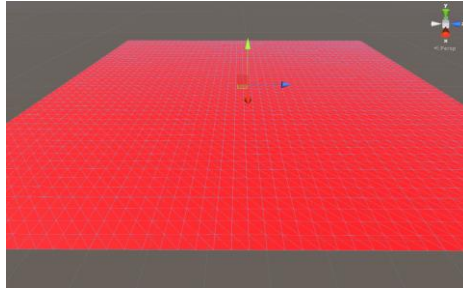


1. Diary Log

For this assignment, I've been given the task to create a procedurally generated terrain, the theme that I've select to use is the **Fort Town** theme.

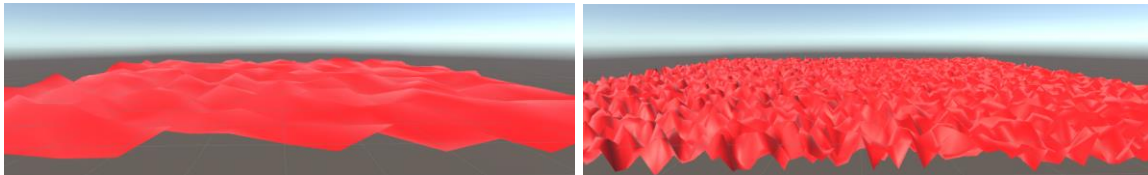
1.1 Random Terrain Heights

To begin with, I started with a plane, that I subdivided into many different vertices across the plane, this would allow for detailed height maps in future.



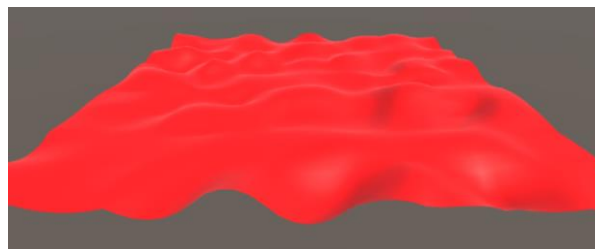
Plane subdivided into many vertices

From there, I needed to make the terrain height randomly generated, so to do this, I used the Rand() function to create a random value, then scaled it to an appropriate height value. This seemed promising when applied to a low subdivision plane, however for my project I wanted a more detailed terrain, increasing the subdivision count meant that the random values are too jagged and pointy; this is due to the fact that all points have been generated randomly, rather than random relative to their neighbors.



Left: Random height, SubDiv=20. Right: Random height, SubDiv = 100.

To solve this problem, I used Perlin noise, an algorithm used for creating random values that are relative to their neighbors. In one dimension, this is simple: for each random point, create it relative to the offset of the previous point. In two dimension it's a little bit harder, to help me develop this algorithm, I took inspiration from Biagioli's post (Biagioli, 2014). From that I removed some of the irrelevant sections like the permutation, and changed it from a 3D noise space to a 2D one, as a height map is only 2 dimensions.



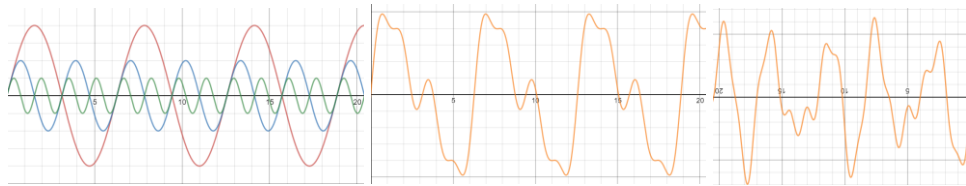
Perlin noise applied to the terrain

As Perlin noise works on a grid-based system, when it links to those grid anchor points, it can produce a kink in the terrain, to solve this, I used a mathematical formula: $6t^5 - 15t^4 + 10t^3$ (Biagioli, 2014)

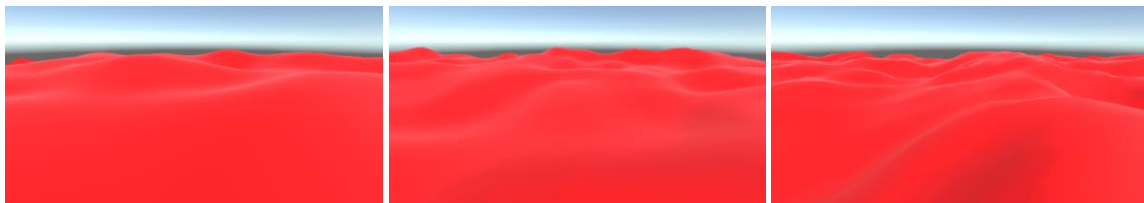


Left: Kinks Present before formula is applied. Right: Kinks removed with formula

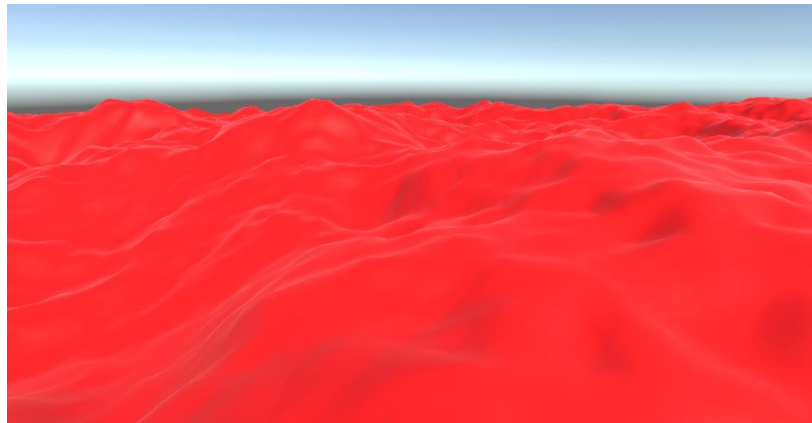
The result from this is a smooth terrain with random points being created relative to their neighbors, however, this is too smooth when compared to a natural terrain. In order to add more detail, I used the concept of adding the same waveform on to its self, but with decreasing amplitudes and increasing frequencies.



Proof of Concept: 3 Sine waves, for each, amplitude is halved, while frequency is doubled. Last image shows result if frequency is approximately doubled, result is less repetitive.



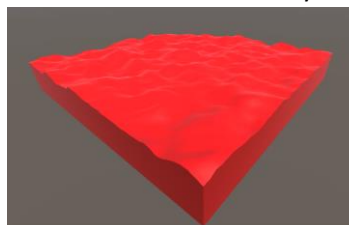
Left: 1 Iteration, SubDiv=50. Middle: 2 Iterations, SubDiv=50. Right: 3 Iterations, SubDiv=50.



10iterations, SubDiv=255

From here, I simply just need to tweak four values: starting amplitude, starting frequency, plane size and subdivision count to get my terrain working looking how I want it.

In addition to creating the random noise terrain, to make the terrain fit into the world more, I added walls to the edge of the terrain to a certain specified height, this made it look more natural in the scene as it could be a tangible object as it's not infinitesimally thin like before.

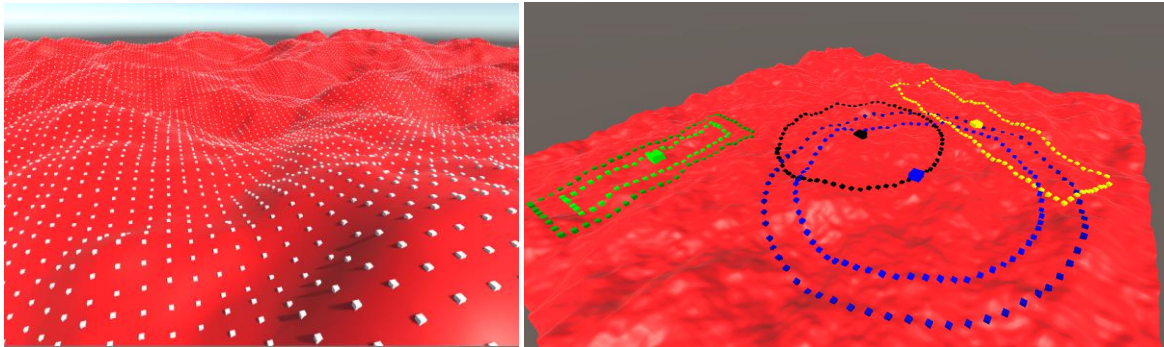


Walls added to side of terrain to make it look like nicer.

1.2 Adapting terrain height suitable for object placement

I knew that before starting, I should create helper functions that can convert between the terrain vertices and world coordinates, to test that, I placed a cube at each vertex. From there, I created classes for circles and rectangles, which will be the basic shape of the terrain sculpting (with a fade

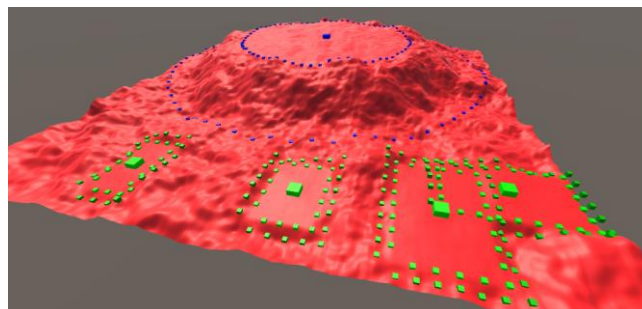
range on both too). To visualise these I created helper functions that will place many boxes to mark out the shapes.



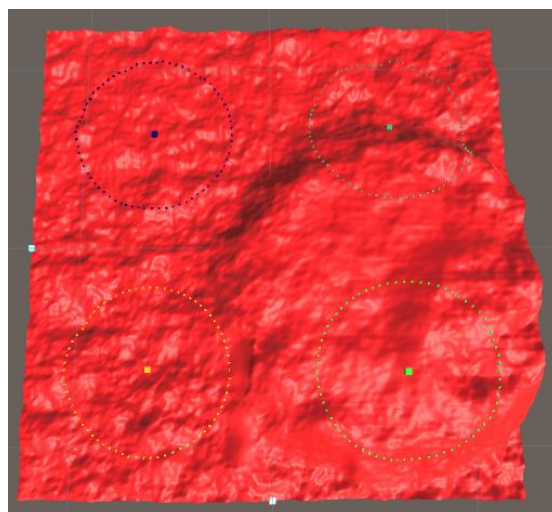
Left: Testing the conversion of verts to coordinates. Right: shape visualizer helper functions

Using the newly defined shape objects I created a class called selection, this class will scan for vertices within the shape bounding box and add it to the selection. Vertices within the fade zone will have a selection intensity based upon how close they're to the inner radius. Additionally, selections can be added together, where intensities of both selections will be added (maximum intensity 1), this allows be to create selections based upon many shapes added together; the same can be done with selection subtraction.

