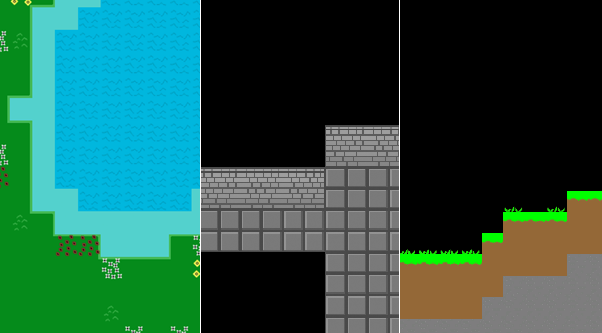
Teesside University

Computing Project

**(COM3051-N)**

Final Year Project: Report

Tilemap Generator



Aziz Arar (S6053935)

Contents

[Abstract 3](#_Toc7641012)

[Introduction 3](#_Toc7641013)

[Development and Implementation: Sprint 1 5](#_Toc7641014)

[Generation Window 5](#_Toc7641015)

[2D Top down 5](#_Toc7641016)

[Perlin Noise World Map 5](#_Toc7641017)

[Development and Implementation: Sprint 2 7](#_Toc7641018)

[Generation Window Updates 7](#_Toc7641019)

[2D Top down 7](#_Toc7641020)

[BSP Dungeon 7](#_Toc7641021)

[Perlin Noise World Map Updates 8](#_Toc7641022)

[Development and Implementation: Sprint 3 8](#_Toc7641023)

[2D Side scrolling 8](#_Toc7641024)

[Patch Notes 9](#_Toc7641025)

[Research 10](#_Toc7641026)

[World & Level Generation 10](#_Toc7641027)

[Perlin Noise 10](#_Toc7641028)

[Binary Space Partitioning 10](#_Toc7641029)

[Game Engines and Plugins 10](#_Toc7641030)

[Use of Level Generation in Games 11](#_Toc7641031)

[Terraria 11](#_Toc7641032)

[Enter the Gungeon 11](#_Toc7641033)

[Social and Ethical Considerations 11](#_Toc7641034)

[Testing and Evaluation 11](#_Toc7641035)

[Peer Feedback Received 11](#_Toc7641036)

[Conclusion 12](#_Toc7641037)

[Evaluation 12](#_Toc7641038)

[Future Plans 12](#_Toc7641039)

[Reflection of the Artefact 12](#_Toc7641040)

[References 13](#_Toc7641041)

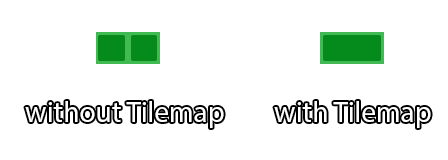
# Abstract

|  |
| --- |
|  |

# Introduction

My artefact is a Unity Editor Plugin which would randomly generate 2D maps using the Tilemap system. I have decided to take advantage of the Unity 2017 feature as it allows tiles to be painted on a grid using a Palette of various Tiles. By including the extra features found on Unity’s Tilemap 2d extras [1] repo, it allows a further extension of how these tiles can interact. I use various Tile types in my examples, such as Terrain tiles, Rule tiles, animated tiles and Random tiles. The 2d extras is not necessary for my Tilemaps, but it creates it gives the illusion of a more natural environment.

This is an example of the advantages of using a Tilemap. The left side shows without a Tilemap, and the Right side a Tilemap with Grass Terrain Tiles.



A window will be visible from the options Menu [Aziz > Tilemap Generator] which would allow the user to select the type of level they want generated, such as a World Map and Dungeon. The user would decide the size of the grid, and the Tiles generated on it. There are extra options such as options to generate a collision layer and whether foliage should generate, and the density of it.

There are two creation modes: Simple and Advanced. Simple is the general user trying to generate a level easily. Advanced mode allows the user to remove Grid restrictions, and to change the Animation Framerate, Pixels per Unit and Cell Size of the Tilemap.

I have decided to use Scrum as the methodology of this Project as it allows me to split my tasks up into sprints, to allow me to focus and finish one thing at a time. I am using HacknPlan to plan my tasks. This allows me to make sure certain features of the Tilemap Generator is up to scratch before ending the sprint. If I find any bugs during another sprint, I can add them to a bug list.

