0 = 0
373 20:7/0 = 0
374 21:7/0 = 0
375 22:7/0 = 0
376 23:7/0 = 0
377 24:7/0 = 0
378 25:7/0 = 0
379 26:7/0 = 0
380 27:7/0 = 0
381 28:7/0 = 0
382 29:7/0 = 0
383 30:7/0 = 0
384 31:7/0 = 0
385 32:7/0 = 0
386 33:7/0 = 0
387 34:7/0 = 0
388 35:7/0 = 0
389 36:7/0 = 0
390 37:7/0 = 0
391 38:7/0 = 0
392 39:7/0 = 0
393 40:7/0 = 0
394 41:7/0 = 0
395 42:7/0 = 0
396 43:7/0 = 0
397 44:7/0 = 0
398 45:7/0 = 0
399 46:7/0 = 0
400 47:7/0 = 0
401 48:7/0 = 0
402 0:8/0 = 0
403 1:8/0 = 0

404	$2:8/0 = 0$
405	$3:8/0 = 0$
406	$4:8/0 = 0$
407	$5:8/0 = 0$
408	$6:8/0 = 0$
409	$7:8/0 = 0$
410	$8:8/0 = 0$
411	$9:8/0 = 0$
412	$10:8/0 = 0$
413	$11:8/0 = 0$
414	$12:8/0 = 0$
415	$13:8/0 = 0$
416	$14:8/0 = 0$
417	$15:8/0 = 0$
418	$16:8/0 = 0$
419	$17:8/0 = 0$
420	$18:8/0 = 0$
421	$19:8/0 = 0$
422	$20:8/0 = 0$
423	$21:8/0 = 0$
424	$22:8/0 = 0$
425	$23:8/0 = 0$
426	$24:8/0 = 0$
427	$25:8/0 = 0$
428	$26:8/0 = 0$
429	$27:8/0 = 0$
430	$28:8/0 = 0$
431	$29:8/0 = 0$
432	$30:8/0 = 0$
433	$31:8/0 = 0$
434	$32:8/0 = 0$
435	$33:8/0 = 0$
436	$34:8/0 = 0$

437 35:8/0 = 0
438 36:8/0 = 0
439 37:8/0 = 0
440 38:8/0 = 0
441 39:8/0 = 0
442 40:8/0 = 0
443 41:8/0 = 0
444 42:8/0 = 0
445 43:8/0 = 0
446 44:8/0 = 0
447 45:8/0 = 0
448 46:8/0 = 0
449 47:8/0 = 0
450 48:8/0 = 0
451 0:9/0 = 0
452 1:9/0 = 0
453 2:9/0 = 0
454 3:9/0 = 0
455 4:9/0 = 0
456 5:9/0 = 0
457 6:9/0 = 0
458 7:9/0 = 0
459 8:9/0 = 0
460 9:9/0 = 0
461 10:9/0 = 0
462 11:9/0 = 0
463 12:9/0 = 0
464 13:9/0 = 0
465 14:9/0 = 0
466 15:9/0 = 0
467 16:9/0 = 0
468 17:9/0 = 0
469 18:9/0 = 0

470 19:9/0 = 0
471 20:9/0 = 0
472 21:9/0 = 0
473 22:9/0 = 0
474 23:9/0 = 0
475 24:9/0 = 0
476 25:9/0 = 0
477 26:9/0 = 0
478 27:9/0 = 0
479 28:9/0 = 0
480 29:9/0 = 0
481 30:9/0 = 0
482 31:9/0 = 0
483 32:9/0 = 0
484 33:9/0 = 0
485 34:9/0 = 0
486 35:9/0 = 0
487 36:9/0 = 0
488 37:9/0 = 0
489 38:9/0 = 0
490 39:9/0 = 0
491 40:9/0 = 0
492 41:9/0 = 0
493 42:9/0 = 0
494 43:9/0 = 0
495 44:9/0 = 0
496 45:9/0 = 0
497 46:9/0 = 0
498 47:9/0 = 0
499 48:9/0 = 0
500 0:10/0 = 0
501 1:10/0 = 0
502 2:10/0 = 0