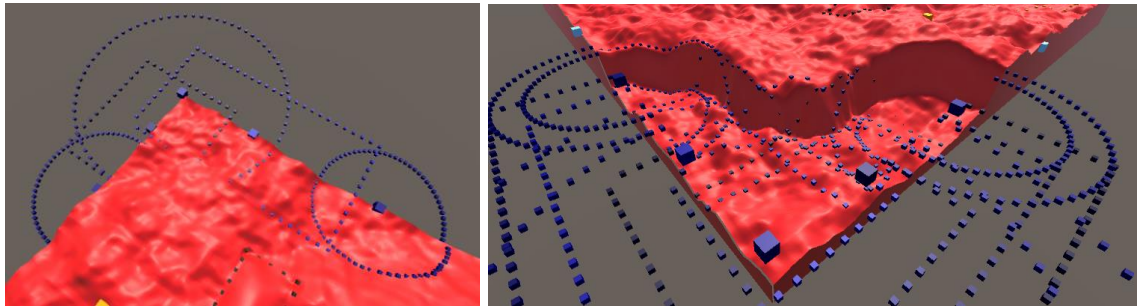
From there I continued to make tools that can use the selection, firstly I made a tool that will change the height of the selection, next I made a tool that will average the selection, for flatter land, within the selection region with fade values to ease the transition.



Raised hill, partially flattened top, also flattened squares that could represent farmland.



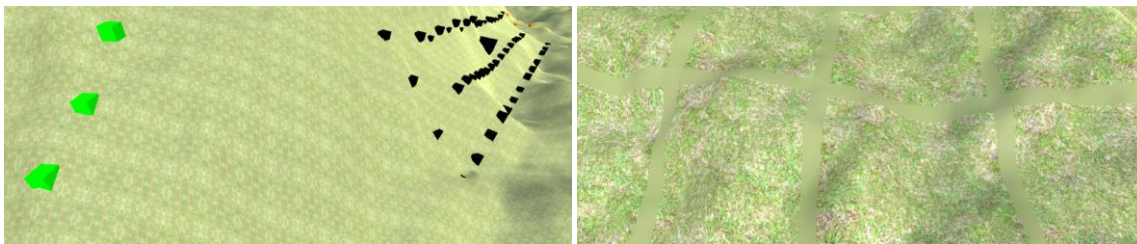
Generated map randomly split up into 4 quadrants: shore area, forest, farmland and hill/village



Left: Marking out shoreline with 2 rectangles and three circles. Right: Terrain lowered around generated region (with fade region)

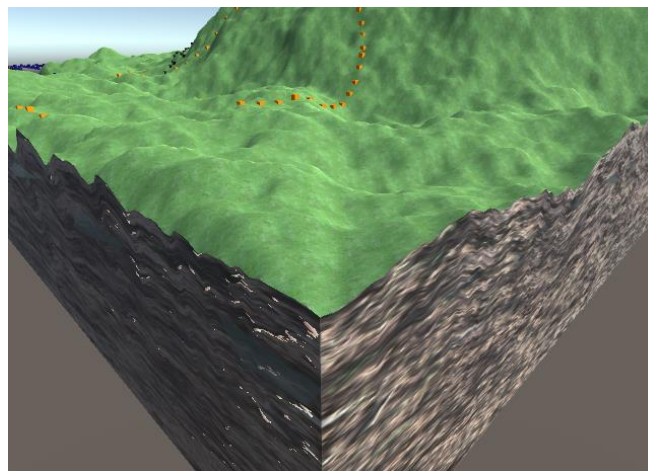
1.3 Texturing the terrain

As the terrain has been created with a plane object, I've had to assign the UV coordinates myself, at first I tried a few different variations on the tiling code, creating effects as seen below.



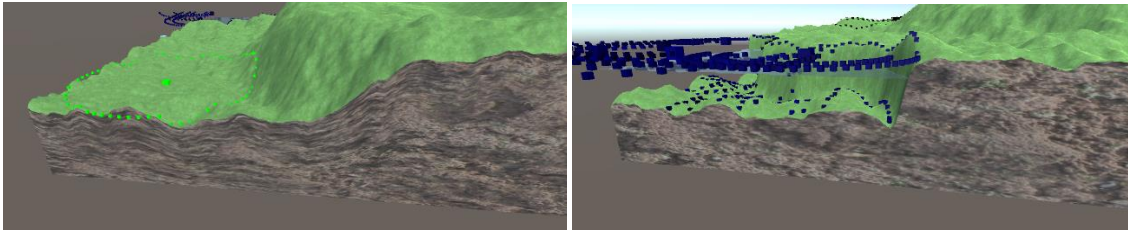
Left: One Texture per tile, reflected on even tiles. Right: Tiles enlarged across many tiles, but seam showing after each texture iteration.

After some tweaking I eventually got code that would allow me to specify how many tiles a texture should be displayed across, I set this value to 4 in the example below, I also tinted the colour to green to make it look nicer. I also applied a mud texture to the 'terrain walls', this also has a normal map to it.



Grass and mud texture applied

After looking at the sides of the terrain walls, I noticed the texture is stretched to the height of the tile segment, which created a weird effect, scaling the UV cords relative to the height of the tile fixes this. However, after 'fixing' this, I realised that the previous effect looked a lot more like the layers of crust on the earth's surface when compared to the 'fixed' version, so I reverted those changes.

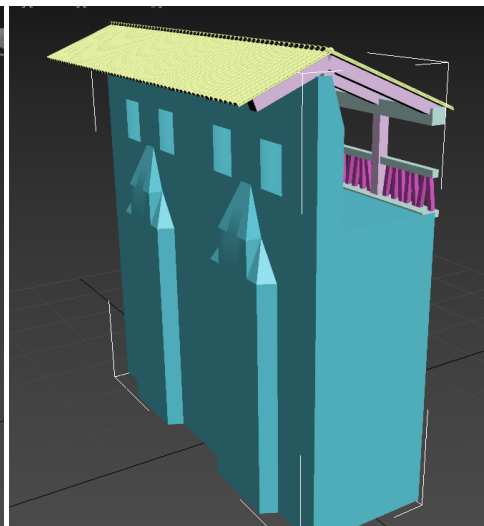
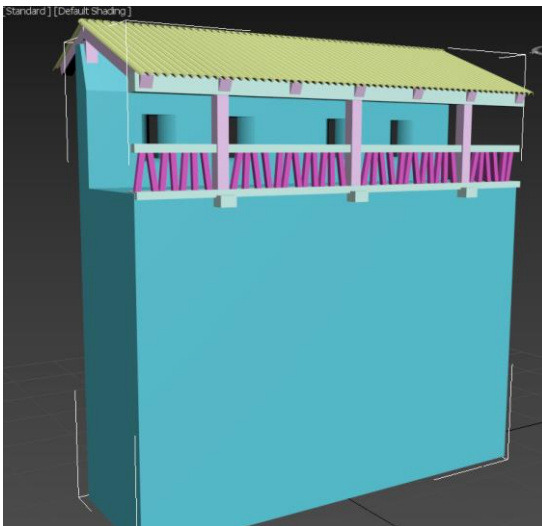
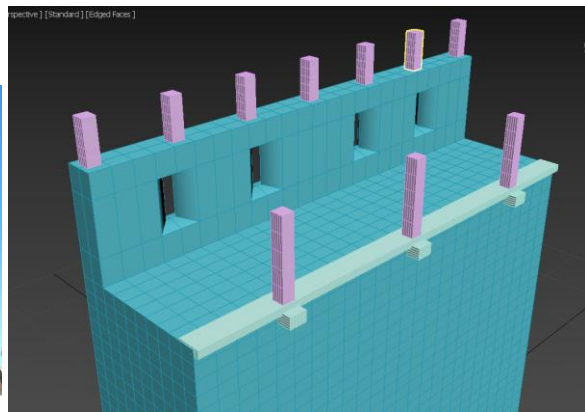


Left: Before, Right: After ('fixed')

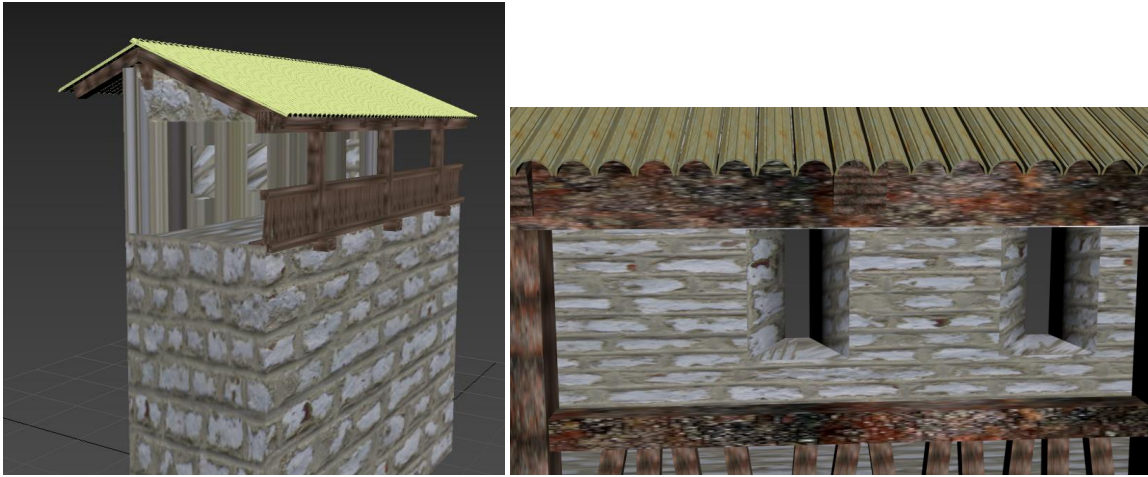
1.4 The city wall

To design the wall, I thought I'd make it into segments, a wall segment that is a walkway that connect the towers on the corners together, this way the walls can connect in a straight line, allowing the prefabs to connect together, and then only make a change in wall direction when it gets to a corner prefab section.

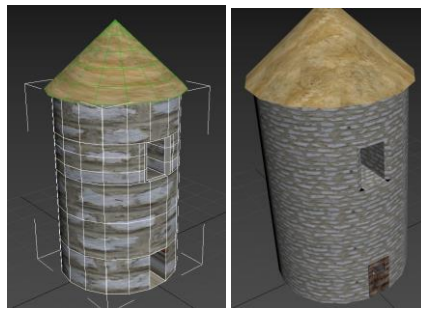
First I used 3dx max to develop the wall segment, I took inspiration from this from the castle wall in Tallinn Estonia, where I have visited previously.



