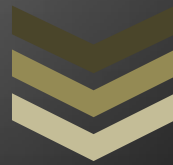




PRC0303 – Final Year Project Report – Houndtor Histories & Legends



Gareth Wright 10239101

This project aims to deliver the reader a report on the complete development of a visualisation driven, hybrid game development that leverages the local moorland area of Hound Tor, Dartmoor National Park. Devon in the UK through transposition into a 3 Dimensional interactive replica environment, alongside local historic and factual game play elements used to gently drive the overall experience.



Page left intentionally blank

I Abstract

This document delivery is the final submission works for the Computer and Games Development BSc Hons Final year Project Module.

It contains a finalised and collated report that culminates with a deliverable product that is available for download on through the accompanying blog.

The document follows a report style and contents are provided.

At the back of the document are all design, code and build elements, where it was not reported on as part of the main document.

II Acknowledgments

Thanks and acknowledgments to:

University of Plymouth: Science and Technology Department

CGD (Computer & Games Development)

Dr Nigel Barlow, Dr Torbjorn Dahl, Professor Shirley Atkinson, Martin Beck.

Gareth Williams:

Marius Varga:

Luke Angell:

Ewan Armstrong:

James Hayter:

Eric5h5, _zombience_, Unity Answers community:

Andrew Cuffe:

Dr Dan Livingstone:

Dan thanks. You have basically helped me at every stage this year, you have helped me to challenge my own ideas, create new ideas where there were none, offered the kindest ear to the years misfortunate events and basically fought at my side to deliver myself from this final hurdle. Your whole-hearted kindness and compassion has without doubt been the most notable memory to take away from the herds of interesting and lovely people I met during my year at Plymouth.

I hope one day your labour of love, loves you back.

I can't think of more words that describe how grateful I am to have met you and the CGD crowd at Plymouth, so thanks for the one of my best years ever!

Mum & Dad:

You've underpinned every choice I make with love and support throughout my life, regardless to say I couldn't of done this without you two.

Mike & Becky: Hey sis, what a year eh. The things you have had to cope with so far this year do not bear thinking about but they have been very real evidencing the one thing it cannot take away and that is your beautiful nature and lovely mind. I'm back now so thanks for waiting patiently while this still beckoned to be finished. I got some ace movies overdue to be watched and we might still catch the BBQ weather☺. My thoughts are always with you sis, thanks for support you have given, when it should of been me supporting you.

Sophie, Oliver, Lily: Thanks for putting up with me; I have been a selfish entity of which you have all worked around for four whole years. It is over. I love you all. Party time!

Table of Contents

I Abstract.....	2
II Acknowledgments	3
1 Introduction	6
2 Project Objectives	7
3 Field and Key Players	8
4 Technologies	9
5 Methods of Approach.....	11
5.1 Code Design Principles carried through to project.....	11
5.1.1 Factory Design Pattern with prefabbed components	11
5.1.2 Singleton Design Pattern in the code	14
5.1.3 Conformance to XP	14
5.1.4 Brief Project Scope summarisation.....	16
5.2 Project Documentation Routes	16
5.2.1 Package Diagram.....	16
5.2.2 User Stories.....	17
5.2.3 Use cases.....	18
5.2.1 Class Definitions and Diagrams.....	19
5.2.2 Sequence Diagrams	20
5.2.3 Class Dependencies – An Overview	21
6 Game Design	22
6.1 Level design	22
5.2 UI design	25
6.2.1 Start Screen UI	25
6.2.2 In-Game UI	25
6.2.3 Game over UI	25
6.2.4 In Game options.....	26
6.3 Elements of Design used to create Game Assets	27
6.3.1 Autodesk and Blender	27
6.3.2 Adobe Photoshop CS5	27
5.3.3 FL Studio, Ableton Live 8, Line 6 –GearBox Software	27
6.3.4 Breakdown of Scene Artwork and Models Created	29
7 Implementations.....	32
7.1 Development Phase 1	32
7.2 Development Phase 2	37
7.3 Development Phase 3	42
8 Project Management	44

9 Product Test.....	46
10 Final Critical Evaluation.....	49
Delivery of Working Game Mechanics	49
Delivery of Project Final Hand Prototype	50
Delivery of Gameplay & Design Documentation	50
Delivery of Demonstrable product available for download over two tested platforms	51
Delivery of own code, where feasible, and stick to a C# only code rule	51
11 Conclusion.....	52
11.1 Conclusion.....	52
Skills learnt or enhanced:	53
11.2 Recommendations for the future development of the project.....	53
13 Bibliography	55

1 Introduction

This project covers the full and final delivery of the final year project report.