503 3:10/0 = 0
504 4:10/0 = 0
505 5:10/0 = 0
506 6:10/0 = 0
507 7:10/0 = 0
508 8:10/0 = 0
509 9:10/0 = 0
510 10:10/0 = 0
511 11:10/0 = 0
512 12:10/0 = 0
513 13:10/0 = 0
514 14:10/0 = 0
515 15:10/0 = 0
516 16:10/0 = 0
517 17:10/0 = 0
518 18:10/0 = 0
519 19:10/0 = 0
520 20:10/0 = 0
521 21:10/0 = 0
522 22:10/0 = 0
523 23:10/0 = 0
524 24:10/0 = 0
525 25:10/0 = 0
526 26:10/0 = 0
527 27:10/0 = 0
528 28:10/0 = 0
529 29:10/0 = 0
530 30:10/0 = 0
531 31:10/0 = 0
532 32:10/0 = 0
533 33:10/0 = 0
534 34:10/0 = 0
535 35:10/0 = 0

536 36:10/0 = 0
537 37:10/0 = 0
538 38:10/0 = 0
539 39:10/0 = 0
540 40:10/0 = 0
541 41:10/0 = 0
542 42:10/0 = 0
543 43:10/0 = 0
544 44:10/0 = 0
545 45:10/0 = 0
546 46:10/0 = 0
547 47:10/0 = 0
548 48:10/0 = 0
549 0:11/0 = 0
550 1:11/0 = 0
551 2:11/0 = 0
552 3:11/0 = 0
553 4:11/0 = 0
554 5:11/0 = 0
555 6:11/0 = 0
556 7:11/0 = 0
557 8:11/0 = 0
558 9:11/0 = 0
559 10:11/0 = 0
560 11:11/0 = 0
561 12:11/0 = 0
562 13:11/0 = 0
563 14:11/0 = 0
564 15:11/0 = 0
565 16:11/0 = 0
566 17:11/0 = 0
567 18:11/0 = 0
568 19:11/0 = 0

569 20:11/0 = 0
570 21:11/0 = 0
571 22:11/0 = 0
572 23:11/0 = 0
573 24:11/0 = 0
574 25:11/0 = 0
575 26:11/0 = 0
576 27:11/0 = 0
577 28:11/0 = 0
578 29:11/0 = 0
579 30:11/0 = 0
580 31:11/0 = 0
581 32:11/0 = 0
582 33:11/0 = 0
583 34:11/0 = 0
584 35:11/0 = 0
585 36:11/0 = 0
586 37:11/0 = 0
587 38:11/0 = 0
588 39:11/0 = 0
589 40:11/0 = 0
590 41:11/0 = 0
591 42:11/0 = 0
592 43:11/0 = 0
593 44:11/0 = 0
594 45:11/0 = 0
595 46:11/0 = 0
596 47:11/0 = 0
597 48:11/0 = 0
598 0:12/0 = 0
599 1:12/0 = 0
600 2:12/0 = 0
601 3:12/0 = 0

602 4:12/0 = 0
603 5:12/0 = 0
604 6:12/0 = 0
605 7:12/0 = 0
606 8:12/0 = 0
607 9:12/0 = 0
608 10:12/0 = 0
609 11:12/0 = 0
610 12:12/0 = 0
611 13:12/0 = 0
612 14:12/0 = 0
613 15:12/0 = 0
614 16:12/0 = 0
615 17:12/0 = 0
616 18:12/0 = 0
617 19:12/0 = 0
618 20:12/0 = 0
619 21:12/0 = 0
620 22:12/0 = 0
621 23:12/0 = 0
622 24:12/0 = 0
623 25:12/0 = 0
624 26:12/0 = 0
625 27:12/0 = 0
626 28:12/0 = 0
627 29:12/0 = 0
628 30:12/0 = 0
629 31:12/0 = 0
630 32:12/0 = 0
631 33:12/0 = 0
632 34:12/0 = 0
633 35:12/0 = 0
634 36:12/0 = 0

