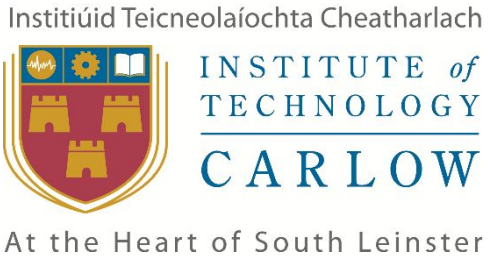


Computer Games Development Project Report Year IV

Brian O'Neill

C00213756

03/05/2020

Procedural Generation of Game Assets

Contents

Acknowledgements

I would like to thank the following people who assisted in completing this project including:

My lecturers for providing me with the knowledge over the past few years to help me tackle this project.

To my supervisor Noel O'Hara for providing incredible support and assistance throughout the year. His help throughout the year was invaluable and helped me a great deal.

Project Abstract

Motivation:

Although Procedural Generation has been around for many years, it is nowadays a particularly useful tool for indie developers and other big name developers. With distribution platforms like Steam, Epic Games launcher, Play Store and Apple's IOS Store, it's now easier to create and publish your games than ever. But the challenge for small developers is no longer distribution, the challenge is content these days. Even small indie developers can produce a vast amount of content through the use of procedural generation, with reasonably low amounts of money and manpower invested. For instance Minecraft is a good example of a small developer making it big by using procedural generation. When it was published the game took the world by storm and it hasn't slowed down since. The development team sold the IP to Microsoft for $2.5 billion dollars[Microsoft n.d]. Since then, many games from larger and smaller developers have started to incorporate these techniques into their games too. Some developers are creating whole games from procedural generation at this point and some are doing very well with this but these techniques are constantly developing and will potentially become so much more than we can imagine right now.

This is what inspired me to pursue a procedural generation thesis, as I find the entire field incredibly exciting.

My Aim:

My goal for this project is to develop an editor in which the player can use procedural generation techniques to create a piece of a game world, including a road network using a directional tool, buildings, vegetation and rocks. Showing how procedural generation will allow you to build varying game assets each time and also showing how small teams of developers may be able to use this object generation method to build highly replayable games.

Method:

I approached this topic by deciding on which engine I wanted to use.I was mainly stuck between two engines either Unreal or Unity. After researching Unreal Engine, I had learned that Unreal Engine was a fantastic engine for studios and teams of people but as I am working alone on this project I decided to use Unity which is a much more solo developer friendly engine. So using this information I decided to use unity for visualizing my project.

After that I decided to look at games that used these techniques and after looking through several different games and genres I found that many games were using procedural techniques to build their game terrains. With this knowledge, I explored more ways of using procedural generation for other parts of a game world. Since it seems to be a very common asset to use procedural generation in.

As I continued other games I realized that SimCity creates its roads using procedural methods, Creating roads is definitely done less in the industry versus creating terrains. Ithen

began to think how I would go about creating something like this in my project. I had created a test for this by making a very simple way of making a curve and then adding points to the end of it to try to simulate the method used in SimCity, However I ultimately scrapped this idea as I began to see through the use of this method that it is not the feel I was looking interested in for my project. However I kept one of the techniques used and that one was the directional tool for creating the road in a particular direction.

After this testing and seeing the way it felt using that method, I decided to use a tile-based method and upon testing this is exactly what I was looking for.

Then I started looking at other possible items that I could bring into the scene to enhance the look and overall appearance of the environment and I came to the conclusion after studying various technologies that are commonly used in industry. One of the technologies I came across was SpeedTree, which is a well known tree generator and I decided that trees would be the perfect addition. AS the more objects that I can create and the more diversity in my world the better.

So I began looking into possible methods I could use and one jumped out at me and after further research into it I realised using this same method I could create grass and this was perfect, So I ultimately decided to pursue this method.

However the placement of the trees around the map would be very important as you don't want them to just spawn randomly throughout the scene and so I to look into possible solutions to this and one method I found was Poisson Disk Sampling which seemed to me to be an effective method at solving the case that I had previously mentioned.

I had learned in previous college years that perlin noise was a procedural method used to create believable terrains. I wanted to implement this inside my editor but I did not want to use it for a map so I decided that maybe using it to create an object would work well. So I began to think which objects would work well with this method and one that would be possible for me to use came to mind and that object was rocks since they can be jagged and are very rarely smooth this jaggedness could work well with perlin noise.

Then I decided that to add to the assets I have already mentioned, I believe that creating buildings alongside the road as you move the road building tool would be a worthwhile addition to the project. This allows you to create an interesting stretch of road with all these assets being created around it.

I strongly believe that with these I can show just how procedural generation of game assets is a worthwhile practice in video games.

Project Introduction and Research Question

“Procedural Generation of Game Assets”

I'm developing an editor using Unity3D, My Editor will allow the player to create a small area of a game world which will be populated using both player input and also using procedurally generation to decide where to build and also how to build each of the assets.

The player input will affect things like the direction of the roads, the size and age of the trees and the spacing and density of the grass.

The reason I chose to create this project using Unity was because I really wanted to get some exposure to creating in 3D as over the last few years I have not been exposed to it a lot. The reason I chose this research topic was due to my own interest on how games such as minecraft,diablo and borderlands are able to keep a loyal audience even after 10 or more years and still surprise players by the sheer number of possibilities that are possible within these games.

I find it fascinating that even after such a long time of playing a game that new possibilities are still yet to be discovered due to their procedural generation of game content. This is the reason I believe that my project is an important example of how you can use procedural generation to create varied areas of a game world to produce interesting and varied areas everytime it is generated.

The potential impact of procedural generation is significant. It can be used industry wide to reduce costs and lower development time. The demand for the product will increase,Due to the amount of content that can be generated is basically infinite. This can be done by separating the world into chunks like minecraft where each time a new chunk is loaded it's completely new as long as that chunk has not been loaded before.