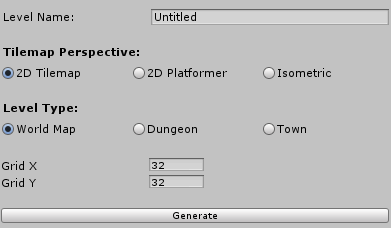
The reason I have decided to base my

Why this Artefact? The users.

# Development and Implementation: Sprint 1

## Generation Window

Firstly, to start the project off, I began with creating a Basic window which will be used so the User can decide what they want generated. To do this, I used the Unity ‘Editor Window’ Class, this class has various GUI Elements which will save time when putting this together.

This is the first draft of my editor window, using the GUI Layout Elements available to me, using Labels, Selection Grids, a Text Field and Int Fields I was able put this together. While it is not yet functional, I can begin working on the first part of the generation, which is the World Map using Perlin Noise.

I set the path of the Generation Window to the Unity Menu Bar > Aziz > Tilemap Generation. This way it should be immediately accessible by users.

## 2D Top down

### Perlin Noise World Map

To start on the Perlin Noise, I had to decide how I was going to connect it to the Tilemaps. I decided on using 2 arrays. An array of positions, and an array of Tiles. Unity’s ‘Mathf’ class has a Perlin Noise function inside of it which I decided to use.

Firstly, I set the size of the positions array by creating a new Vector3Int which was the width and height of the grid multiplied, with the size of the Tile Array copying the positions size. The Tile Array was an array of TileBase, which in Unity Tilemaps is a specific Tile type, such as Grass or Water. This way when the heights are calculated, this would be set to Grass for example, and when it sets the tile, it would know exactly what to set.

I then ran a loop, which for every position in the array, it would run the Perlin noise function to determine what would generate there.

|  |
| --- |
| for (int index = 0; index < positions.Length; index++)  {  positions[index] = new Vector3Int(index % gridX, index / gridY, 0);  float height = Mathf.PerlinNoise((float)positions[index].x / 10, (float)positions[index].y / 10);  } |

I need to create tiles, so I can see if the Perlin noise is working correctly, so using Photoshop 4 different tiles. (See Figure 2)

https://raw.githubusercontent.com/Bodmass/AzizArarFYP/37d8d603c627acf270738ede87707ac608bc0a1f/Aziz%20Arar%20s6053935%20FYP%20-%20Tilemap%20Procedural%20Generation/Assets/Demo/Sprites/2DTopdownWorldMapSpritesheet.png?token=AGKIN57DKCYTZVMRWJ47J6S4ZRQZE

Figure

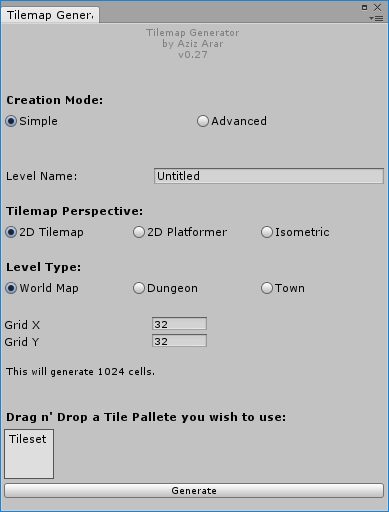
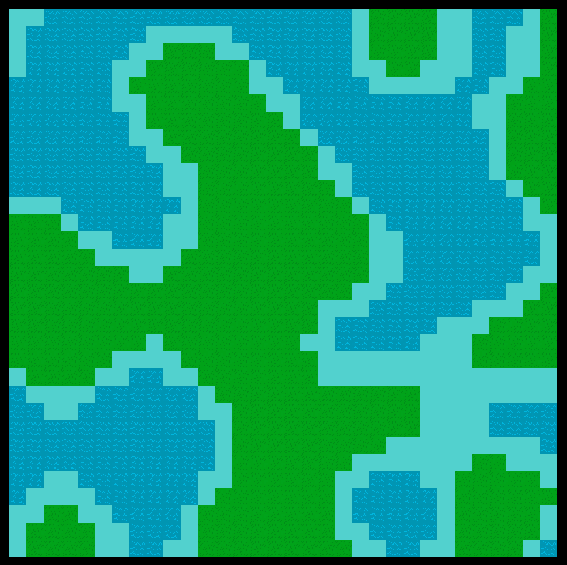
I created my own tiles to avoid any copyright issues, as the tiles will be used in a Demo included in the final version. Now I have the tiles setup, I can use them with the Perlin Noise.