In chapter 2, the project objectives are recovered by short description to familiarise the reader to the goals reported toward in the document ahead.

Chapter 3 returns to key players and field identification.

Chapter 4 recaps on Technologies like Unity 3D, used throughout the project's life, and other elements used in each part of the build.

Chapter 5 runs through methods of approach to designing the system with additional material located in the appendices at the end.

In chapter 6, we see the game design laid out and reported upon, whereby breakdowns are given on level design and UI design identifying both conceptual and resultant stages. Moving later into further design elements outside of the Unity environment like OST audio production and 3d modelling software.

Chapter 7 covers the implementation phase over the iterations testing phases, where reports on any changes have been documented.

Chapter 8 returns to project management to divulge the outcome of following the processes undertaken to run the projects timeline.

In chapter, 9 Product test tables identify problem areas and provide feedback given for final release before public test phase.

In chapter 10 we round up the report with conclusions and recommendations for the future of the prototype.

After this the appendices are there to support the document and expose the incremental deliverables the project required by providing both previous documents before this was delivered.

2 Project Objectives

2.1 Deliver Working Game Mechanics

Delivering good game mechanics is obviously essential to a development like this, and denotes optimized game functionality and easily readable code content

2.2 Deliverable Project Final Hand Prototype

Iterations of the developments have been released alongside development, though the last version has been close to the bone with a final better-polished release just days prior to the project hand in due to unforeseen circumstances.

2.3 Gameplay & design documentation

The systems documentation provides all the detailed overviews of classes and falls through to code where documentation was covered under XP principles of code commenting.

2.4 Demonstrable product

During the final stages, the testing of the product is even more critical and the demonstrable product is available online for public testing at "<http://hountorgame.blogspot.co.uk/2013/08/hound-tor-history-legends-virtual.html>". At all stages, the release testing deadlines were met. However, there were phases of the release that occurred later than expected but it transpired that the actual overall delivery deadline was achieved, as can be seen on the blog where one can download the final prototype.

2.5 Create own code where feasible and stick to a C# only code rule

Through the conversion of any JavaScript's into C# because of Scripting advantages and clearer code instructions for others viewer of it. JavaScript, being interpreted and loosely typed makes for heavy intermediary conversion and calculation at runtime for Unity 3D, making C# the better scripting tool that also makes more succinct use of its ported original .NET framework accessible from Mono. It is not the first recommended language by Unity, but is fast becoming the de facto standard by many programmers. Unfortunately, many previous versions of Unity have utilised Unity-Script (their version of JavaScript) so finding succinct tutorials that cover c# are difficult to find. To complete the task nonetheless, the Unity Answers and Forums were joined in anticipation of help if any sticky patches occurred. There did, mainly due to Unity quirks and the like but this site proved invaluable and encouraged me to release many questions that were either up voted to an important for all community question, as some issues were common but as yet unresolved due to poor communication along the lines.

In the Appendices, section at the back under Appendix D is the complete code set created for this project by this author. There are some codes that deal with image effects that are JavaScript based but necessary that are not in the code section as this area has been kept purely to my code and the leveraged code being the image effects and dynamic particle will be converted to C# at the next dev phase, post Project hand in.

These are included in the prototype but are expressly not a part of the final prototype and are there to make up for areas that could not be reached during the development.

3 Field and Key Players

There is a potential (via port to tablet (Android or IPAD)) to deliver the game as a more casual “point and touch” (to get taken to a point in the game world) type of virtual experience, making the controls less and less complex has always been a goal of the build which in turn opens the scope up to further key players, and though released on both web player and standalone, where windows users can leverage an XBOX controller or keyboard and mouse controls.

However, and with the above in mind current Key Players in the field of Visualisation technology are companies such as

- Interactive Systems Studios, Plymouth.
- This company specialises in visualisations and are currently in phase testing for a localised project dealing in environments suiting different educational aspects available at the University of Plymouth.



(Interactive Systems Studio, 2013) The image above shows Part of the visualisations currently under development by the ISS team regarding a University project

Further key players are the Military where application of visualisations for field training is seen as essential to safer operations. This stretch from armed manoeuvres over all factions of the military and breach bang and clear virtual conflict training for Special Forces.

Further player may include national bodies such the Tourist board and the government run National Parks. These are potential interests for the evolution of the prototype to becoming something more scientifically based than experiential with potential to visualise water table data over real landscapes to analysis the data visually.