My goal for the outcome of this project is to create a tool that allows for the creation of small pieces of a game world with an editor that has integrated procedural generation techniques.

Background

Procedural Generation is the technique of creating content mostly for use in some form of media. The content is created with the use of algorithms rather than a designer or someone of a similar job role creating these assets and laying out the world by hand.These algorithms are created using a mixture of both randomized values and other values that are controlled by the user of these algorithms, this allows for enough freedom for the user to create a near infinite number of assets that all have some degree of difference between them.

Procedural generation has been especially prominent in the creation of video games since the 1970's but even before this it was being used for other work using computers. However procedural generation can be seen also in nature so in this way it has been around much longer than you would think as it can be seen all around us today . In fact most of the world around us right now is created using a natural version of procedurally generation, the trees around us no two trees are exactly the same there is always some difference. Islands created by erosion due to the sea over millions of years,Coastlines being constantly battered by waves are all examples of this.[Blatz and Korn 2017]

Literature Review

In this section, I will begin by defining procedural generation based on the existing literature to provide an understanding of the topic I will address and also give a brief history of the topic since its inception to date. The goal of this review is to explain procedural generation of game assets and then provide examples of potential algorithms and uses of each of them.

Defining Procedural Generation

Procedural generation can be described as the creation of data using algorithms rather than having a designer or programmer do the work manually.The concept of randomness is important, as this will help ensure that a large number of possible types of generated assets can be modified by changing a small few parameters. In one way, procedural generation could be seen as an algorithm that takes some data from a designer or artist to produce an object closely resembling the artist's vision but with sufficient degree of variation in mind[Halliwell 2008].Another way to describe procedural generation is to see it as a group of rules or algorithms that can create assets automatically.*'Procedural Generation is a family of algorithms and procedures used for generating contents in an automatic way rather than manually'* [Aversa 2015].While in another more simplistic way it can just be described the *'creation of data by computers'*[Van Brummelen and Chen n.d].The definition of procedural generation varies from role to role in the industry because it is such a large subject area and extends into several different areas such as AI and graphics which may adjust the concept depending on the expertise field of the users.

History of Procedural Generation

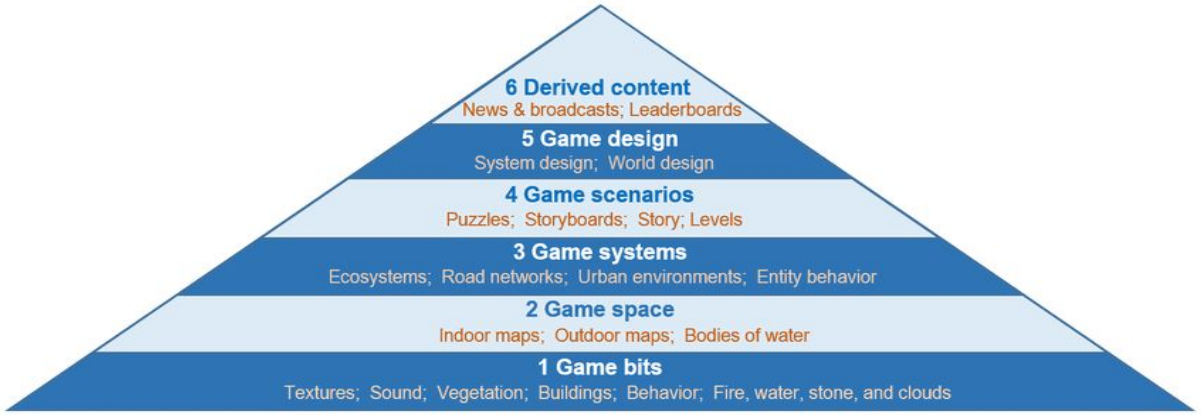
Procedural generation has been around in the games industry for almost 40 years now beginning with games known for their dungeon generation, These include world-renowned titles such as 'Rogue'(1980) and Blizzard's 'Diablo' game series[Gamasutra 2012]. Over the past few years many new games have been released that are using one of these techniques, Such games that have taken great advantage of these are games in the looter shooter genre and also in the RPG genre , which are two genres where the player wishes to have as much content as possible to explore and partake in or having as much loot as possible, Be those weapons,armor,accessories,quests and also possible scenarios. There are many of these types of games available on the market at the moment some of these include 'Diablo' and 'Borderlands'. Other genres use it to generate their game world's by doing this they create games that have a high replay value and can keep people coming back for more and more new experiences. But some games use this idea and increase it exponentially. One such game is called 'No Man's Sky', this is one of the most ambitious games using such techniques, using them to procedurally generate planets, animals and plants, as well as the universe itself. It is said that there are over 18 quintillion planets in No Man's Sky, or 18,446,744,073,709,551,616 planets to be accurate[Murray 2014]. All of which have varying sizes, varying flora and fauna and also varying ecosystems alongside other structures for the player to explore.

Procedural generation has been used in other sources of media, such as film. The Lord of the Rings movies for example, Used a high end artificial intelligence software by the name of MASSIVE to create the thousands of soldiers that fought in the onscreen battle scenes[MASSIVE n.d]. Other movies this software has been used in include Avengers Endgame and 300.

Procedural Generation Techniques

Procedural generation can really be anywhere from trivial to extremely elaborate, it all comes down to the users needs. Procedural generation can be anything from choosing a start number and an end number and then generating the numbers needed between for example from 0 to 100 and then have the computer pick out each odd number, using this example we use the computer to generate data using an algorithm.

Procedural generation can be classified into six types. Some of these can be created during development or even while someone is playing the game. Some of these algorithms all they might need is a simple seed for the random number generation whereas other algorithms have more control over the limits of the algorithm.