635 $37:12/0 = 0$
636 $38:12/0 = 0$
637 $39:12/0 = 0$
638 $40:12/0 = 0$
639 $41:12/0 = 0$
640 $42:12/0 = 0$
641 $43:12/0 = 0$
642 $44:12/0 = 0$
643 $45:12/0 = 0$
644 $46:12/0 = 0$
645 $47:12/0 = 0$
646 $48:12/0 = 0$
647 $0:13/0 = 0$
648 $1:13/0 = 0$
649 $2:13/0 = 0$
650 $3:13/0 = 0$
651 $4:13/0 = 0$
652 $5:13/0 = 0$
653 $6:13/0 = 0$
654 $7:13/0 = 0$
655 $8:13/0 = 0$
656 $9:13/0 = 0$
657 $10:13/0 = 0$
658 $11:13/0 = 0$
659 $12:13/0 = 0$
660 $13:13/0 = 0$
661 $14:13/0 = 0$
662 $15:13/0 = 0$
663 $16:13/0 = 0$
664 $17:13/0 = 0$
665 $18:13/0 = 0$
666 $19:13/0 = 0$
667 $20:13/0 = 0$

668 21:13/0 = 0
669 22:13/0 = 0
670 23:13/0 = 0
671 24:13/0 = 0
672 25:13/0 = 0
673 26:13/0 = 0
674 27:13/0 = 0
675 28:13/0 = 0
676 29:13/0 = 0
677 30:13/0 = 0
678 31:13/0 = 0
679 32:13/0 = 0
680 33:13/0 = 0
681 34:13/0 = 0
682 35:13/0 = 0
683 36:13/0 = 0
684 37:13/0 = 0
685 38:13/0 = 0
686 39:13/0 = 0
687 40:13/0 = 0
688 41:13/0 = 0
689 42:13/0 = 0
690 43:13/0 = 0
691 44:13/0 = 0
692 45:13/0 = 0
693 46:13/0 = 0
694 47:13/0 = 0
695 48:13/0 = 0
696 0:14/0 = 0
697 1:14/0 = 0
698 2:14/0 = 0
699 3:14/0 = 0
700 4:14/0 = 0

701 5:14/0 = 0
702 6:14/0 = 0
703 7:14/0 = 0
704 8:14/0 = 0
705 9:14/0 = 0
706 10:14/0 = 0
707 11:14/0 = 0
708 12:14/0 = 0
709 13:14/0 = 0
710 14:14/0 = 0
711 15:14/0 = 0
712 16:14/0 = 0
713 17:14/0 = 0
714 18:14/0 = 0
715 19:14/0 = 0
716 20:14/0 = 0
717 21:14/0 = 0
718 22:14/0 = 0
719 23:14/0 = 0
720 24:14/0 = 0
721 25:14/0 = 0
722 26:14/0 = 0
723 27:14/0 = 0
724 28:14/0 = 0
725 29:14/0 = 0
726 30:14/0 = 0
727 31:14/0 = 0
728 32:14/0 = 0
729 33:14/0 = 0
730 34:14/0 = 0
731 35:14/0 = 0
732 36:14/0 = 0
733 37:14/0 = 0

734	$38:14/0 = 0$
735	$39:14/0 = 0$
736	$40:14/0 = 0$
737	$41:14/0 = 0$
738	$42:14/0 = 0$
739	$43:14/0 = 0$
740	$44:14/0 = 0$
741	$45:14/0 = 0$
742	$46:14/0 = 0$
743	$47:14/0 = 0$
744	$48:14/0 = 0$
745	$0:15/0 = 0$
746	$1:15/0 = 0$
747	$2:15/0 = 0$
748	$3:15/0 = 0$
749	$4:15/0 = 0$
750	$5:15/0 = 0$
751	$6:15/0 = 0$
752	$7:15/0 = 0$
753	$8:15/0 = 0$
754	$9:15/0 = 0$
755	$10:15/0 = 0$
756	$11:15/0 = 0$
757	$12:15/0 = 0$
758	$13:15/0 = 0$
759	$14:15/0 = 0$
760	$15:15/0 = 0$
761	$16:15/0 = 0$
762	$17:15/0 = 0$
763	$18:15/0 = 0$
764	$19:15/0 = 0$
765	$20:15/0 = 0$
766	$21:15/0 = 0$