The Tourist board may have an interest in the product as the experience shamelessly promotes the local Dartmoor area, one of the most prolific attractions in Devon.

4 Technologies

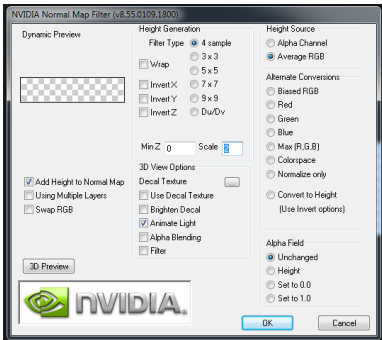
As this was a student project and with an unforeseen later purchase of Unity3D Pro for education license, cost had already breached any limitations as they were set to zero, aside from fuel allowance to hound Tor for reference shots and a final thought purchase of a dynamic particle effect for a dollar.

In order to keep cost to a minimum from hereon in the process of learning modelling software was undergone, this

As can be seen in **section 5: 5.3.4** of this document, the process of getting used to the modelling software was a challenge but it did result an outcome of models to use that were not cost based, though by no means perfect at this stage.

As such, the technologies undertaken to get this project finished, is composited in the table below:

Technology	Usage in project
Unity 3D Pro edition (education license)	Game development environment where the game will be assembled both visually and through code application. Plus any addition of third party assets are pipelined through Unity either via the standard assets packages or Asset Store downloads
Mono Develop / Visual Studio 2010	Code and document the code. The Mono Develop IDE was a satisfactory development IDE to use with Unity and it integrates well with the environment
Development Language	C# strongly typed OO language
Modelling Software products: <ul style="list-style-type: none"> Autodesk™ - 3DS Max 2013 (student ed) Autodesk™ - Maya 2013 (student ed) Autodesk™ - MudBox 2013 (student ed) Blender™ 2.6.4 (open source) Luxology™ Modo (student ed) Pixlogic™ Sculpttris Alpha (free software by maker of ZBrush) 	All of the programs mentioned on the left were undertaken to establish the simplest route for beginners into modelling software programs. It was found that any one should be stuck with singularly and time spent will make good. The reason being the programs are involved. They require specific know how or, in similarity to programming, require knowledge enough to ask the right questions. Addendum to this, it was found that the wealth of "Youtube" tutorials available were often poorly relayed or spoken of without clear guidance to specific actions, making them difficult to follow at times.
Map Data procurement: <ul style="list-style-type: none"> Google™ Earth for tests Global Mapper™ for actual aerial shot that went forward to create the height map in modelling software MicroDem™ 	<ul style="list-style-type: none"> To gain the height map data via USGS Topological Plugin To get an only slightly better resolution top down view of Hound Tor than could be obtained through either Google Earth or Maps. for testing a method to extract the initial height map data from a Google Earth plugin that overlays topological data – this resulted in a dead end due to the resolution of the height map being too low with little or no differentiation in colour at the zoom level that was needed, basically the hound Tor area was too small for the USGS Topological Society plugin to work effectively. This was the caveat found with most packages offering some way of obtaining real map height data, whereby an area like Hound Tor figured out to be excessively small, most specifying a minimum 10, 000 km square

<ul style="list-style-type: none"> World Machine 	<p>selection get the best out of software like Global Mapper.</p> <ul style="list-style-type: none"> One technology that was not used but worth a mention was World machine TM. This software is an excellent way of creating bespoke height map driven environments, but as at the time of reading and throughout the project's past life span, World Machine cannot take in real DEM data that height map data is formatted to. It is a common conception that this will be addressed one day, though when is not made clear by world Machine on their website and related forums
<p>Bitmap Editors:</p> <ul style="list-style-type: none"> Adobe Photoshop CS5 TM NVidia Normal map filter (Photoshop plugin) to convert texture images to normal representations and including alpha version of the image. 	<ul style="list-style-type: none"> All texture editing 
<p>Audio Tools (DAW):</p> <ul style="list-style-type: none"> FL Studio 10 Abelton Live 8 GearBox / Line 6 	<ul style="list-style-type: none"> All audio digital creation and post processing of all in game audio (ambient, key signifier sounds, nature, narratives) Live recording of narratives at the desktop Used to input the recording microphone
<p>Many of the above technologies are further covered in section 5: 5.3 on the subject of extra elements used in the design and build, with screen shots available for some of the already mentioned technologies found in the Appendices under Appendix E</p>	

Several of the above technologies, Unity, C# and other tools are discussed in a more in depth research section found in Part 2 of the project deliverables. This can be found in the Appendices under Appendix

5 Methods of Approach

5.1 Code Design Principles carried through to project

5.1.1 Factory Design Pattern with prefabbed components

The projects classes created were relatively independent of each other denoting a much stronger indication of component oriented programming.