[Bontchev 2016]

In procedural generation some of the most desirable qualities are both controllability and believability as when you are creating objects you don't want the players to feel like the environment is out of place or wrong, you need it to look real and fit the area. That is where controllability comes in because if you have a high degree of customisation, then you can more easily shape the content that the algorithm creates.

This technique is about feeding the algorithm some of the parameters you're looking for, and then the algorithm will produce the data that fits your requirements along with a certain amount of randomness.

An example of a procedural generation technique:

Lindenmayer Systems (L-Systems)

The Lindenmayer system was created by Aristid Lindenmayer in 1968.He was a Hungarian botanist. He created the system to show the behaviour of plants and the growth process of these plants.

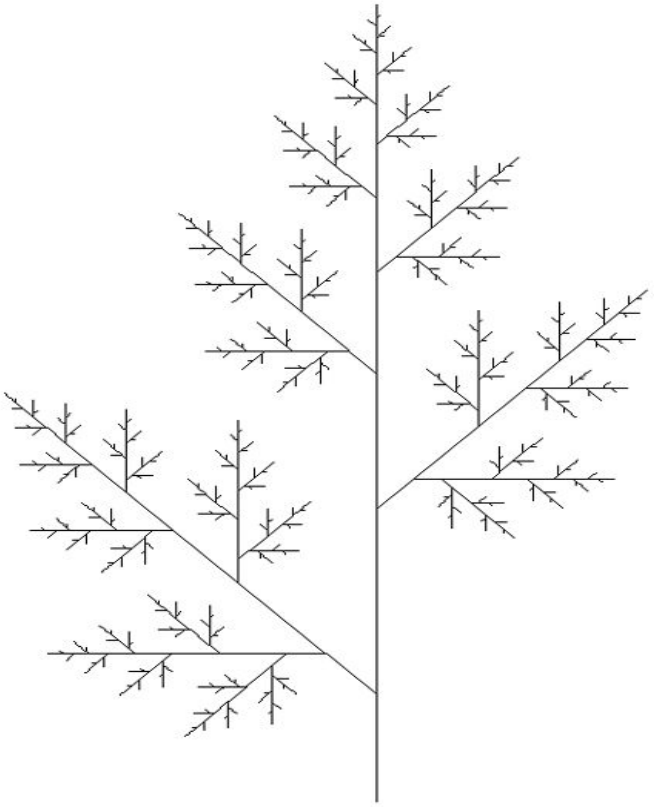
"Two principal areas include generation of fractals and realistic modelling of plants"[Ochoa 12 February 1998].

L-Systems work well for objects like these by recursively iterating over an axiom that the user specifies and based on the contents of the axiom, Rules are called and rewrite this axiom. The rules are a series of string characters that all have different actions they perform and then based on the rules and the number of iterations,angle and the length you are looking for different kinds of flora can be generated from this. The more characters the user has defined with different actions the more customisation that the user has over the creation.[MorphoCode n.d]

L-System Leaf

Written by Paul Bourke

axiom = a
F -> >F<
a -> F[+x]Fb
b -> F[-y]Fa
x -> a
y -> b
angle = 45
length factor = 1.36



[Bourke July 1991]

Study

To begin my research on the topic of *Procedural Generation of Game Assets* using Unity. I began first by looking into game engines.

From my own prior knowledge of different engines, My choice was between two different engines that would be ideal for what I am trying to accomplish. These engines are Unity and Unreal.The reason I decided on using Unity for my project was due to my lack of experience in the use of unity. I had used unity in a module from a previous year but I had not used it to an extent where I felt comfortable with it and so with this knowledge I decided on using unity and since unity has an extensive collection of documents that explain the ins and out of the engine so this was a key factor in helping me get up to speed with it[Unity Technologies 2020] and the opportunity to improve my knowledge on this engine was something that I was excited about.

Another reason I decided to use Unity over using other engines was due to the fact that Unity provides me with the opportunity to fulfill my aim to make my editor in a 3D world and also since over the past years in college I have not had much exposure to working in 3 dimensions.

This study gives me an understanding of the engine itself and there an understanding of how far I can push Unity's capabilities in this project.

After this I began into looking into specific objects that I could procedurally generate in my project. I found many examples across the games industry ranging from whole worlds to the smallest pieces of grass. I felt that given the time I have to complete this project and also the complexity of some of the assets currently being done across the industry that I would have to pick something that was smaller in scale rather than a complete world. So I decided to look into the creation of roads as it is not the most common asset that is created using procedural generation as only a handful of games use this method, In most games most of the locations or areas in such games have set paths to and from each other.

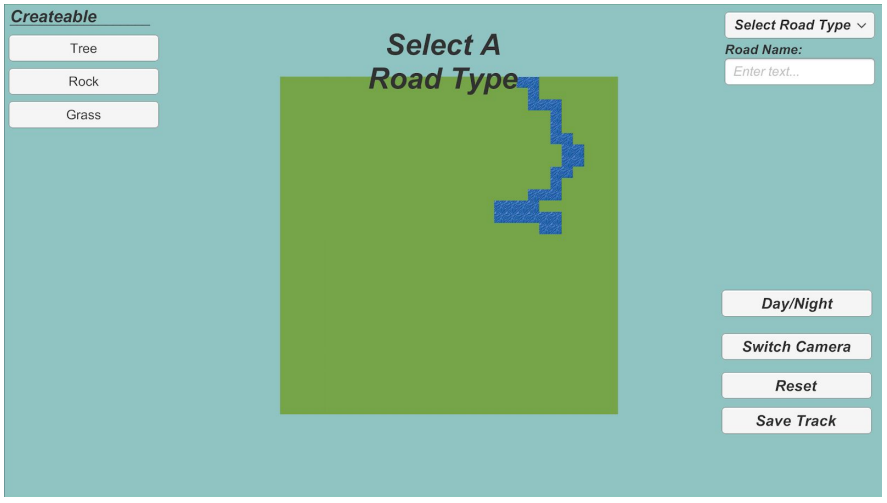
I then began looking into different editors that are used to create roads, games such as *SimCity* and *Cities: Skylines*. Both of these games can create roads by using a directional tool that you drag to both extend and change the roads directions. I saw how effective these methods are at creating roads and after I saw this method I decided that I would like to incorporate this directional tool into my game, I then began to look into creating bezier curves to achieve the same outcome as these types of road creators. However I later decided that this would not be the best method for what I wanted to do as I was looking more into creating a tile based system instead. However some of the ideas used in these editors I decided would work well in my editor such as the directional tool to allow me to pick the direction of the road as my aim for the road creation is to make it as simple and straightforward to use as possible.