767 22:15/0 = 0
768 23:15/0 = 0
769 24:15/0 = 0
770 25:15/0 = 0
771 26:15/0 = 0
772 27:15/0 = 0
773 28:15/0 = 0
774 29:15/0 = 0
775 30:15/0 = 0
776 31:15/0 = 0
777 32:15/0 = 0
778 33:15/0 = 0
779 34:15/0 = 0
780 35:15/0 = 0
781 36:15/0 = 0
782 37:15/0 = 0
783 38:15/0 = 0
784 39:15/0 = 0
785 40:15/0 = 0
786 41:15/0 = 0
787 42:15/0 = 0
788 43:15/0 = 0
789 44:15/0 = 0
790 45:15/0 = 0
791 46:15/0 = 0
792 47:15/0 = 0
793 48:15/0 = 0
794 0:16/0 = 0
795 1:16/0 = 0
796 2:16/0 = 0
797 3:16/0 = 0
798 4:16/0 = 0
799 5:16/0 = 0

800 6:16/0 = 0
801 7:16/0 = 0
802 8:16/0 = 0
803 9:16/0 = 0
804 10:16/0 = 0
805 11:16/0 = 0
806 12:16/0 = 0
807 13:16/0 = 0
808 14:16/0 = 0
809 15:16/0 = 0
810 16:16/0 = 0
811 17:16/0 = 0
812 18:16/0 = 0
813 19:16/0 = 0
814 20:16/0 = 0
815 21:16/0 = 0
816 22:16/0 = 0
817 23:16/0 = 0
818 24:16/0 = 0
819 25:16/0 = 0
820 26:16/0 = 0
821 27:16/0 = 0
822 28:16/0 = 0
823 29:16/0 = 0
824 30:16/0 = 0
825 31:16/0 = 0
826 32:16/0 = 0
827 33:16/0 = 0
828 34:16/0 = 0
829 35:16/0 = 0
830 36:16/0 = 0
831 37:16/0 = 0
832 38:16/0 = 0

833 39:16/0 = 0
834 40:16/0 = 0
835 41:16/0 = 0
836 42:16/0 = 0
837 43:16/0 = 0
838 44:16/0 = 0
839 45:16/0 = 0
840 46:16/0 = 0
841 47:16/0 = 0
842 48:16/0 = 0
843 0:17/0 = 0
844 1:17/0 = 0
845 2:17/0 = 0
846 3:17/0 = 0
847 4:17/0 = 0
848 5:17/0 = 0
849 6:17/0 = 0
850 7:17/0 = 0
851 8:17/0 = 0
852 9:17/0 = 0
853 10:17/0 = 0
854 11:17/0 = 0
855 12:17/0 = 0
856 13:17/0 = 0
857 14:17/0 = 0
858 15:17/0 = 0
859 16:17/0 = 0
860 17:17/0 = 0
861 18:17/0 = 0
862 19:17/0 = 0
863 20:17/0 = 0
864 21:17/0 = 0
865 22:17/0 = 0

866 23:17/0 = 0
867 24:17/0 = 0
868 25:17/0 = 0
869 26:17/0 = 0
870 27:17/0 = 0
871 28:17/0 = 0
872 29:17/0 = 0
873 30:17/0 = 0
874 31:17/0 = 0
875 32:17/0 = 0
876 33:17/0 = 0
877 34:17/0 = 0
878 35:17/0 = 0
879 36:17/0 = 0
880 37:17/0 = 0
881 38:17/0 = 0
882 39:17/0 = 0
883 40:17/0 = 0
884 41:17/0 = 0
885 42:17/0 = 0
886 43:17/0 = 0
887 44:17/0 = 0
888 45:17/0 = 0
889 46:17/0 = 0
890 47:17/0 = 0
891 48:17/0 = 0
892 0:18/0 = 0
893 1:18/0 = 0
894 2:18/0 = 0
895 3:18/0 = 0
896 4:18/0 = 0
897 5:18/0 = 0
898 6:18/0 = 0

899 7:18/0 = 0
900 8:18/0 = 0
901 9:18/0 = 0
902 10:18/0 = 0
903 11:18/0 = 0
904 12:18/0 = 0
905 13:18/0 = 0
906 14:18/0 = 0
907 15:18/0 = 0
908 16:18/0 = 0
909 17:18/0 = 0
910 18:18/0 = 0
911 19:18/0 = 0
912 20:18/0 = 0
913 21:18/0 = 0
914 22:18/0 = 0
915 23:18/0 = 0
916 24:18/0 = 0
917 25:18/0 = 0
918 26:18/0 = 0
919 27:18/0 = 0
920 28:18/0 = 0
921 29:18/0 = 0
922 30:18/0 = 0
923 31:18/0 = 0
924 32:18/0 = 0
925 33:18/0 = 0
926 34:18/0 = 0
927 35:18/0 = 0
928 36:18/0 = 0
929 37:18/0 = 0
930 38:18/0 = 0
931 39:18/0 = 0