I set up 4 TileBases in the Perlin Noise Generation Script, for Shore, Grass, Water and Mountain respectively. I needed to decide a height for the water, and as it is a World Map, I wanted there to be a high volume of water. I decided to start with.

By default I set the tiles to the Shore to ensure a Tile would appear in case of an error. I decided to split it like this, as the Height is 0 to 1. I did not include Mountains at the time in the Perlin Noise.

|  |  |
| --- | --- |
| Tile | Height |
| Water | 0f – 0.39f |
| Shore | 0.4f – 0.49f |
| Grass | 0.5f – 1.0f |
| Mountain | N/A |

|  |
| --- |
| for (int index = 0; index < positions.Length; index++)  {  positions[index] = new Vector3Int(index % gridX, index / gridY, 0);  float height = Mathf.PerlinNoise((float)positions[index].x / 10, (float)positions[index].y / 10);      tileArray[index] = Shore;  if(height >.5f)  {  tileArray[index] = Grass;  }  else if(height >.4f)  {  tileArray[index] = Shore;  }  else  {  tileArray[index] = Water;  }  thisMap.SetTiles(positions, tileArray); |

This proved to be successful, and generated a Map on a 32x32 Grid, which looked like this.

At this point, I wanted to implement this into the Generation Window, as I had to manually input values and tiles for the World Map Generation to use.

I gave the Generate button function, creating a Grid on the Scene with a Tilemap in it. This Tilemap contains the Generation Script. By having the Perspective and Level Type filters I was able to disable the Generate Button if I have not made it yet, so when it comes to getting feedback from my peers, they would not be able to unintentionally cause errors.

I wanted to Input a drag a drop system which would allow a set of tiles to be put in and sent together, However it does not have function at this time, so I still manually have to apply the Tiles. I also updated the Grid IntField, locking from going below 0.

I added a label beneath the Grid settings, which would give information. It would go between 2 errors, stating the grid is below 0 and when the grid is over 256. In this case, the Generate button would be disabled. If they are not, it would say the amount of cells that would be generated.

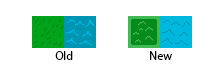
# Development and Implementation: Sprint 2

## Generation Window Updates

## 2D Top down

### BSP Dungeon

### Perlin Noise World Map Updates

Firstly, in my set of updates to the Perlin Noise World Map, I wanted to update the sprites and tiles included in my Demo to show how the tiles included 2d extras from Unity Technologies. The various different tiles I wanted to include was a Terrain tile for the grass, this will improve the overall look of how the grass connects together. The terrain tile allows you to create a sprite for each different position for the tile, which when you draw it automatically connects any nearby terrain tiles.

I also updated the tiles for Water by using an animated tile, which was a modification of the current water, and using the Flag effect in Adobe Premier Pro CC, then exporting 60 frames as PNGs.

Secondly, after receiving feedback about the level generator thus far, I wanted to improve the performance of the World Map generation. This is especially noticeable when you generate at bigger sizes, it takes longer for the Perlin Noise function significantly longer to run. I set out of a way to improve this and I came across the Unity Jobs class. An interface included in this called ‘IJobParallelFor’ allows for multiple threads to work in parallel to each other. Theoretically if I had the Perlin Noise work on multiple threads at the same time, it should significantly improve performance and loading time. I implemented a new job called Perlin Job. I managed to make the Job work, however using Deep Profiling, I did not notice a performance increase. I believe this is down to not fully understanding how to take advantage of the Jobs, but I decided to leave it in as it did not hinder performance.