After I felt confident with my study on roads I began looking into creating other roadside objects and other objects you might see from the road and one type of asset that came to mind

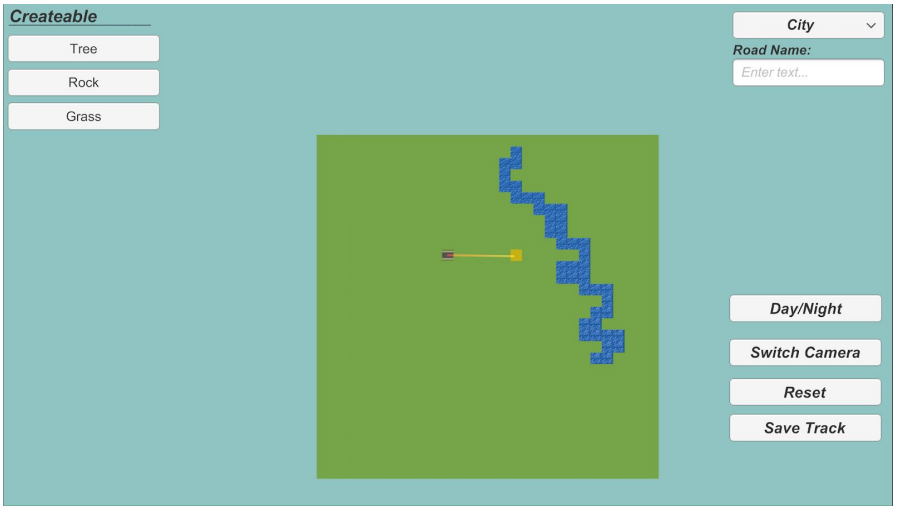
was vegetation, this would include grass and trees. I began researching into different methods to create believable vegetation, I found a few methods but one of those methods stood out more than the rest of them. That method is known as L-Systems, it is a well known technique that is used widely in the study of plant development and is one of the major methods for modelling developing plants. After seeing how effectively you could model vegetation using this method I decided that this method would be perfect for what I was trying to achieve. So I began looking into it in more depth and began finding some research that helped me a great deal in understanding the ins and outs of this algorithm so that I would be able to create my own implementation to create the vegetation in my editor.

Project Description

When the player starts up the editor, They are greeted by the editor UI and also a blank area along with an area of water running through it. If the player does not like the placement of this water you are able to reset this area to get another more agreeable one.

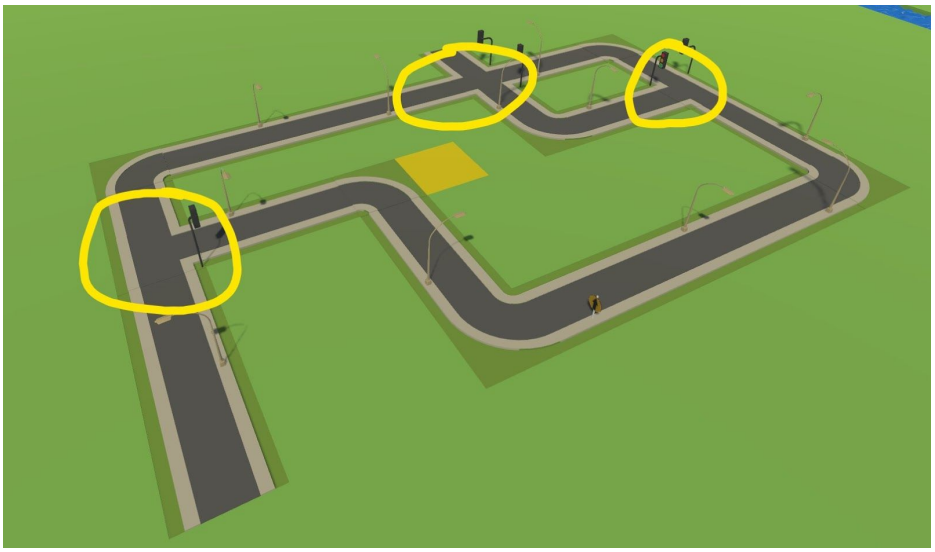


Once the area is suitable then if you look at the upper middle of the screen the player will receive a message telling them to choose a road before proceeding as they will not be able to do anything until the road type has been chosen.



Once the initial point and road type has been chosen, This is when the creation really begins, from the initial point of the road the player drags the cursor and when they move the cursor a line is created between this the initial point and the cursor. This line acts as a directional tool for the player to know in which direction they will move in. Once the player has chosen a direction and releases the mouse button then the angle is sent to the algorithm and from here it decides on the correct road to create for that given situation. As the player creates the road you may begin to overlap on some of the tiles and if that happens under certain circumstances then the editor will decide that a tile should be replaced with another tile that suits the current situation. so for example if two roads are perpendicular to each other and they intersect then

this point of intersection is changed to an intersection tile or if the player decides they want a turn on a previous created road then just move back to that road and then create the new corner road and if this happens this point is change to a T-Junction tile to create a seamless looking road. As you can see in the circled points below.