932 40:18/0 = 0
933 41:18/0 = 0
934 42:18/0 = 0
935 43:18/0 = 0
936 44:18/0 = 0
937 45:18/0 = 0
938 46:18/0 = 0
939 47:18/0 = 0
940 48:18/0 = 0
941 0:19/0 = 0
942 1:19/0 = 0
943 2:19/0 = 0
944 3:19/0 = 0
945 4:19/0 = 0
946 5:19/0 = 0
947 6:19/0 = 0
948 7:19/0 = 0
949 8:19/0 = 0
950 9:19/0 = 0
951 10:19/0 = 0
952 11:19/0 = 0
953 12:19/0 = 0
954 13:19/0 = 0
955 14:19/0 = 0
956 15:19/0 = 0
957 16:19/0 = 0
958 17:19/0 = 0
959 18:19/0 = 0
960 19:19/0 = 0
961 20:19/0 = 0
962 21:19/0 = 0
963 22:19/0 = 0
964 23:19/0 = 0

965 24:19/0 = 0
966 25:19/0 = 0
967 26:19/0 = 0
968 27:19/0 = 0
969 28:19/0 = 0
970 29:19/0 = 0
971 30:19/0 = 0
972 31:19/0 = 0
973 32:19/0 = 0
974 33:19/0 = 0
975 34:19/0 = 0
976 35:19/0 = 0
977 36:19/0 = 0
978 37:19/0 = 0
979 38:19/0 = 0
980 39:19/0 = 0
981 40:19/0 = 0
982 41:19/0 = 0
983 42:19/0 = 0
984 43:19/0 = 0
985 44:19/0 = 0
986 45:19/0 = 0
987 46:19/0 = 0
988 47:19/0 = 0
989 48:19/0 = 0
990 0:20/0 = 0
991 1:20/0 = 0
992 2:20/0 = 0
993 3:20/0 = 0
994 4:20/0 = 0
995 5:20/0 = 0
996 6:20/0 = 0
997 7:20/0 = 0

998	$8:20/0 = 0$
999	$9:20/0 = 0$
1000	$10:20/0 = 0$
1001	$11:20/0 = 0$
1002	$12:20/0 = 0$
1003	$13:20/0 = 0$
1004	$14:20/0 = 0$
1005	$15:20/0 = 0$
1006	$16:20/0 = 0$
1007	$17:20/0 = 0$
1008	$18:20/0 = 0$
1009	$19:20/0 = 0$
1010	$20:20/0 = 0$
1011	$21:20/0 = 0$
1012	$22:20/0 = 0$
1013	$23:20/0 = 0$
1014	$24:20/0 = 0$
1015	$25:20/0 = 0$
1016	$26:20/0 = 0$
1017	$27:20/0 = 0$
1018	$28:20/0 = 0$
1019	$29:20/0 = 0$
1020	$30:20/0 = 0$
1021	$31:20/0 = 0$
1022	$32:20/0 = 0$
1023	$33:20/0 = 0$
1024	$34:20/0 = 0$
1025	$35:20/0 = 0$
1026	$36:20/0 = 0$
1027	$37:20/0 = 0$
1028	$38:20/0 = 0$
1029	$39:20/0 = 0$
1030	$40:20/0 = 0$