I never knew what the opposite side of the wall looked like, so I added some creative detail to make it look less flat. From there I then removed many of the unnecessary points and lines within the model to aid performance. I then had to adjust some of the UV mapping to allow the textures to be applied to the model in the correct way.

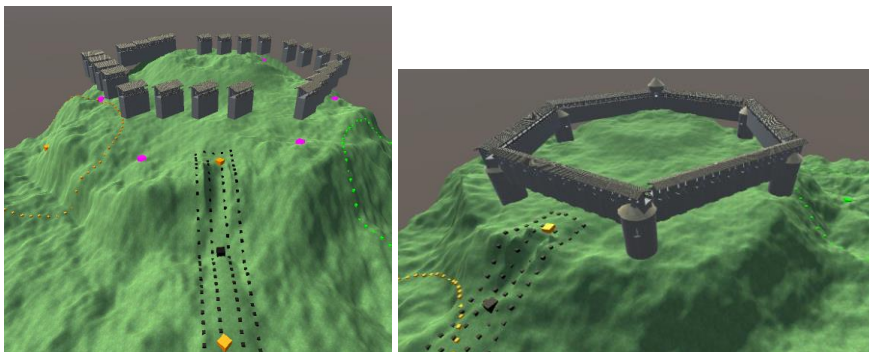


Left: Textures applied with default UV mapping (broken in places). Right: UV mapping fixed on back wall and wooden beams



Development of corner tower

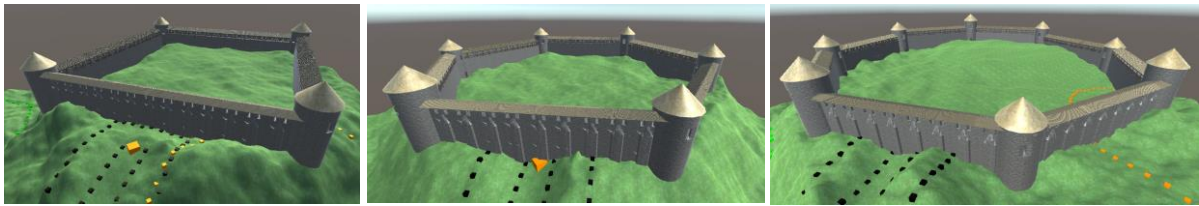
From there I tried to arrange the wall segments into a hexagon wall configuration (with the exit being in between the mid-point of the two points of the hexagon).



Left: First attempt at trying to place the walls, Right: Next attempt

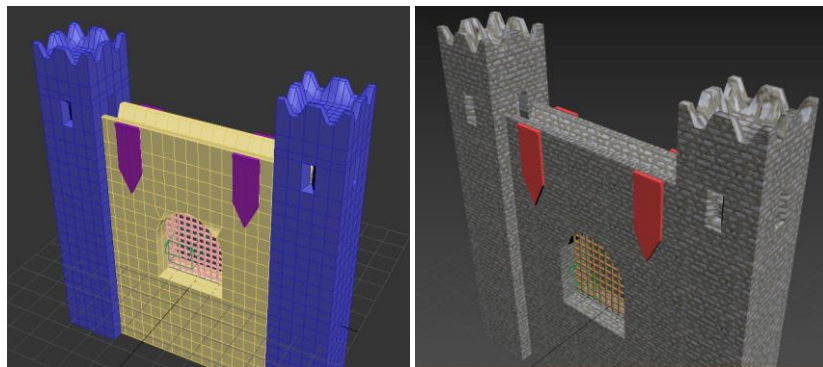
After my second attempt, I made the walls and towers face outwards by computing the normal vector along that 2D line and using its rotation. I realised a few problems; one is the height of the walls and the wall corners, to solve this I got another version of the object (just the wall mesh, not extra features like a roof) and offset this below. The towers appear at their respective heights and the walls between them are set at the average of the two heights). Next problem was the fact that sometimes there is a gap between a wall and a tower, this is because the distance is a float figure and I'm trying to fit an integer number of walls between them and so it doesn't always divide perfectly. So to solve this I checked if another one could be placed half a wall distance away, making it overlap with the other section exactly halfway along.

From there, I added a random for the shape of the walls, it can be a square, pentagon, hexagon, heptagon or octagon.



Left: Square walls. Middle: Hexagon walls. Right: Octagon walls

From there I designed an entrance that is the width of two wall segments, with additional towers at the sides, it consists of a portcullis at the center entrance. When developing it, I added additional wall that goes down into the terrain so no matter how much the terrain changes, it will look ok.



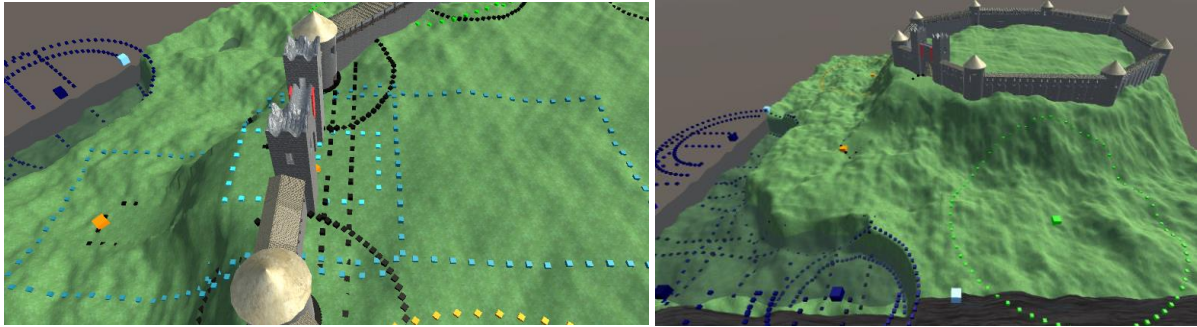
Development of Entrance

From there I placed it into the world next to where the path starts, to do this I had to remove 2 walls segments from the center of the wall for it to take its place.



Entrance added to the scene

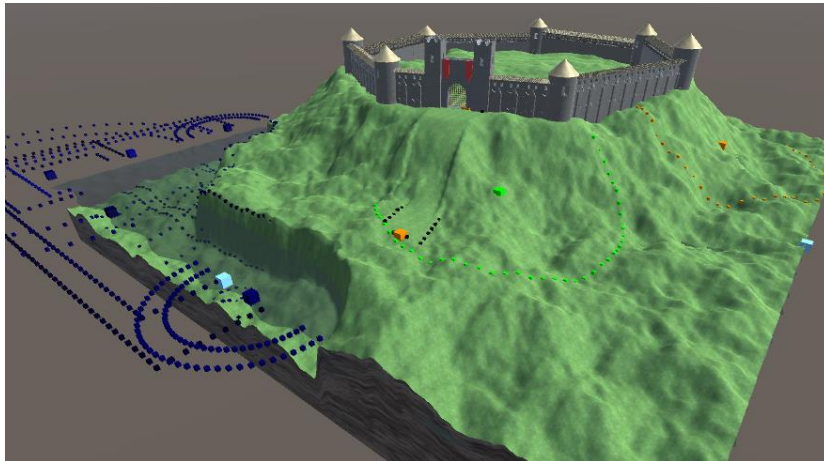
At this point, the walls appeared correctly, but the ground that it's placed on wasn't always even and the height of the main entrance put the doors for the side towers was sometimes incorrect. Not only that but there wasn't raised terrain in front of the main gate to act as a pathway. To solve this, various selections were made, and from that smoothed, raised and leveled out.



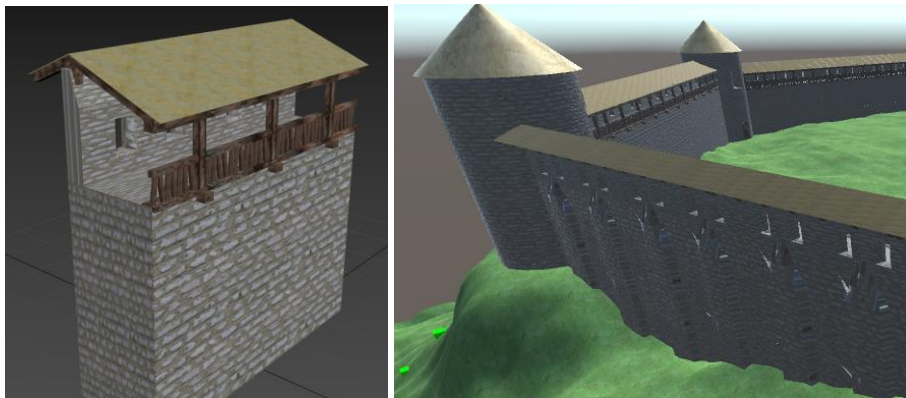
Left: Markers being laid out for making walkway. Right: Terrain adapted, markers hidden.

From there, I noticed that the terrain looked fairly repetitive every time it was generated; this was because the shore and the hill always appeared on opposite corners to prevent the shore touching the hillside.

To solve this, I checked which quadrant the hill was in, and if it was fairly equally spread in both X and Z axes, then the shore would appear opposite, however if it had a bias in either X or Z, then it could allow the shore to be in an adjacent quadrant without the shore touching the hill side. Additionally, when this happens, the shore will only extend further in the direction away from the hill.