Along the roads that are made I am spawning roadside objects which range from road signs to lamp posts as well as traffic lights at certain points. These are there for both visual and in some cases functional uses. For example the lamp posts actually light up the game if the player decides to switch the scene to night mode.

As the roads are made also along the side buildings are created using an algorithm to create the floors of the houses. The walls for the houses are selected using another function which decides which type of wall to create at that point be it a normal complete wall or one of the window variants and then once this is done it selects the door. The door is created on the

front of the house and deletes one of the walls and replaces it with the door.

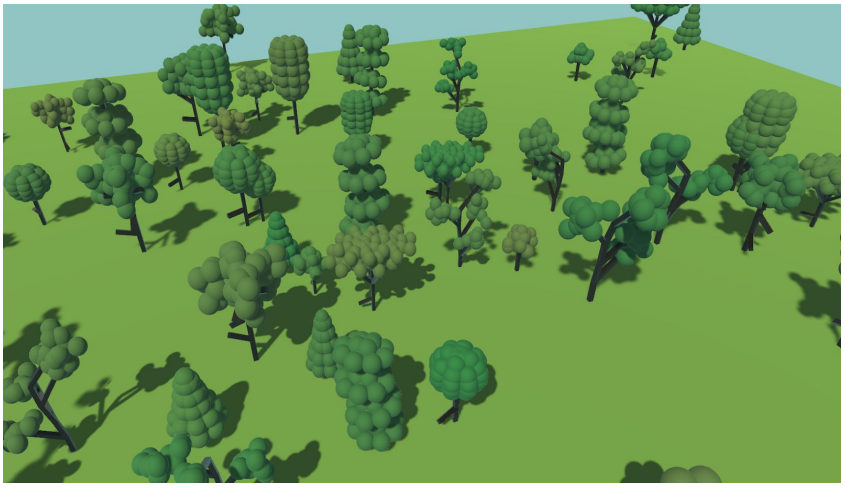
The roofs of these houses are created using procedural runtime mesh generation. Which is creating a custom mesh at runtime based on the parameters of the house and then placing these custom meshes at the correct points on the tops of the houses. I have several types of roofs and when the house spawns the algorithm decides which roof to use in that situation.

On these roofs depending on the roof type other roof add-ons can be added to include as seen in the image above dormer windows.

Below is an example of a completed road.

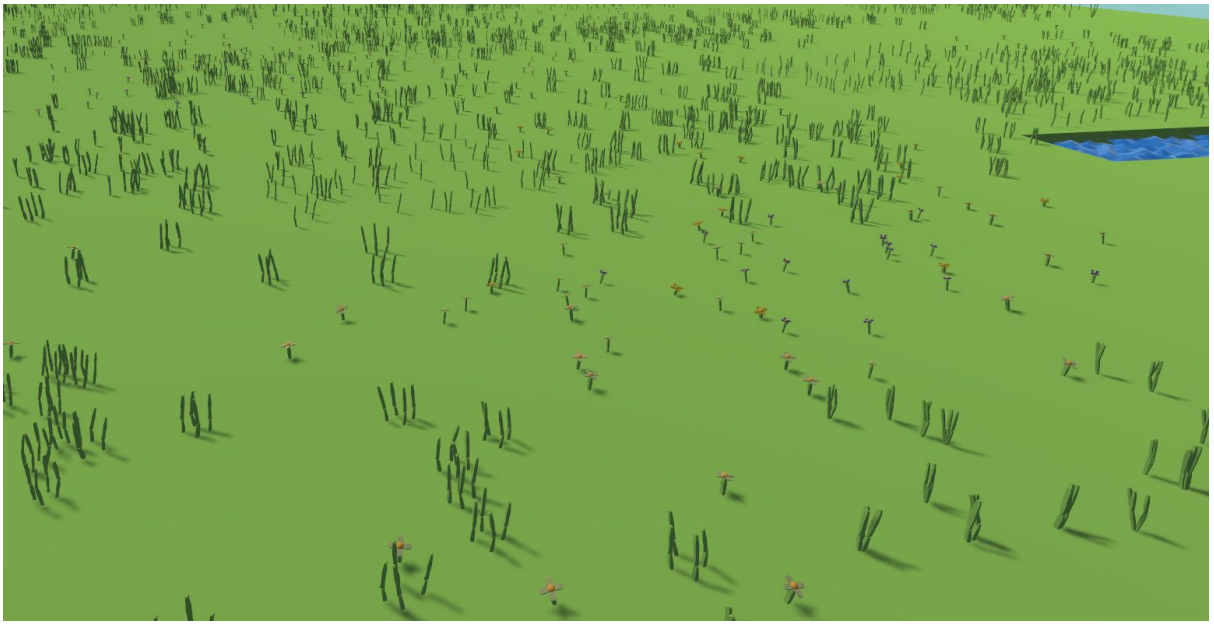
Once the player has completed this. They are then able to create other objects throughout the space. Using the menu on the left hand side of the UI the player is given the choice between Trees,Rocks and grass objects. The trees are place around the scene using an algorithm known as Poisson Disk Sampling which takes a point and the picks a random direction and then moves in that direction and checks if its outside the radius and if it's not then the point is not added to the list otherwise its added to a list and returned. Once the position has been

decided then the L-System creates the tree from a choice of a few different rulesets that I created.