1031	$41:20/0 = 0$
1032	$42:20/0 = 0$
1033	$43:20/0 = 0$
1034	$44:20/0 = 0$
1035	$45:20/0 = 0$
1036	$46:20/0 = 0$
1037	$47:20/0 = 0$
1038	$48:20/0 = 0$
1039	$0:21/0 = 0$
1040	$1:21/0 = 0$
1041	$2:21/0 = 0$
1042	$3:21/0 = 0$
1043	$4:21/0 = 0$
1044	$5:21/0 = 0$
1045	$6:21/0 = 0$
1046	$7:21/0 = 0$
1047	$8:21/0 = 0$
1048	$9:21/0 = 0$
1049	$10:21/0 = 0$
1050	$11:21/0 = 0$
1051	$12:21/0 = 0$
1052	$13:21/0 = 0$
1053	$14:21/0 = 0$
1054	$15:21/0 = 0$
1055	$16:21/0 = 0$
1056	$17:21/0 = 0$
1057	$18:21/0 = 0$
1058	$19:21/0 = 0$
1059	$20:21/0 = 0$
1060	$21:21/0 = 0$
1061	$22:21/0 = 0$
1062	$23:21/0 = 0$
1063	$24:21/0 = 0$

```

1064    25:21/0 = 0
1065    26:21/0 = 0
1066    27:21/0 = 0
1067    28:21/0 = 0
1068    29:21/0 = 0
1069    30:21/0 = 0
1070    31:21/0 = 0
1071    32:21/0 = 0
1072    33:21/0 = 0
1073    34:21/0 = 0
1074    35:21/0 = 0
1075    36:21/0 = 0
1076    37:21/0 = 0
1077    38:21/0 = 0
1078    39:21/0 = 0
1079    40:21/0 = 0
1080    41:21/0 = 0
1081    42:21/0 = 0
1082    43:21/0 = 0
1083    44:21/0 = 0
1084    45:21/0 = 0
1085    46:21/0 = 0
1086    47:21/0 = 0
1087    48:21/0 = 0
1088
1089    [sub_resource type="TileSet" id="TileSet_qtrb6"]
1090    sources/0 = SubResource("TileSetAtlasSource_1e80b")
1091
1092    [node name="TileMap" type="TileMap"]
1093    texture_filter = 1
1094    tile_set = SubResource("TileSet_qtrb6")
1095    format = 2
1096    script = ExtResource("2_01dn1")

```



```

1097
1098     [node name="AcceptDialog" parent="." instance=ExtResource("3_e0ur6"
1099         )]
1100
1101     [node name="WinDialog" parent="." instance=ExtResource("4_ecfaa")]
1102     title = "You Found the Treasure!"

```

C.6.5 tile_map.gd

```

1     extends TileMap
2
3     const buildings: Array[Vector2i] = [
4         Vector2i(0, 19),
5         Vector2i(1, 19),
6         Vector2i(2, 19),
7         Vector2i(3, 19),
8         Vector2i(4, 19),
9         Vector2i(5, 19),
10        Vector2i(6, 19),
11        Vector2i(7, 19),
12        Vector2i(8, 20),
13        Vector2i(0, 20),
14        Vector2i(1, 20),
15        Vector2i(2, 20),
16        Vector2i(3, 20),
17        Vector2i(4, 20),
18        Vector2i(5, 20),
19        Vector2i(6, 20),
20        Vector2i(7, 20),
21        Vector2i(8, 20),
22        Vector2i(0, 21),
23        Vector2i(1, 21),

```

```

24     Vector2i(2, 21),
25     Vector2i(3, 21),
26     Vector2i(4, 21),
27     Vector2i(5, 21),
28     Vector2i(6, 21),
29     Vector2i(7, 21),
30     Vector2i(8, 21)
31 ]
32 const trees: Array[Vector2i] = [
33     Vector2i(0,1),
34     Vector2i(1,1),
35     Vector2i(2,1),
36     Vector2i(3,1),
37     Vector2i(4,1),
38     Vector2i(5,1),
39     Vector2i(6,1),
40     Vector2i(7,1),
41     Vector2i(0,2),
42     Vector2i(1,2),
43     Vector2i(2,2),
44     Vector2i(3,2),
45     Vector2i(4,2)
46 ]
47 const PLAYER_SPRITE: Vector2i = Vector2i(24, 7)
48 var player_placement_cell: Vector2i
49 const rings: Array[Vector2i] = [
50     Vector2i(43, 6),
51     Vector2i(44, 6),
52     Vector2i(45, 6),
53     Vector2i(46, 6)
54 ]
55 var ring_placement_cell: Vector2i
56