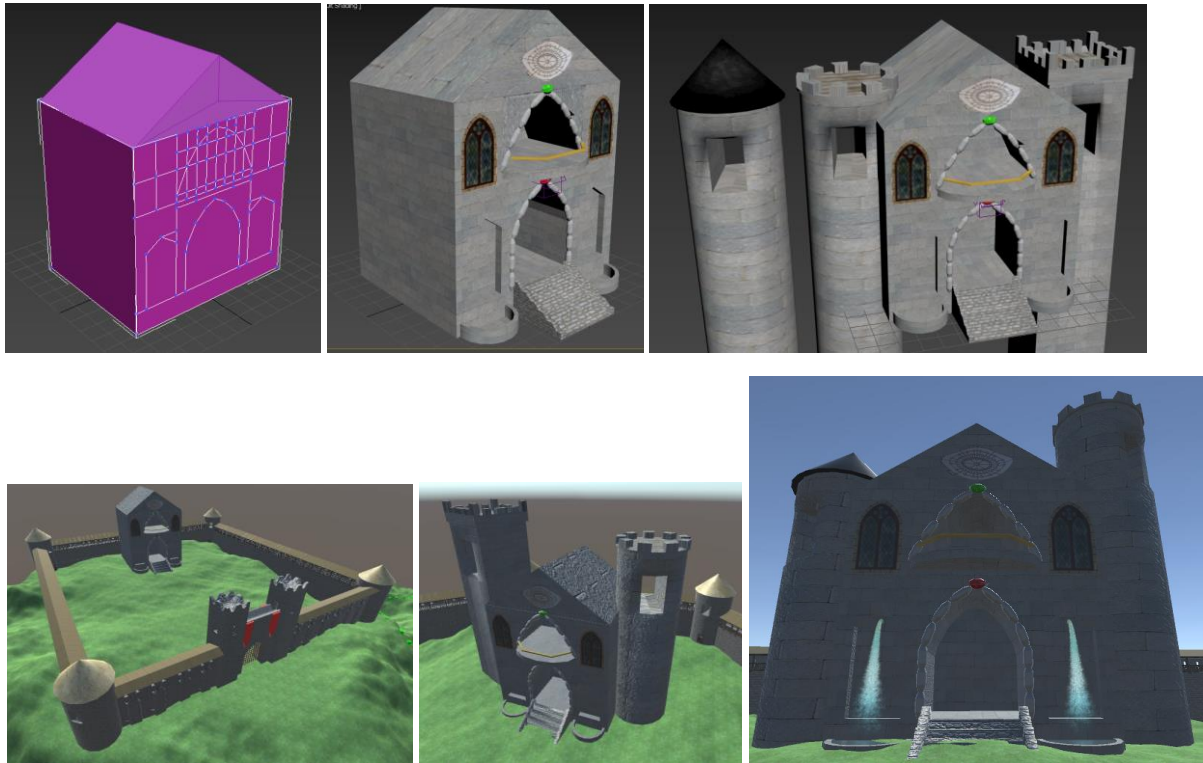
Shore appearing in adjacent quadrant to the hill, extending shoreline in direction away from hill.



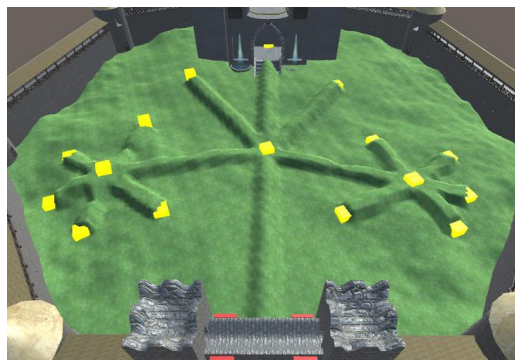
Low poly alternative for the wall roof, to improve performance

1.5 Castle, Inner-City Pathway, Houses and Structures

I then started to work on a castle, made up of a center piece and two towers along the sides. For the castle, I wanted it to randomly adapt the two side towers, there is 3 variants that it can pick for each side making a total of unique 9 combinations. Additionally, for each of the towers spawned, their height and orientation is randomly selected too. From there I added pools of water to appear either side of the castle and water particle emitters to act as a water fountain.




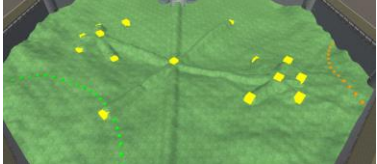
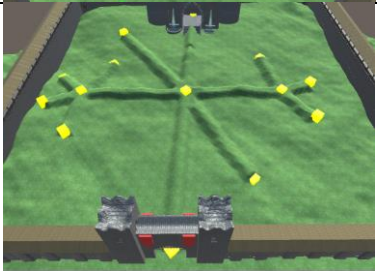
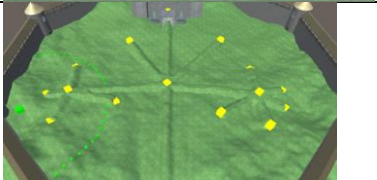



From there, I went on to develop pathways for the city, this was created by drawing a line from between entrance and castle, and from the center point, branch out two pathways to either side, of opposing angles. One of each of the pathways will have a larger angle difference to directly left or right of the mid-point. The pathways that have an angle closer to left or right of the center point branch off into smaller points; these branches are shorter when they're directed closer to the edge of the wall, stopping it from going over. The length of the pathways is dependent on the hill size to ensure it always fits in, and to make it use the space optimally.

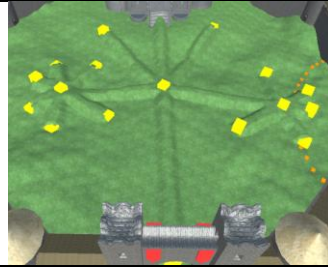


Trying to ensure that the pathway is always within bounds and not too far from the bounds, I tried out replacing random values for either their maximum or minimum values to test the most extreme

cases possible. Path size Range: (325 to 375 + 200 to 275), Mountain size range: (mapSize * 0.25 to 0.325), number of walls range: (4 to 8).

| | | |
|--|--|--|
| <p>Test: Largest path size, Smallest mountain size, Smallest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Largest path size, Smallest mountain size, Largest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Smallest path size, Largest mountain size, Smallest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Smallest path size, Largest mountain size, Largest number of walls</p> <p>Passed! More space could be utilized.</p> |  | |
| <p>Test: Largest path size, Largest mountain size, Smallest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Largest path size, Largest mountain size, Largest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Smallest path size, Smallest mountain size, Smallest number of walls</p> <p>Passed!</p> |  | |
| <p>Test: Smallest path size, Smallest mountain size, Largest number of walls</p> | | |

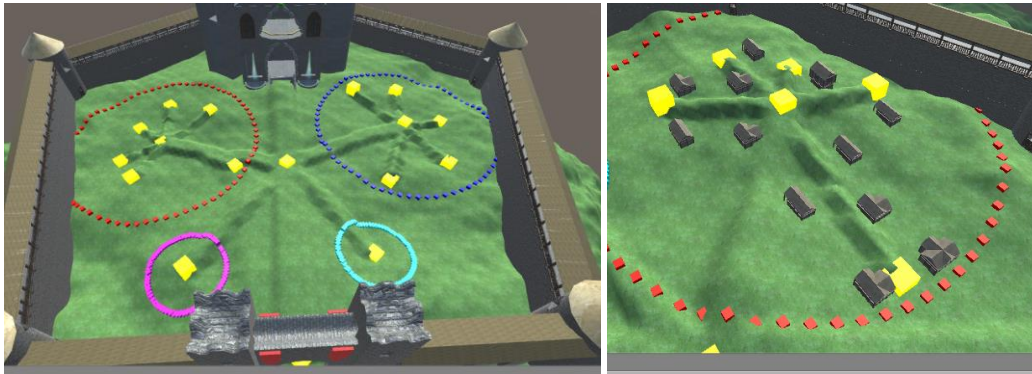
Passed!



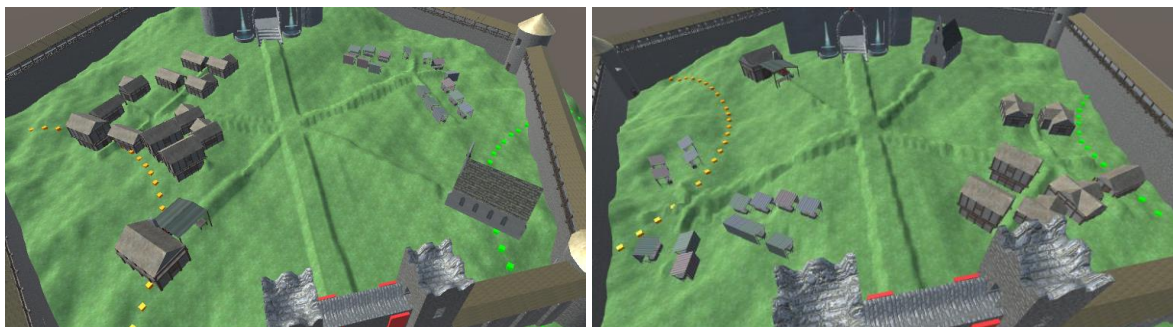
From there, I started developing some structures that could be placed within the city, and some outside the city (i.e. guard tower).

| | | |
|-----------------------------|--|--------------|
| | | |
| Development of House 1 | House 1 | House 2 |
| | | |
| House 3 | House 4 | Blacksmith |
| | | |
| Development of Church | Church | Guard Tower |
| | | |
| Development of market stall | Obtaining the texture from an image found online using photoshop | Market Stall |

From there it was time to place the various structure in certain places, this was to have the houses appear in either the left or right branching section, and the market stalls in the opposite one. From there, one of the center branches without attached sub-branches (a lone branch) will have the church, while the other will have a blacksmith. These have been marked out in the section below, the radius of the house and stall spawns is the length of the largest sub-branch, plus some padding.

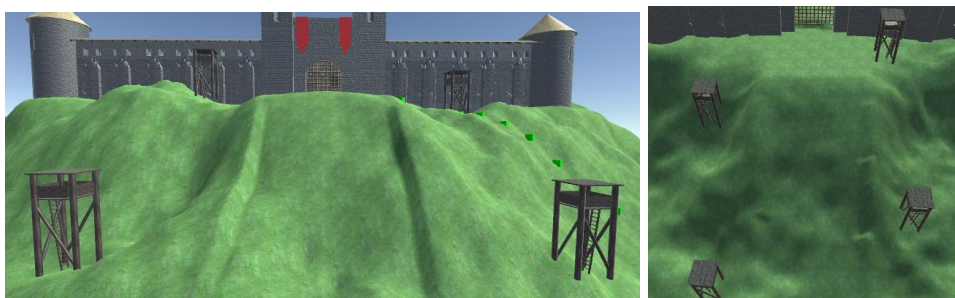


The way in which the placement algorithm works, is along each of the paths, a rectangle section is made twice, one with a larger width than the other. A selection is made with my selection mechanism for the larger rectangle, subtracting a selection of all of the neighbouring path's smaller rectangles; this leaves a series of points within the selection nearby, but not on a pathway. As an additional measure, whenever a house or stall is placed, it checks if it's within a certain distance from other structures, and from the centre of the path splitting. From there I needed to orient the houses and stalls, so I used the cross product of the path direction and up, to get the perpendicular vector, and from there used inverse tan to find the angle.



Some examples of houses, stalls, the church and blacksmith being spawned

Guard towers were placed outside the wall, at random distances from the entrance in both forward and sideways directions, staying clear of the path way, and orienting the internal ladder towards the pathway.