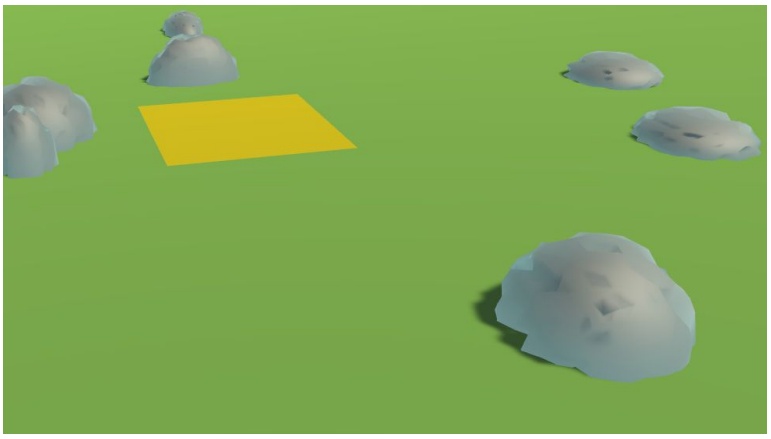


Using these two methods I am able to create completely different areas each time as the points are almost never the same. The trees also have an age statistic that changes both the size and color of the tree depending on the age.

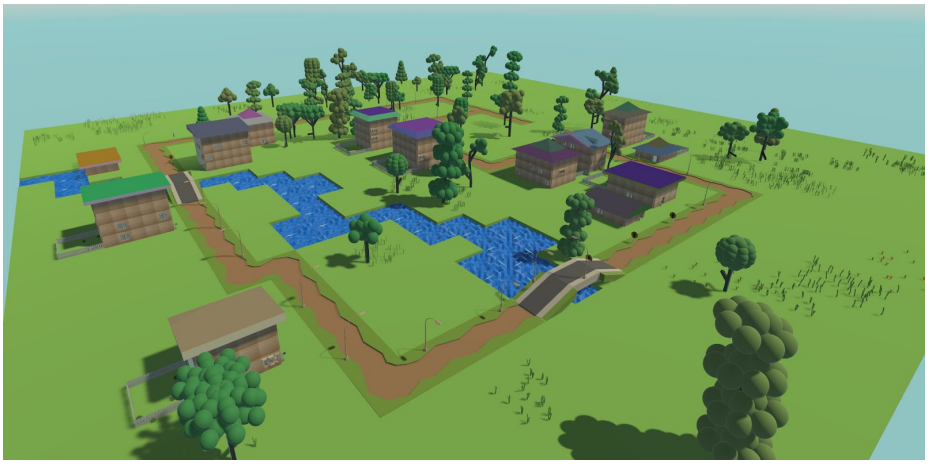
Grass is another object I have created using L-Systems, the grass has a few different variants and along with that the grass also has some flowers mixed in with them to add more variation. The edible parts of the grass in the UI are the density of the grass objects which dictates the number of grass blades there are in that area and also another that spaces out the blades of grass to have them cover a larger surface area.



The rocks do not have any editable variables, but they are created using a simple perlin noise alterion to their meshes. I take their meshes and apply perlin noise to the rocks vertices this creates an irregular shaped mesh which creates the rocks.



Once the player has completed using all these features then the once blank area has now become an area filled with procedurally generated objects. In this way no two generations of the area will ever be the same.



Another feature is the camera switch option that allows the player to take control of the camera and move down to inspect the area from much closer and by clicking the player may delete any object that they do not wish to be placed.

All these mechanics are crucial for the effectiveness of the editor.

Bringing all these features and mechanics together and implementing them into an editor I feel answers the question I put forward at the start of my project and that is procedurally generating game assets in unity.

Project Milestones

October 2019:

I began coding an early idea for how the roads would be made. My initial idea was an idea using splines which is basically a curve that connects two or more points. I believed that this would allow me to create believable and smooth curved roads. However as I had a working piece of spline code that would allow me to show off the curve itself I believed that this was not the style I was looking to use. So I opted to try another method one that was more in line of the type of road I wanted to create. The method I was looking to try was to create a more tile based road creator that would use tiles instead of adding points to a pre existing line to add more road surface. So I began creating this and started with creating a square of connecting roads that knew its size by the sizes the player inputted but as I was testing this method and as I got other people to test it. the same conclusion was usually reached. That it was a very rigid system. So I went back to the drawing board with the actual creation of my tile based road.

November 2019:

At this point I had successfully created a road builder that I was happy with the player would choose the direction that they wanted to move and then using the angle of the player's mouse from the previous point a new road would be created and would slot into place alongside the other roads. The road was able to create corners,T-Junctions and intersections if any roads overlapped.I held another test and the results of this test where the total opposite of the earlier method, This one was much more well received and that this method was much more flexible in creation and allowed the user more freedom.

It was at this point that I decided to look into a way to create my trees and thanks to my research I had a great start with the knowledge I had gained about L-Systems I decided that this would be a good place to start and began working on creating a string based rule system to help me create my trees.

December 2019:

By this time I had a working L-System that was able to generate based on the rules input by me however I did not have a working rule that would create a tree with believable branches. It was just a basic rule that created blocks along the lines to simulate the trunk and branches of a tree being created. On the topic of the roads this month I decided to add a new feature to the roads that allowed you to move the roads up off the ground like a bridge which would come into play next month when I would be able to add in the asset to make the bridges make sense. By getting the L-System operational was a great milestone for my project as it meant that tree generation could be done.

January 2020:

Over the christmas break I implemented a water tile that is just a plane and im manipulating it vertices using a wave like motion to simulate water moving and as I implemented last month

this is where the bridges come into play they will allow you to make roads over these rivers. At this point this process is still manual as I need to be able to test this functionality wherever I am. I started working on UI and began with fields that allow you to choose the style of roads be they Dirt roads or city roads and also a way to name the track you've made so that then using the new save button which calls a new script that saves the newly created environment and places it into the a folder so it will be available for reuse.

February 2020:

At this stage I was happy enough with progress on the roads, So I decided to start work in the houses and I began working on the basic house where you could choose the size of the house,number of floors and the windows in the house. Then I began placing them along the sides of the road. I made a change that allows you to swap cameras and move down into the environment to let the user get a good look at what has happened and what's been created.