```

```

57  var noise: FastNoiseLite
58  @export_enum("Perlin", "Simplex", "Simplex Smooth", "Value", "Value
    Cubic") var noise_type: String = "Simplex Smooth"
59  @export var fractal_type: FastNoiseLite.FractalType
60  @export var cellular_distance_type: FastNoiseLite.
    CellularDistanceFunction
61  #@export_range(1, 10, 1) var octaves: int = 5
62  @export_range(0.0, 1.0) var noise_frequency: float = 0.894
63
64  #@onready var timer: Timer = Timer.new()
65  #@export_range(10, 200, 10) var player_movement_speed: int = 100
66  @export_range(-1.0, 1.0) var tree_cap: float = -0.048
67  @export_range(-1.0, 1.0) var building_cap: float = -0.252
68  @export_range(0.0, 0.5) var building_overtakes_tree: float = 0.12
69  var x_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_width") / tile_set.tile_size.x
70  var y_tile_range: int = ProjectSettings.get_setting("display/window
    /size/viewport_height") / tile_set.tile_size.y
71
72  # Called when the node enters the scene tree for the first time.
73  func _ready() -> void:
74      randomize()
75      var start_time: float = Time.get_ticks_msec()
76      set_noise()
77      paint_tiles()
78      place_player()
79      place_ring()
80      var new_time: float = Time.get_ticks_msec() - start_time
81      print("Time taken: " + str(new_time) + "ms")
82      $AcceptDialog.dialog_text = "You're a hollow Golem who seeks the
    ultimate treasure; a ring that's got something on top of it
    . It's somewhere in this large village and barely visible to
    your naked eyes, which took us " + str(new_time) + "

```

```

    milliseconds to generate (" + str(new_time / 1000.0) + "
    seconds), but you'll stop at nothing to get what you want.
    You can chow down every tree and fauna that stands in your
    way of the ring, but your Achilles heel is any bricks and
    mortar, which WILL make you stop at your tracks. Since it's
    easy to get lost in here, we'll tell you that you're in
    position " + str(player_placement_cell) + " in this big
    village of size " + str(Vector2i(x_tile_range, y_tile_range)
    ) + ". However, it is YOUR job to find the ring, so are you
    ready to attain the treasure that is rightfully yours?!"

83     $AcceptDialog.visible = true
84     $AcceptDialog.confirmed.connect(_on_AcceptDialog_closed)
85     $AcceptDialog.canceled.connect(_on_AcceptDialog_closed)
86     $WinDialog.confirmed.connect(_on_WinDialog_confirmed)
87     $WinDialog.canceled.connect(_on_WinDialog_canceled)
88     get_tree().paused = true
89
90     func _on_WinDialog_confirmed() -> void:
91         get_tree().reload_current_scene()
92
93     func _on_WinDialog_canceled() -> void:
94         get_tree().quit()
95
96     func _on_AcceptDialog_closed() -> void:
97         $AcceptDialog.visible = false
98         get_tree().paused = false
99
100    func _get_random_placement_cell() -> Vector2i:
101        return Vector2i(randi() % x_tile_range, randi() % y_tile_range)
102
103    func place_player() -> void:
104        player_placement_cell = _get_random_placement_cell()
105        while get_used_cells(0).has(player_placement_cell):

```

```

106         player_placement_cell = _get_random_placement_cell()
107     set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)
108
109     func place_ring() -> void:
110         ring_placement_cell = _get_random_placement_cell()
111         while get_used_cells(0).has(ring_placement_cell):
112             ring_placement_cell = _get_random_placement_cell()
113         set_cell(0, ring_placement_cell, 0, rings.pick_random())
114
115     func _is_not_out_of_bounds(cell: Vector2i) -> bool:
116         return cell.x >= 0 and cell.x < x_tile_range and cell.y >= 0 and
            cell.y < y_tile_range
117
118     func _physics_process(_delta: float) -> void:
119         var previous_cell: Vector2i = player_placement_cell
120         var direction: Vector2i = Vector2i.ZERO
121         if Input.is_action_pressed("ui_up"): direction = Vector2i.UP
122         elif Input.is_action_pressed("ui_down"): direction = Vector2i.
            DOWN
123         elif Input.is_action_pressed("ui_left"): direction = Vector2i.
            LEFT
124         elif Input.is_action_pressed("ui_right"): direction = Vector2i.
            RIGHT
125         var new_placement_cell: Vector2i = player_placement_cell +
            direction
126         if (not get_used_cells(0).has(new_placement_cell) or trees.has(
            get_cell_atlas_coords(0, new_placement_cell)) or
            new_placement_cell == ring_placement_cell) and
            _is_not_out_of_bounds(new_placement_cell):
127             player_placement_cell = new_placement_cell
128             set_cell(0, previous_cell, 0) # deletes contents of previous
                cell (atlas_coords = Vector2i(-1, -1))
129             set_cell(0, player_placement_cell, 0, PLAYER_SPRITE)