1.6 Forest and Flora

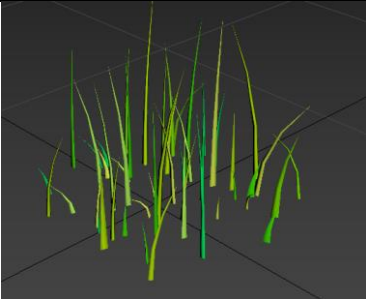
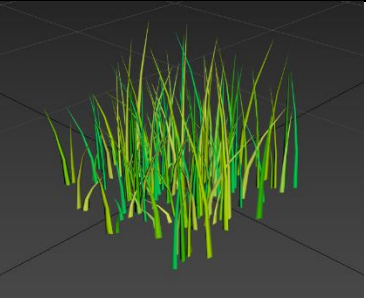
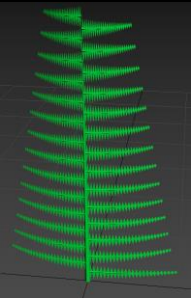
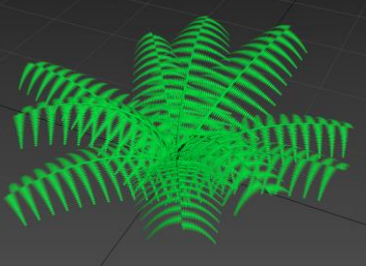

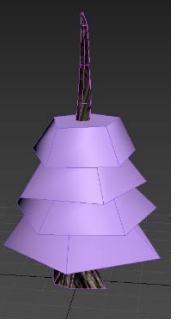
Flora will consist of 3 different types, this will be forest, farmland and inner city small flora (except for a tree or two). The farmland should be placed on flatland, while a forest doesn't matter if it's on a slope.

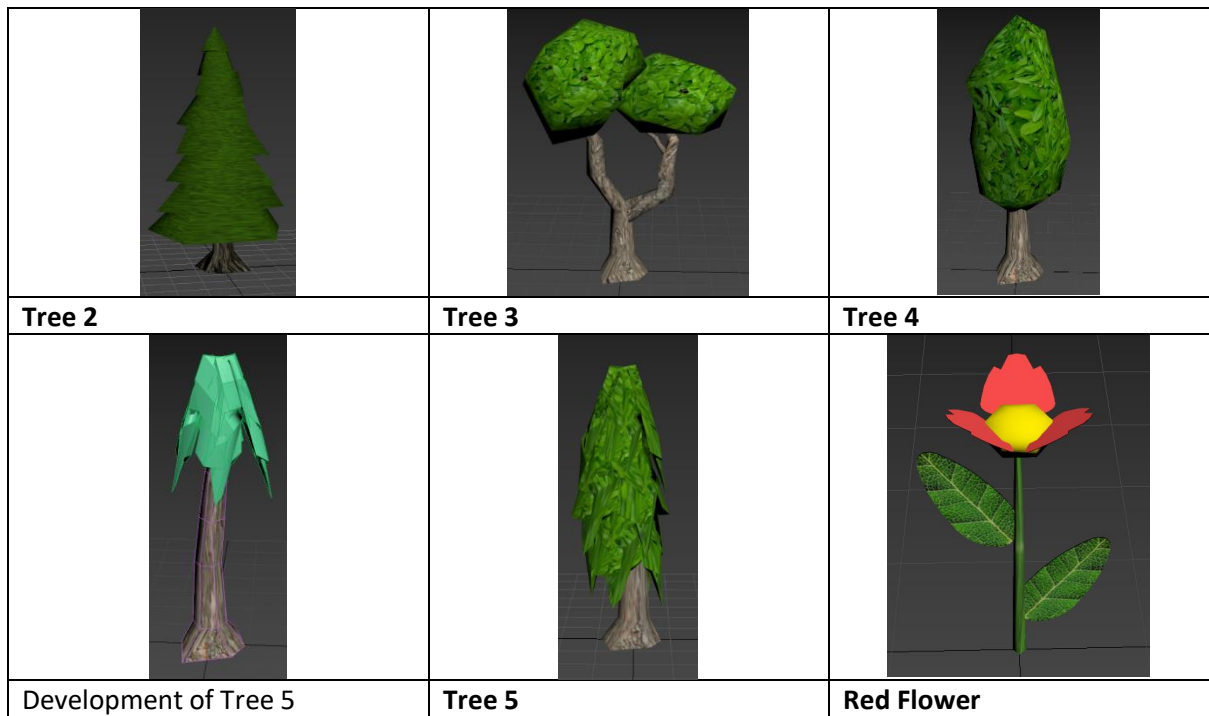
To combat this, I wrote an algorithm that checks 5 areas within the forest quadrant and 5 areas in the farmland quadrant. For each I checked all points and performed a variance calculation for each Y value, this would tell me which area is flattest, and if the other quadrant has the flattest land, then the quadrants are swapped.



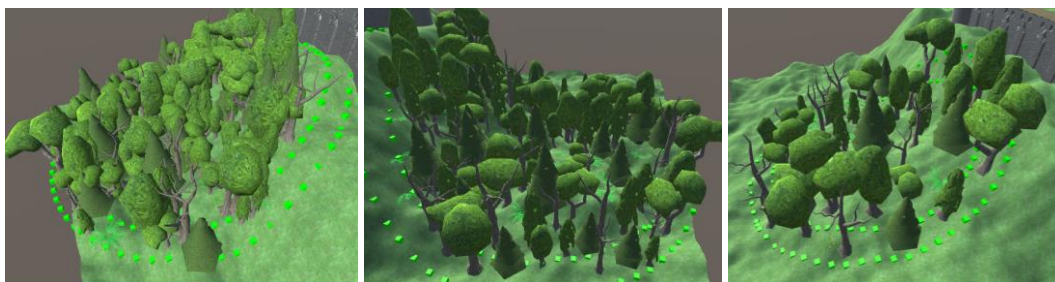
Example of farmland (orange Circle) being picked for its flatter land compared to forest (green circle)

From there, I developed some flora

| | | |
|---|--|---|
|  |  |  |
| Grass (less) | Grass (lots) | Development of Fern |
|  |  |  |
| Fern | Tree 1 (Dead) | Development of Tree 2 |

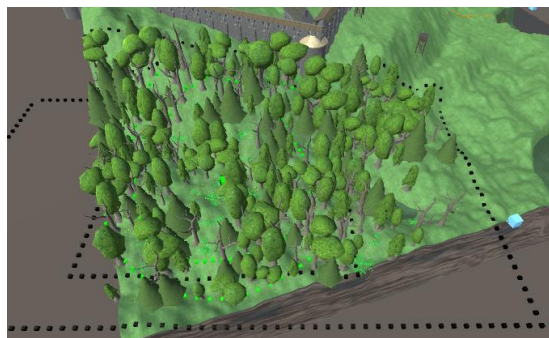


To place the trees & flora I used a similar algorithm as what was used to place the houses and stalls, for each of the terrain vertices, it checks if it's nearby to another flora, to see if it can be placed. The differences with this version are as there's a fade boundary, this is used as a probability of placement which can be set to decrease either linearly or exponentially. Additionally, different types of flora can have a different gap distance.



Tree gap = 30, 50, 80

From there, I made modifications to the selection, adding a large rectangle perpendicular to the vector between the forest center and the hill center, with a fade on to 'fade' the trees out. I additional placement checks ensuring it will not spawn within the city, within the assigned farmland area or within the water.



Across the rest of the terrain, small patches of grass and the 'odd tree' are placed across the landscape.



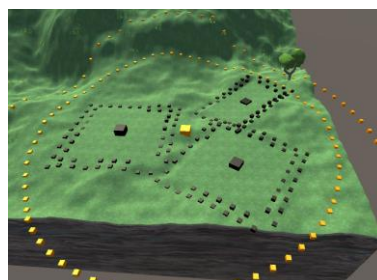
As for the flowers, they are randomly placed alongside the center pathway.



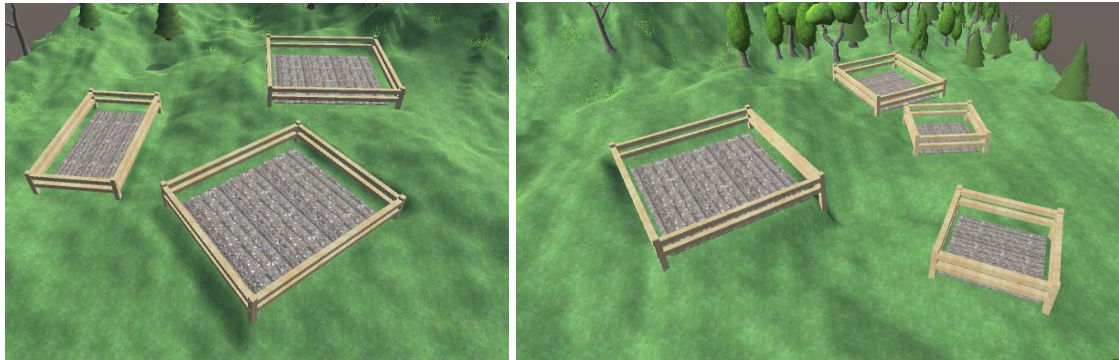
1.7 Farmland

| | | |
|---------------------------------------|--|---|
| | | |
| <p>Created fence bars to go along</p> | <p>Created fence post to go vertically</p> | <p>Photoshopped texture smaller, and made it a seamless texture</p> |

To allocate the farmland, I wrote an algorithm that will have X attempts to try and place a square of farmland (of random size) somewhere within the circle so that it doesn't collide with any other piece of farmland. As the algorithm, might be lucky or unlucky given the number of attempts and this will vary the amount of soil farmland being generated.



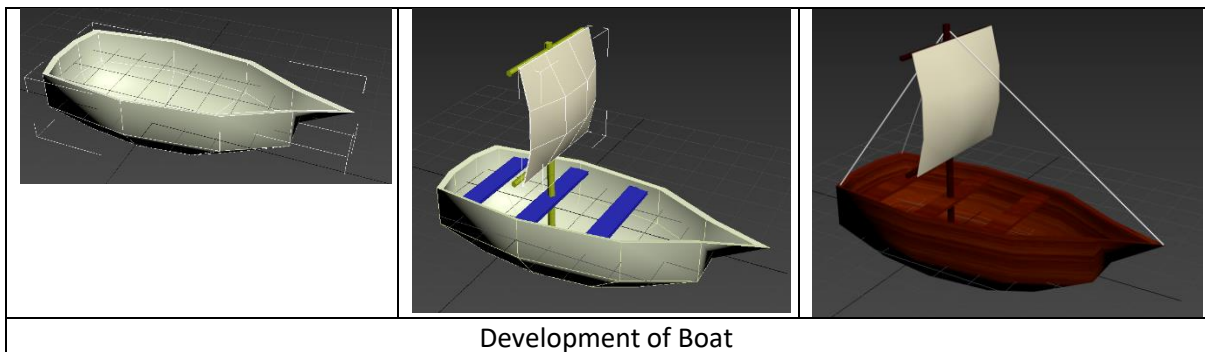
Farmland being selected, and flattened in the area.