I also made minor changes to the UI that now allow you to completely reset the starting area.

March 2020:

At this point in the project I began creating the roof for the houses, Which was the hardest part of it for me. I Started with just using objects I had created in blender but this was not working out quite how I would like so I began redoing it again but this time I decided to create it using procedural mesh generation instead so I created the roof from vertices and triangles I created at runtime. This was still early days in this as I had to make more types of roofs.

April 2020:

I had talked about creating a roof using procedural mesh generation and this had developed a lot over the early part of april. This was a great milestone for the building generation for me and from there I continued to make more roofs.After I had made enough variations of roofs I then moved onto changing the L-System to allow me to create my own grass using the L-System and also while doing this I began creating new rules for the tree generation so that there would be more variations in the trees. I began updating the way that the trees would be positioned on the map, I began using poisson disk sampling to place the trees around in a more natural way. At this stage I began encountering some performance issues since I had hundreds of objects in the scene, So to counter this I implemented a way to combine all the meshes of an object to a single mesh to help with the performance and then I deleted the other objects after combining them all. I began fixing the manual way I have the roads moving over the water to an automatic approach.

Earlier in my development I had found a way to save the created area into a folder, However this feature began to show some bugs after changing to the the roof types and also since I was now combining all the objects meshes at runtime this meant that the meshes were being lost once the game was not running anymore this lead to errors with the saving, So I decided to leave it in the project for now to show that it was possible to save the objects, However it is currently not functioning as expect so I removed the method for saving.

Results and Discussion

The final product is as an Editor that could be used using procedural generation techniques to create a specific area of a game world. The player starts with a blank plain and a message will appear on the screen asking the player to choose a road. when the player uses the on-screen interface to select a path, then the player can select their starting location. First the editor will create a standard road and then from here they will select the path to create based on the line angle generated by the user from the point of origin.

When the construction of a road is underway. The editor will build lamp posts, signs and houses along the roadside. The houses are built in a procedural way, so that no two houses are the same. As the road gets bigger the likelihood of overlapping roads gets much higher, so as a solution to this I made it so that when the roads overlap under some conditions road tiles are replaced with intersection tiles and T-Junction tiles.

Once the player has finished creating the roads they move onto creating the grass,trees and rocks. They again are all procedurally generated. These objects can be placed using a rectangle select tool that would show you the area in which these objects would be created inside.

However over the course of working on this project I have noticed two potential problems with procedural generation.

The first is the impact on performance that I had experienced before I had implemented any performance increases. Using my L-System I am creating objects for the trees so each part of the tree is a seperate object so some trees depending on the size can be a combination of dozens of objects and this causes a performance hit. From this I can tell that if you do not take care with some of the techniques some of them can cause performance issues and in certain games performance is the most important thing.

The second problem comes down to empty space, every so often you can notice parts of the world that might be empty or lacking the same amount of detail in them as the rest of the world. I have noticed over the course of development you are sometimes left with an area of the map that can be left lacking detail. This in a small space is not as noticeable however in a game where the terrain or planet let's say is generated you don't want large spaces of emptiness. In a game this can cause a break in immersion for the player and lower their overall experience in the game.

As a result of all this, It shows that no two pieces of the world that I create using my editor are exactly the same and from this I can conclude that procedural generation is an excellent technique to use if you are looking to create game assets that are varied and this can improve replayability overall in your games by creating new scenarios each time as long as you are careful and manage the potential problems I mentioned earlier as failing to address these issues can have a negative effect on what you were initially trying to achieve.

Technical Achievements

Throughout the entirety of the project my aim was to create an editor that incorporates procedural generation techniques. I believe I have achieved this goal as, Firstly I have managed to create a fully functioning editor that has a working interface and incorporating the procedural techniques also into the editor was itself a technical achievement for myself.

A few other examples of some of my other technical achievements over the course of the project:

- Since I was creating this project in Unity I became quite familiar with the ins and outs of the unity engine and its capabilities and now have become more comfortable with using game engines.
- I have also been exposed to many new algorithms that I had never known about before this project, such as L-systems,fractals and poisson disk sampling along with many others. which was a fantastic experience even If i did not implement them learning about them was a fantastic and exciting experience.
- Over the course of the project I learned a great deal about working in a 3D space as I have been working mostly in 2D over the past few years. The experience of working in 3D has been a great benefit and I now feel more comfortable working in 3D.
- Something I found very interesting over the course of the project was the impact that the generation of assets had on performance. At one point in the project I had hundreds of objects making up a tree and dozens making up the houses. FIxing this required me to think of a way to either combine all those objects into a single object or to increase the size of the objects and lessen the number. I came to the conclusion that combining would be the best option as I would not lose the more fine details of the objects by covering them with larger objects. So I merged all the meshes of the object together to create a single mesh object.

Project Review and Conclusions

In this section I plan to review my project and discuss the development of my project process as well as what went right and also what went wrong. Along with that some things I would do differently now looking back on the project.

A few things that went well I thought in the project were the creation of the roads and the way that I have the roads moving automatically over the water. Since when I first tried to do this I had some trouble trying to determine when I had hit a ground tile after being over the water. Another feature I am very happy with was my L-System which I implemented to a degree of functionality that I was incredibly happy with.

A few things I was less than happy with in my project include my implementation of the rock generator as I feel like given more time I would have been able to flesh out that feature much more but due to time issues I had to prioritize the other features over that feature.

Another problem that occurred early on in development, Was that I had begun looking into using bezier curves to begin my development on the roads. My plan was originally to create the road mesh alongside the bezier curve and then just add points to the curve which would in turn continually add the mesh. While I had a working prototype of the curve itself with no mesh around it, I had then realized that this was not the way I wanted to approach this after this investigation.This looking back now was something that I had approached wrongly at the start and I should have given more time think about what I wanted the road editor to look like.