```

```

130         if player_placement_cell == ring_placement_cell:
131             $WinDialog.visible = true
132             get_tree().paused = true
133
134     # ALGORITHM BEGINS HERE
135
136     func _get_noise_type() -> int:
137         match noise_type:
138             "Perlin": return 3
139             "Simplex": return 0
140             "Value": return 5
141             "Value Cubic": return 4
142             _: return 1 # Return Simplex Smooth by default
143
144     func set_noise() -> void:
145         noise = FastNoiseLite.new()
146         noise.frequency = noise_frequency
147         noise.noise_type = _get_noise_type() as FastNoiseLite.NoiseType
148         noise.fractal_type = fractal_type
149         noise.cellular_distance_function = cellular_distance_type
150         # noise.fractal_octaves = octaves
151         noise.seed = randi()
152
153     func paint_tiles() -> void:
154         for x in range(x_tile_range):
155             for y in range(y_tile_range):
156                 var noise_point: float = noise.get_noise_2d(x * tile_set.
                    tile_size.x, y * tile_set.tile_size.y)
157                 if noise_point < tree_cap and not get_used_cells(0).has(
                    Vector2i(x, y)):
158                     set_cell(0, Vector2i(x, y), 0, trees.pick_random())
159                 if ((building_cap <= tree_cap and randf() <
                    building_overtakes_tree) or (building_cap > tree_cap

```

```

        and noise_point < building_cap)) and not
        get_used_cells(0).has(Vector2i(x, y)):
160     set_cell(0, Vector2i(x, y), 0, buildings.pick_random())

```

C.6.6 accept_dialog.tscn

```

1  [gd_scene format=3 uid="uid://cau5jgogdnf53"]
2
3  [node name="AcceptDialog" type="AcceptDialog"]
4  title = "Tree-Munching Time!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Bring it on!"
9  dialog_text = "You're a hollow Golem who seeks the ultimate
    treasure; a ring that's got something on top of it. It's
    somewhere in this large village and barely visible to your
    naked eyes, but you'll stop at nothing to get what you want.
    You can chow down every tree and fauna that stands in your way
    of the ring, but your Achilles heel is any bricks and mortar,
    which will make you stop at your tracks. Are you ready to
    attain your treasure?w Golem in a black-and-white world, in
    search for your most desired treasure. It's a ring with
    something on top of it. And you'll stop at nothing to get what
    you want. You can chow down every tree and fauna that stands in
    your way of the ring, but your Achilles heel is any bricks and
    mortar, which will make you stop at your tracks. Are you ready
    to attain the treasure that is rightfully yours?!"
10 dialog_autowrap = true

```

C.6.7 win_dialog.tscn

```

1  [gd_scene format=3 uid="uid://b5q8ovcigrvyr"]
2
3  [node name="WinDialog" type="ConfirmationDialog"]
4  title = "Tree-Munching Time!"
5  position = Vector2i(326, 100)
6  size = Vector2i(500, 421)
7  mouse_passthrough = true
8  ok_button_text = "Get Me a New Village"
9  dialog_text = "You found your treasure! Well done, you!"
10
11  Would you like to travel to a new village in the hopes of finding
    another ring? Or would you like to take your treasure home now?
    "
12  dialog_autowrap = true
13  cancel_button_text = "Get Me Out of Here"

```

C.6.8 icon.svg.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"
5  uid="uid://crgf6ascxsdt0"
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.6.9 monochrome_packed.png.import

```

1  [remap]
2
3  importer="texture"
4  type="CompressedTexture2D"

```

```

5  uid="uid://m662wwd4prmn"
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

```