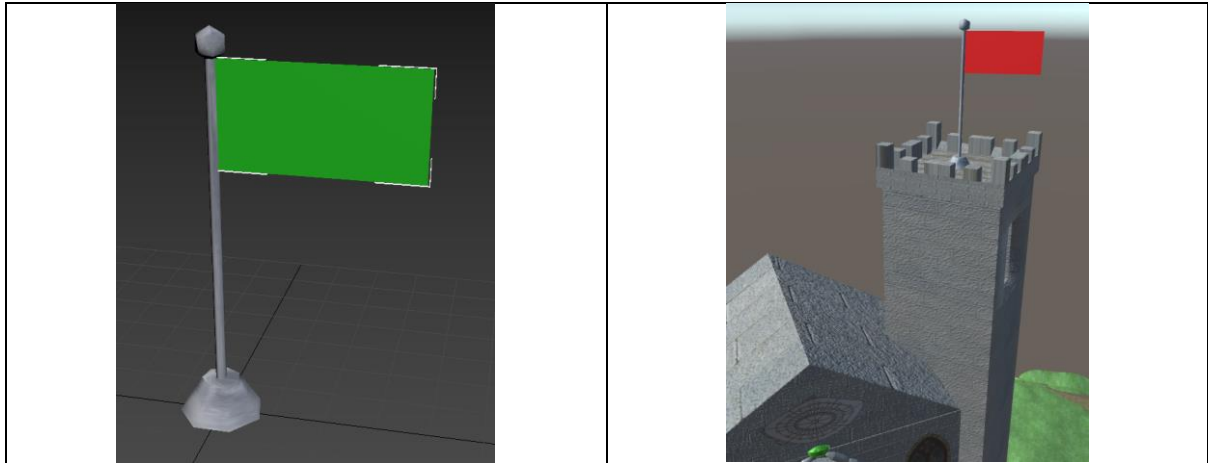
Example of farmlands being generated

1.8 Finishing Touches

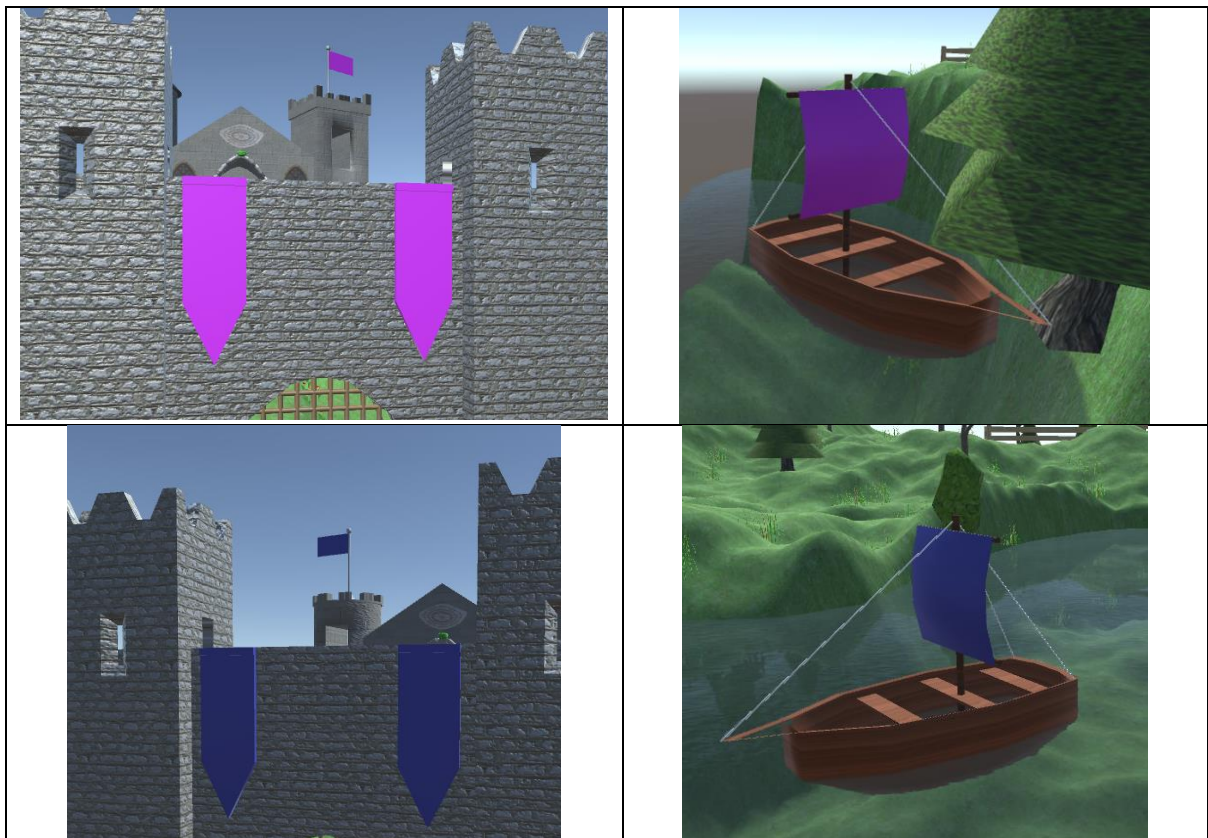
Next I modelled a boat that could appear at the shore. When it spawns it, it will 'bob' up and down on the water, whilst slowly rotating.



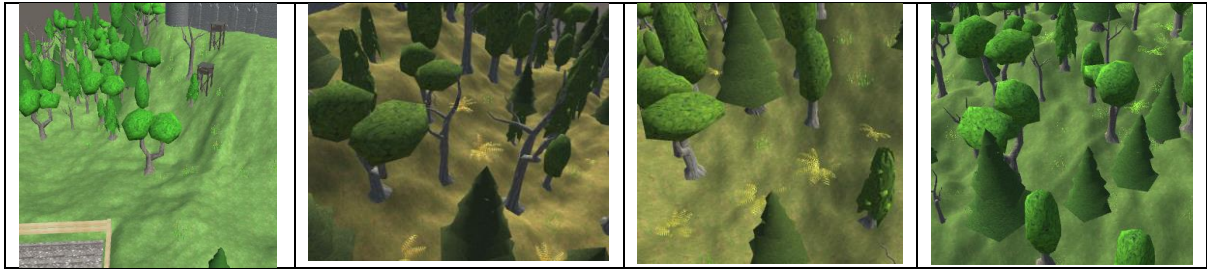
Next I added a flag pole that can be placed on top of the castle tower (if it's not the variant with a pointed roof).



Next I made the program randomly pick a 'colourful' colour that represents the kingdom, this is reflected on the: flag on a pole, the banners on the entrance on the boat sail.



Next I added random seasonal colouring to the grass and leaf materials, example can be seen below. To make the colour system realistic, a hue closer to orange, had to mean a lower colour value, that is darker.



1.9 Final Testing of Maximum and Minimum Random Conditions.

1.9.1 Testing Forest Placement

To test if the forest will always appear correctly it necessary to test only the **largest** forest size, with **varying placement** and the **largest** hill size. There is no need to test small forest as that couldn't cause any problems as the only potential problem is too many trees appear where they're not supposed to. Along with this to ensure that the forest isn't going to collide with the river, the **largest** river size should be chosen too. The only needed variable is the placement of the larger forest.

| | | |
|---|--|--|
| <p>Forest is randomly placed</p> <p>Passed!</p> | | |
| <p>Forest is randomly placed</p> <p>Passed!</p> | | |
| <p>Forest is placed as close to city as allowed</p> <p>Passed!</p> | | |

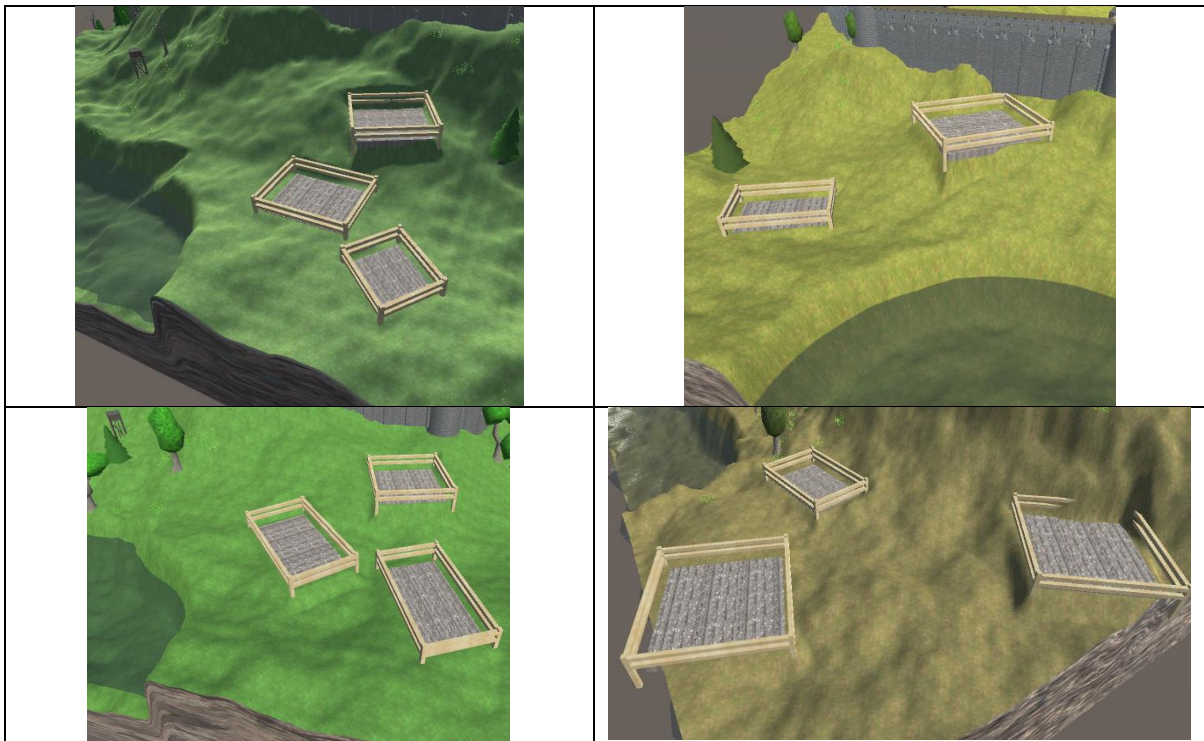
Forest is placed as close to shore as allowed