Early on in the project I had found a method through my research into unitys capabilities to save an object created at runtime. At that point in the project when I had not incorporated all the systems into a single scene I only had placeholder objects for the trees and the buildings it worked like a charm and would save the object out as a prefab, however once I combined all the systems into the one scene and then implemented my new combine mesh function it cause that to cease working as since I was now created meshes at runtime and deleting the previous ones to save performance this lead to the mesh not saving with the prefab and thus losing them after saving. I began looking into ways to tackle this problem and one way I had come up with was to save the whole area and then save the positions of everything on the area into a script or file and then once the player drags the saved object into a script just use those positions to recreate the area. But due to the time frame left in the year, I decided that I would prioritise other tasks in the project before this one and if I had time at the end I would revisit it.

If I had more time to develop the project the saving feature is definitely one I would revisit and try to complete as I think it would be a very good feature to have in the editor as saving your creations and using them inside another project would be a great feature to have. Another thing I would like to revisit would be the performance of the editor as at the moment it is still not as smooth as I would like it to be. Although I had implemented my mesh combining into the project I still feel like there are many other ways that I could optimise the project to run even smoother. Another feature I would implement would be a slow build

mode where the objects that are built are created slowly so you can watch them build from the ground up to show the player them being built at run time.

If I had to start this project I would definitely have spent more time researching into the different methods I could use into building the roads, By the end I was happy with the method I had chosen but I believe that had I taken more time to think about what the completed version of my project would look like I would have reached my current method much faster. I would also have done weekly refactoring as by the end some of the code had become rigid and could most definitely have been more flexible.

To others attempting a similar project in the future I would say to get a good idea of the main assets you wish to generate first and see if there are any other objects that you could incorporate into the project that would elevate that main one up even higher.

I believe that my choice of technology was actually a great choice as the amount of literature on how to use unity helped me tremendously through my development and also with its visuals I was able to produce a much better looking game than doing my project in 2D.

Even so, I believe that my decisions were for the good of the project as a whole throughout and I had to make some difficult decisions and some sacrifices had to be made to deliver this project. But I can honestly claim that this has made my project a success, and I'm very pleased with that.

References

Referenced Publication	Citation	Reference
Website	(MASSIVE n.d)	MASSIVE. (n.d). *About Massive*[online]. (http://www.massivesoftware.com/about.html).(Accessed 27 April 2020).
Website	(Aversa 2015)	Aversa,A. (2015). *Procedural Contents Generation* [Online]. (https://www.davideaversa.it/wp-content/uploads/2015/06/Procedural-Contents-Generation.pdf). (Accessed 14 December 2019)
Website	(Blatz and Korn 2017)	Blatz,M. and Korn,O. (2017). *A Very Short History of Dynamic and Procedural Content Generation* [Online]. (https://www.researchgate.net/publication/315863952_A_Very_Short_History_of_Dynamic_and_Procedural_Content_Generation). (Accessed 13 March 2020)
Website	(Bontchev 2016)	Bontchev,B.(2016). *Modern Trends in Automatic Generation of Content for Video Games*[Online]. (https://www.researchgate.net/figure/Taxonomy-of-procedurally-generated-game-content-after-11_fig7_320615781).Accessed(3 May 2020)
Website	(Burke 1991)	Burke,P. (1991). *L-System User Notes*[Online]. (http://paulbourke.net/fractals/lsys/).(Accessed 15 November 2019)
Website	(Gamasutra 2012)	Gamasutra.(2012). *Procedural Content Generation: Thinking With Modules*[Online]. (https://www.gamasutra.com/view/feature/174311/procedural_content_generation_.php).Accessed(16 January 2020)

Website	(Halliwell 2008)	Halliwell,L.(2008). *Procedural content generation* [Online]. (https://lukehalliwell.wordpress.com/2008/08/05/procedural-content-generation/).(Accessed 17 February 2020)
Website	(Microsoft n.d)	Microsoft. (n.d). *Microsoft purchases 'Minecraft'*[online]. (https://news.microsoft.com/announcement/microsoft-purchases-minecraft/).(Accessed 1 December 2020).
Website	(Morphocode n.d)	Morphocode. (n.d). *Intro to L-systems* [Online]. (https://morphocode.com/intro-to-l-systems/).Accessed(15 november 2019).
Website	(Murray 2014)	Murray,S.(2014). *Exploring the 18,446,744,073,709,551,616 planets of No Man's Sky* [online]. (https://blog.eu.playstation.com/2014/08/26/exploring-18446744073709551616-planets-mans-sky/).(Accessed 2 April 2020)
Website	(Ochoa 1998)	Ochoa,G.(1998). *An Introduction to Lindenmayer Systems* [Online]. (http://www1.biologie.uni-hamburg.de/b-online/e28_3/lsys.html#OVR).(Accessed 14 November 2019).
Website	(Unity Technologies 2020)	Unity Technologies.(2020). *Unity User Manual*[Online]. (https://docs.unity3d.com/Manual/index.html). (Accessed 29 April 2020).
Website	(Van Brummelen and Chen n.d)	Van Brummelen,J and Chen,B.(n.d). *Procedural Generation* [Online].

		(http://www.mit.edu/~jessicav/6.S198/Blog_Post/ProceduralGeneration.html).(Accessed 18 March 2020)

Appendices

Faculty of Science

Open-Book and Remote Assessment Cover Page

Student Name: Brian O'Neill

Student Number: C00213756

Lecturer Name: Philip Bourke

Module: Project II

Stage/Year: 4th Year

Date: 03/05/2020

Declaration

This examination/assessment will be submitted using Turnitin as the online submission tool. By submitting my examination/assessment to Turnitin, I am declaring that this examination/assessment is my own work. I understand that I may be required to orally defend any of my answers, to the lecturer, at a given time after the examination/assessment has been completed, as outlined in the student regulations.