# Development and Implementation: Sprint 3

## 2D Side scrolling

# Patch Notes

As the project only reached Version 0.7, I will include the patch notes in a table below of at what stages I achieved certain tasks. A similar version of the Patch Notes is included in the Project Readme file.

|  |  |  |
| --- | --- | --- |
| Version | Type | Patch Notes |
| 0.7 | | |
| Additions | |  |
| Added Reset Button |
| Added Stone Generation with Height Option |
| Changes | |  |
| Updated 2D Platformer Generation Tiles |
| Moved Foliage Density to Simple Options |
| Tilemap Perspective Label "2D Tilemap" changed to "2D Topdown" |
| Tilemap Perspective Label "2D Platformer" changed to "2D Sidescroller" |
| Removals | |  |
| Isometric Selection (To be reintroduced, Project requires Unity Version Upgrade) |
| Town Selection (To be reintroduced) |
| 0.6 | | |
| Additions | |  |
| Added 2D Platformer Integration to Generation Window |
| Added Foliage option to 2D Platformer |
| Added Foliage Density Option in Advanced Options |
| Changes | |  |
| Grid Size on 2D Topdown now ensures X and Y are the same due to bug |
| 0.5 | | |
| Additions | |  |
| Added Foliage to 2D Topdown : World Map (Demo Includes Random Tile) |
| Added Walls Generation to 2D Topdown : Dungeon |
| Added Grid Cap removal in Advanced Settings (\*) |
| Changes | |  |
| All Tilemaps Generated now default at 60fps |
| Updated 2D Platformer Generation, also now places down Grass and Dirt |
| 0.4 | | |
| Additions | |  |
| Added Collision Generation (Water and Dungeon BG) |
| Added Early Version of 2D Platformer, using 1D Perlin Noise, NYI into the Generation Window |
| 0.3 | | |
| Additions | |  |
| Added Tilemap Generator Base Class, all Generation Scripts will Inherit from this. |
| Added 2D Topdown Dungeon Generation using BSP |
| Removals | |  |
|  | | Removed Drag n Drop which did not work. |
| 0.2 | | |
| Additions | |  |
| Added Objects on the Generation Window for the Tiles |
| Added 2D Topdown World Map functionality to Generation Window |
|  | | Added Advanced Options with Cell Size and Pixel Per Unit options |
|  | | Added Box for a Not Yet Implemented TileBase Drag n Drop |
| 0.1 | | |
| Additions | |  |
| Added Basic Generation Window |
| Added 2D Topdown World Map generation using 2D Perlin Noise |

# Research

## World & Level Generation

There are different types of ways games creation randomness, from having Worlds and Levels generate, and having pre-created sections randomly fit together. It is vital for the players experience, that these games generate in a natural and seamless way. There are many different methods of generating levels.

### Perlin Noise

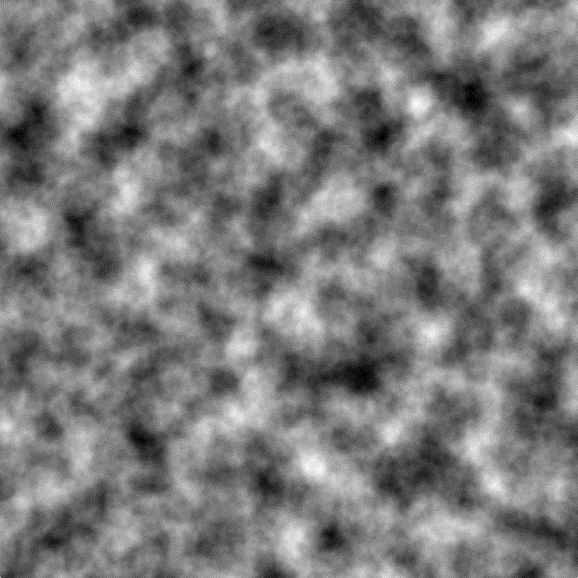
Firstly, Perlin Noise, developed in 1983 by Ken Perlin, is a type of gradient noise. The algorithm can be applied in different dimensions. 1D Perlin Noise generates what seems to be a line on a graph (See Figure). Which can be used for things like animation and, generating 2D hilly terrain. In 2 Dimensions, when generated, its appearance is a texture like static, with different heights/depths, depicted with the intensity of black and white. (See Figure 1). An example of it in use is generating top down outdoor levels using the lowest height for water, and the biggest height for mountains. Another example could be applying the noise to a texture or shader to achieve a dissolve or fire effect.