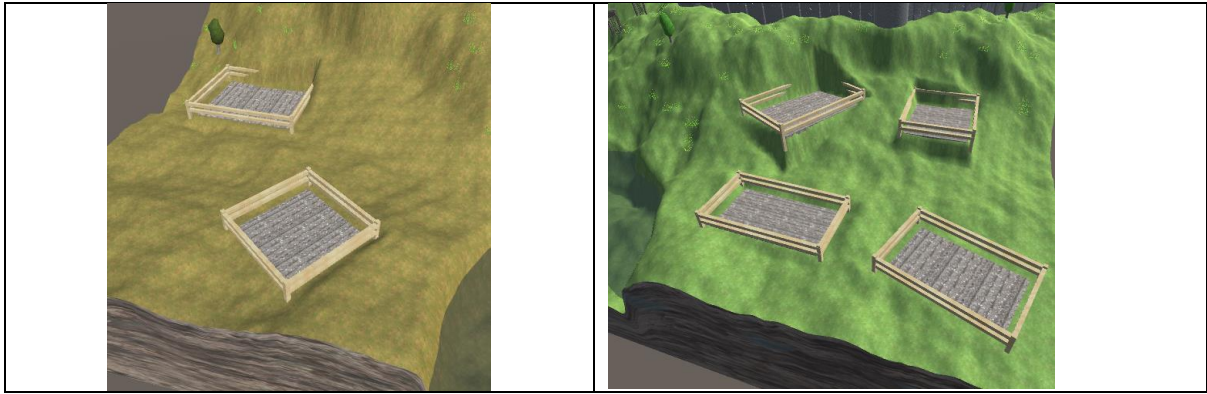
Passed! One tree is borderline, but still on land and still realistic as water levels can change.



1.9.2 Testing Farmland Placement

Farmland is placed because of calculations used to find the flattest land within the area, so I can't just test placement configurations as that's not how the algorithm works. The things to test for, for farmland is if it's on a hill, and if it's appearing in the water. Because of this, the only resulting testing strategy is to set the **hill**, **number of walls** and **shore** to be the **largest**, meaning there's the least possible flat land available, and from there, just see the outcome as it wouldn't be a fair test to place it myself.





The test results show that all of the farm lands appear as they should, some appear on fairly steep inclines, but the ground is flattened nearby, and it still an acceptable layout.

2. Reflection and Critique

Overall I feel like the project was a success, I'm extremely happy with how it turned out. As I was trying to add so many features into the world I was worried that it would make them spawn on top of each other or incorrectly as they're tightly packed, however this was not the case, due to the selection system I implemented along with algorithms to check for neighbouring structures.

The selection system I created was a must-have for this project, without it, it wouldn't have been as well executed. It was inspired by selection tools in software like photoshop, and although this is invisible when looking at the 'marching ants', they have intensity values too. Photoshop also inspired me for the uses of selections such as blurring an image (mean kernel), which for terrains makes it more smooth.

The selection system was necessary because I created my terrain with a plane object, in which I manipulated the vertices. Had I chosen to use a Unity terrain object instead, then the selection wouldn't have been necessary for adapting terrain, but still useful for marking areas to check for other nearby selected shapes. One of the drawbacks that I faced by choosing to use a plane object is that it doesn't support multiple textures per se; it does it you create a different sub-object, which is basically like having two different meshes, this also wouldn't allow for a blending of two materials either.

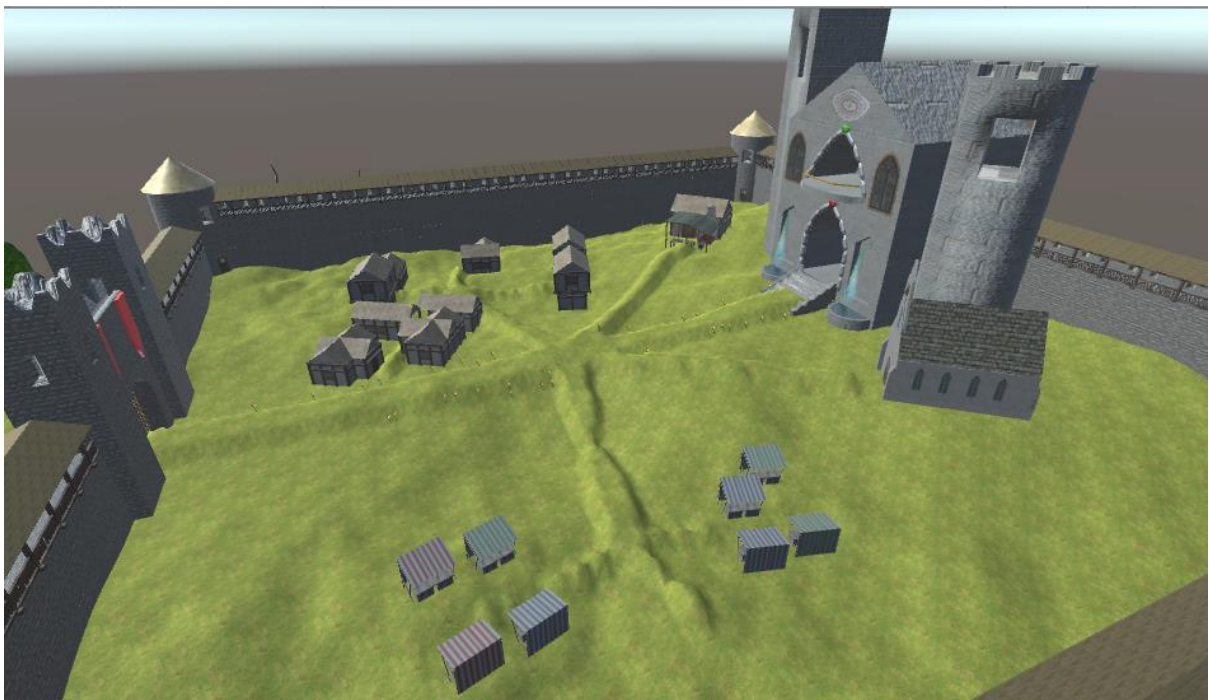
One of the main skills that I've learn from developing this project is the ability to 3D model, before this project I had little to no experience, and by the end of it after modeling so much I can create them effectively and quickly; from this I also learnt about how to apply UV mapping to a mesh. One of the lessons learnt while learning this, was to keep a lot poly count, various models were either recreated or edited because of a high poly count. Another skill acquired was understanding how to rotate objects relative to perpendicular vectors, which was used many times throughout the project.

One of the biggest challenges was trying to create the path network within the city walls that would make effective use of the randomly sized mountain, with random amounts of walls. I feel I did an acceptable job at placing the pathways, but there's room for improvement.

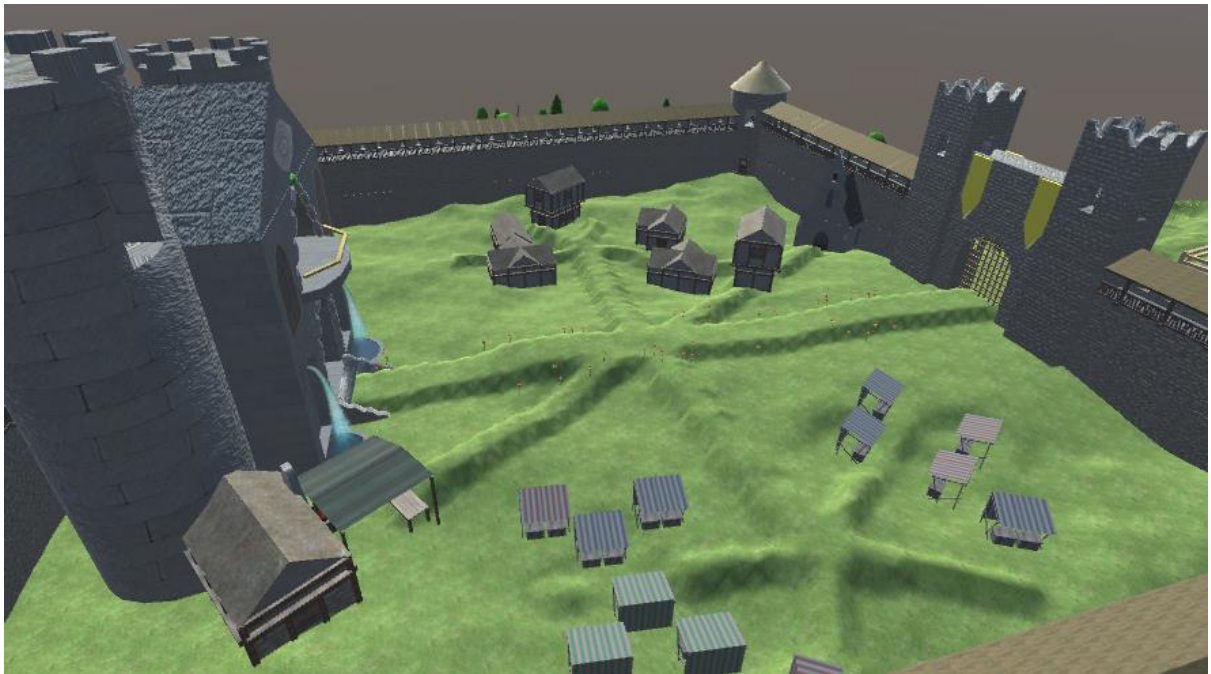
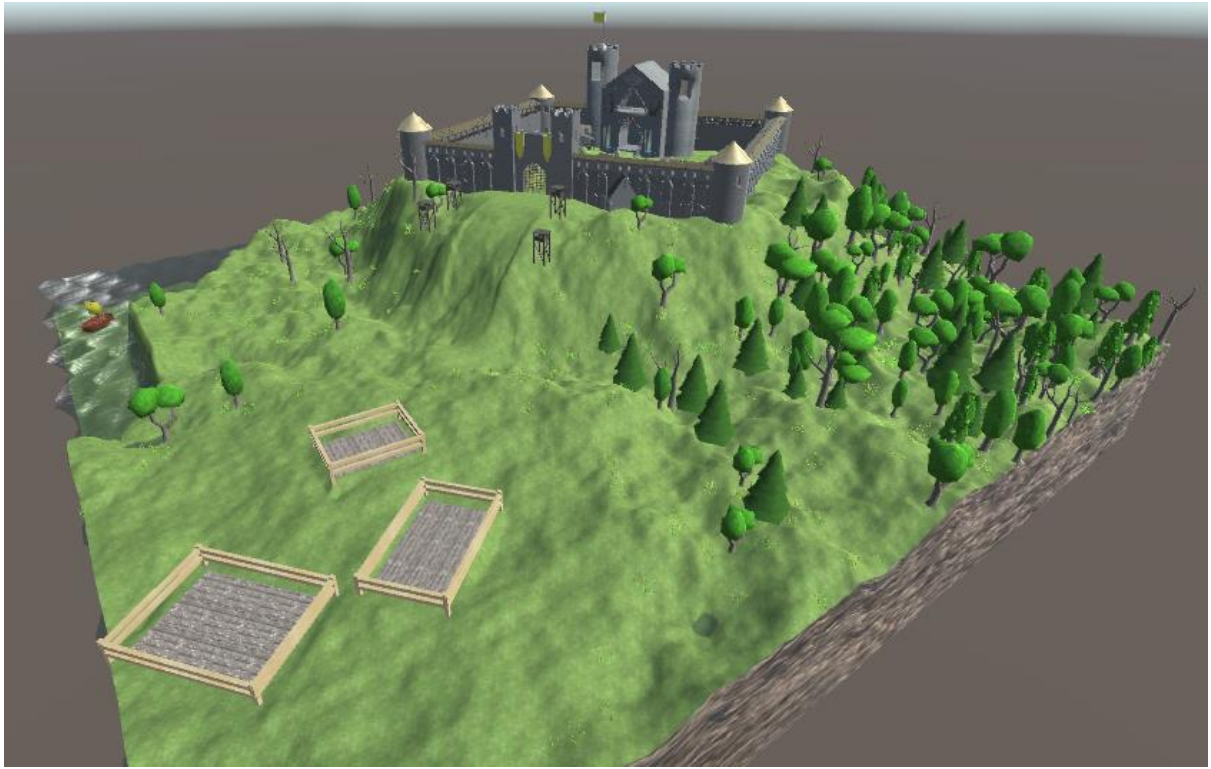
3. Screenshots of final Product

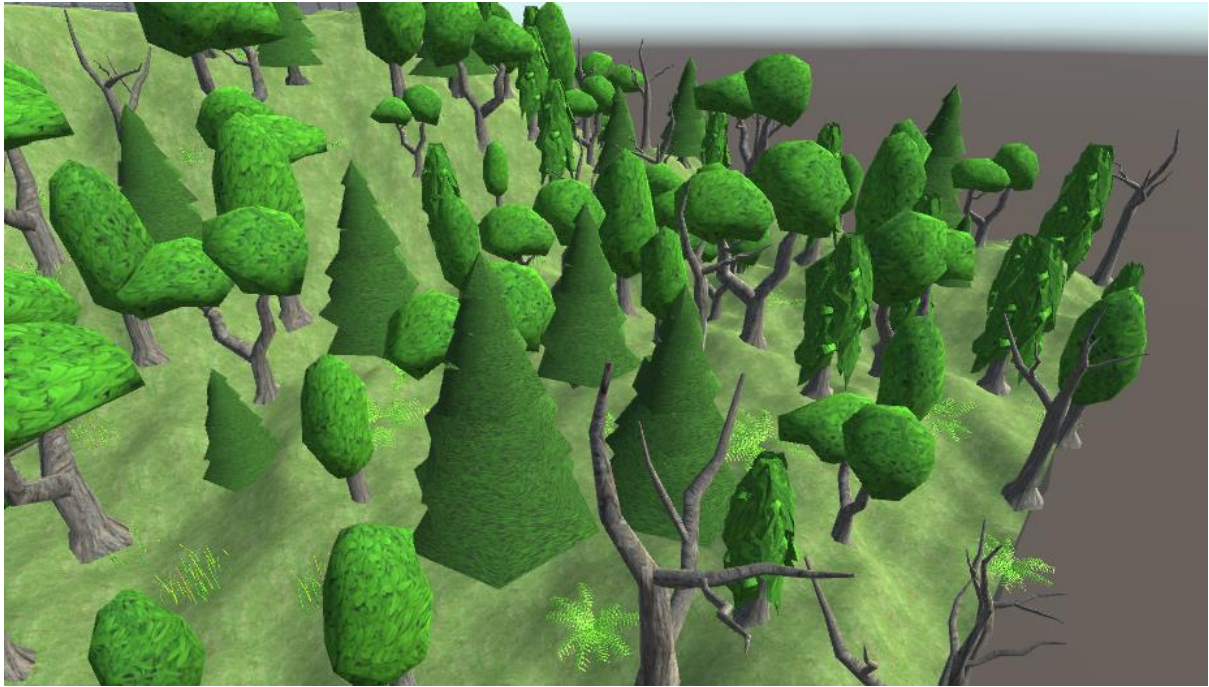


Next Generation:

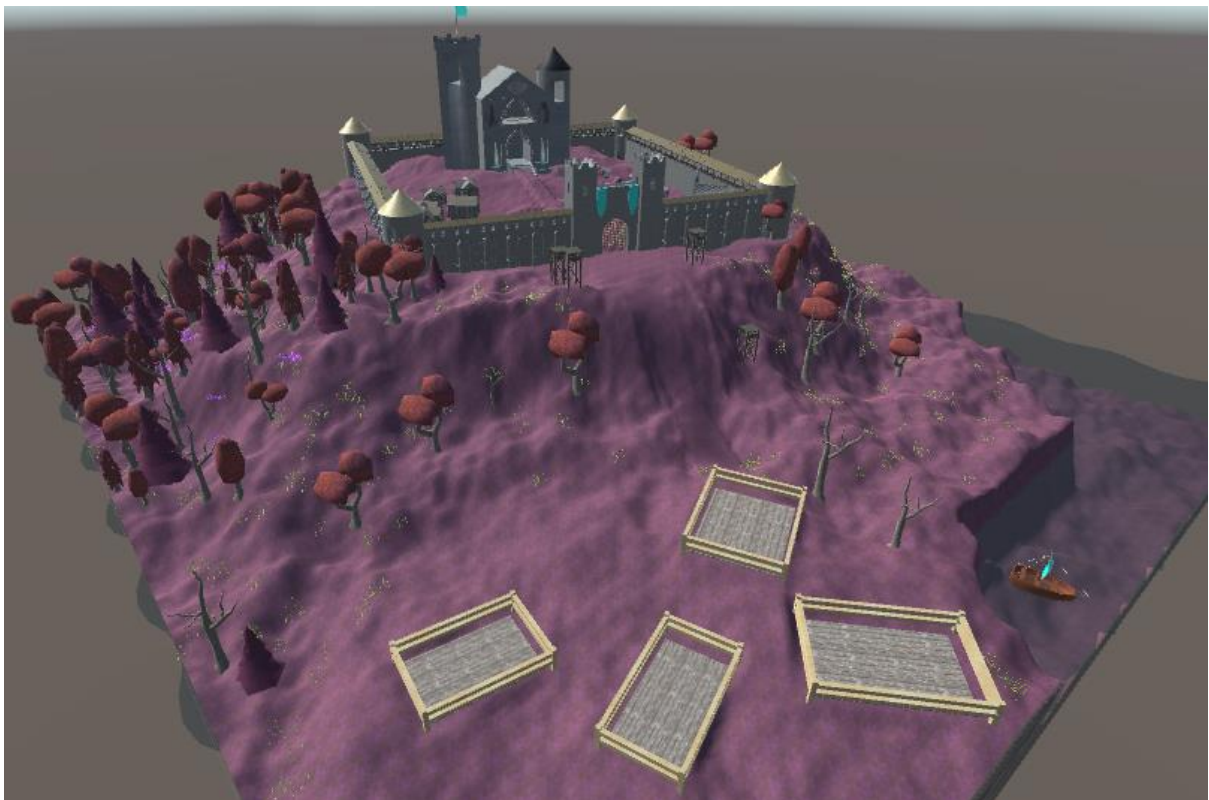


Next Generation:





Bonus Image: (Altered Colour Scheme)



4. References

4.1 Text-Based References

Biagioli, A. (2014) *Understanding Perlin Noise* [online] Available from: <http://flafla2.github.io/2014/08/09/perlinnoise.html> [Accessed 22 February 2017]

4.2 Texture/Particle Resource List

[Water from Unity Standard Assets]

<http://www.mb3d.co.uk/mb3d/Bark and Tree Seamless and Tileable High Res Textures.html>

<http://free-texture-site.blogspot.co.uk/2010/10/free-plaster-wall-texture.html> [Edited Photo]

http://previewcf.turbosquid.com/Preview/2014/08/01_17_25_29/door002.jpgb2b97fa6-42a9-443a-953a-98b45df8b74dLarger.jpg [Edited Photo]

<http://goodtextures.deviantart.com> [Edited Photo]

<https://uk.pinterest.com/pin/264868021806597337/> [Edited Photo]

http://www.artisticlinestudio.com/stained_glass_window6101.htm [Edited Photo]

http://orig06.deviantart.net/dae5/f/2008/287/8/9/black_texture_ray_by_ethenyl.jpg

<https://www.assetstore.unity3d.com/en/#!/content/48580>

http://www.cornucopia3d.com/purchase.php?item_id=6714 [Edited Photo]

<http://staticdelivery.nexusmods.com/mods/110/images/27118-2-1353337757.jpg>

http://www.lughertexture.com/images/textures/details/metals_11/new_metal_12/galvanized_metal_iron_20130603_1352605191.jpg

<http://4.bp.blogspot.com/-zCdYHYKJnxs/U8uz4Tahall/AAAAAAAAAFgM/2rCjAugAdE/s1600/Wood+tree+bark+seamless+texture.jpg>

http://previewcf.turbosquid.com/Preview/2014/08/01_15_24_10/bark1024_01th.jpgd774673f-f549-4546-b7a7-5e30e0f01f53Larger.jpg

http://texturelib.com/Textures/nature/bark/nature_bark_0014_02_preview.jpg

<https://www.filterforge.com/filters/7276.jpg>

http://img02.deviantart.net/eed6/i/2006/134/4/f/texture_leaves_by_kuschelirmel_stock.jpg

http://bgfons.com/upload/leaves_texture1233.jpg

http://1.bp.blogspot.com/-wkBDf9bCrBI/TwRPiqYM6II/AAAAAAAAAQY/CKCAoRkN4F8/s1600/light_wood_free_tiled_background_seamless_texture.jpg

<http://www.myfreetextures.com/wp-content/uploads/2014/10/texture-seamless-wood-1.jpg>

<http://www.myfreetextures.com/wp-content/uploads/2014/10/seamless-wood3.jpg>

<http://www.absba.org/archive/index.php/t-1253214.html>