Figure ?

Perlin Noise can also be generated in the 3rd Dimension, however as we are working with Tilemaps which are exclusively 2D, we will not be using this dimension. I plan to use Perlin Noise for my 2D Topdown and Isometric World Map generation as I can run a Perlin Noise function for each position to determine a height, then save the height information for the Drawing Phase. I also plan to use Perlin Noise in 1 Dimension, this will allow me randomly generate heights to give a hilly effect on the generated level.

### Binary Space Partitioning

## Game Engines and Plugins

In this project I decided to use Unity 2018.2.15f1. This majority of the reason my Artefact was in Unity was to add more support for the Tilemap system which was added in Unity 2017. However, there are other game engines with support for a 2D Tile system, which I could have used instead. Unity has been continually supporting the creation of 2D games, with this system it makes creating 2D games easier as you can paint the tiles on, with different types of tiles available.

Unreal Engine 4 features Paper 2D, this is how Epic Games caters to the creators of 2D games. It allows the creation of Tile Sets and Tilemaps. It is possible to script random generation for the tile system; however, this system did not have an extension with different types of tiles unlike Unity’s Tilemap system.

(<https://docs.unrealengine.com/en-US/Engine/Paper2D>)

## Use of Level Generation in Games

### Terraria

World Generation

### Enter the Gungeon

Pre-created rooms linked together

# Social and Ethical Considerations

It is important to take into consideration the Social, Ethical and Political issues that could come with a project. As I am creating a Unity Plugin, I need to understand what goes into publishing an asset like this online. The Social and Ethical issues I have taken into consideration is the privacy which comes with receiving feedback from my peers. Each person I have received feedback I have outlined that their feedback will be used to improve my project, and their feedback will be stored and possibly used, and asking permission to reference them in a feedback report. For those I could ask I have received permission to take their feedback and apply it.

I need to ensure I comply with the Data Protection Act –

To ensure I do not come into contact with issues regarding Copyright, I decided to make my own tiles. The tiles will be available in the demo and will be released under CC-BY, allowing others to use these tiles if they wish, however must reference me as the creator of them.

# Testing and Evaluation

## Peer Feedback Received

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version | Feedback Received | Actions Taken |
| Samantha Kelly  (Games Art) | 0.3 | “I found it easy to generate a level from the tiles you provided, but for the World Map generation, As an artist, I feel it lacked immersion, adding the option to generate flowers and trees would add this to a generated world. “ | The feedback is greatly appreciated, in an upcoming version of the plugin I plan to add foliage to generate. The user will be able to decide if they want Foliage to generate, and at what |
| Callum Powley  (Games Programming) | 0.2 | “I think the generation is slower when I set the grid to a larger size, can you improve this” | I have attempted to implement a multi-threaded job for the Perlin Noise function. However at this time it requires more testing and tinkering to see if there is a performance increase. |

# Conclusion

## Evaluation

## Future Plans

## Reflection of the Artefact

# References

van der Linden, R., Lopes, R. and Bidarra, R., 2014. Procedural generation of dungeons. IEEE Transactions on Computational Intelligence and AI in Games, 6(1), pp.78-89.

Compton, K. and Mateas, M., 2006, June. Procedural Level Design for Platform Games. In AIIDE (pp. 109-111). Prachyabrued, M., Roden, T.E. and Benton, R.G., 2007, June. Procedural generation of stylized 2d maps.

In Proceedings of the international conference on Advances in computer entertainment technology (pp. 147-150). ACM.

Unity Technologies. 2018. Unity - Scripting API: Tilemap. [ONLINE] Available at: https://docs.unity3d.com/ScriptReference/Tilemaps.Tilemap.html. [Accessed 21 January 2019]

TO REFERENCE

<https://flafla2.github.io/2014/08/09/perlinnoise.html>

<https://docs.unity3d.com/2018.3/Documentation/ScriptReference/Unity.Jobs.IJobParallelFor.html>