As a comparator, this type of scripting moved itself away from the confines and structural integrity of strict OO* found in a game library such as XNA or many others for that matter. This is because Unity's system namespace covers the base definitions and we as developers override the Methods, Interfaces, Enumerators, and Operations.

This sees the developer guided by a set of design patterns that should be of note, in order to comprehend the systems infrastructure better.

Below are the main Design Patterns that are notable in Unity 3D:

- **Abstract Factory -> Factory** – The king of all classes where interfaces are defined to create both related object and object that have no concrete class definitions and that will be further defined later on down the pipeline.
- **Singleton** – This pattern is available to the developer from the off and it would appear that one is actively encouraged to undertake singletons to embody a lot of a game object's component functionality. This is encouraged so that the developer may make tight self-operating components that can be modular and reusable ("pluggable"), either throughout the current game or as an easily port into some other development in the Unity 3D environment.
- **Decorator** – These are the wrappers designed to attach additional responsibility to any object dynamically.
- **Composite** – These patterns are common in the lower classes dealing with access to the graphics side of the application mostly.

Ultimately, The patterns found in the Engine would appear to be designed to make life easier for the developer at the top end, where she may only have to deal with a singular overriding draw call or even (as is found in Unity 3D) removal of any need to even address a draw call. In Unity 3d it is a given that you will want to do such a call, so is invoked at the end of the Engines override Update method.

As we have drilled down into the patterns, similarities can be noted the between XNA and Unity 3D at the lower levels (base components). The structure is actually very similar. (Not in any small, part due to the Mono implementation of the .net framework of course

(*though this can be applied, but could cause bloat of unnecessary custom base classes that were made due to this type of conformance)

As the design patterns indicate, By following the above common ideals when developing a Unity game we can ensure consistency with the systems overall design, which denotes:

- Create Reusable Objects where possible. Singletons that do a specific operation that can be attached to any arbitrary game object
- Script reusable elements to take in externally specified arguments in the editor inspector

- This is achieved via publically specified types/delegates/methods/Events or the c# “[SerializeField]” in which a private type can be accessed and changed in the inspector but remains private to its class at runtime. However, research and project build experience denoted “[SerializeField]” actually only adds serialization to a type’s field and does not exclude it from being changed outside the class. In fact, as stated on the Unity Script Reference website, with respect to field serialization...

‘You will almost never need this. When Unity serializes your scripts, it will only serialize public fields. If in addition to that you also want Unity to serialize one of your private fields you can add the SerializeField attribute to the field.

Unity will serialize all your script components, reload the new assemblies, and recreate your script components from the serialized versions. This serialization does not happen with .NET's serialization functionality, but with an internal Unity one.’

(Unity Technologies, 2013)

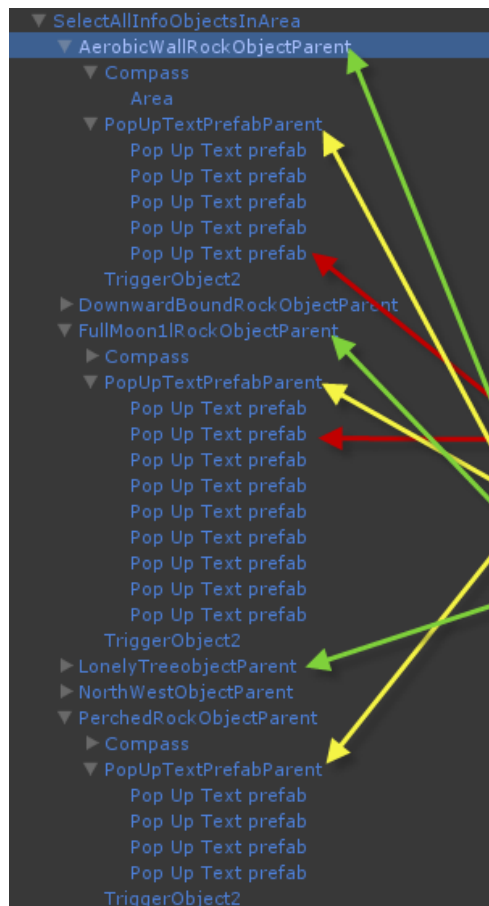
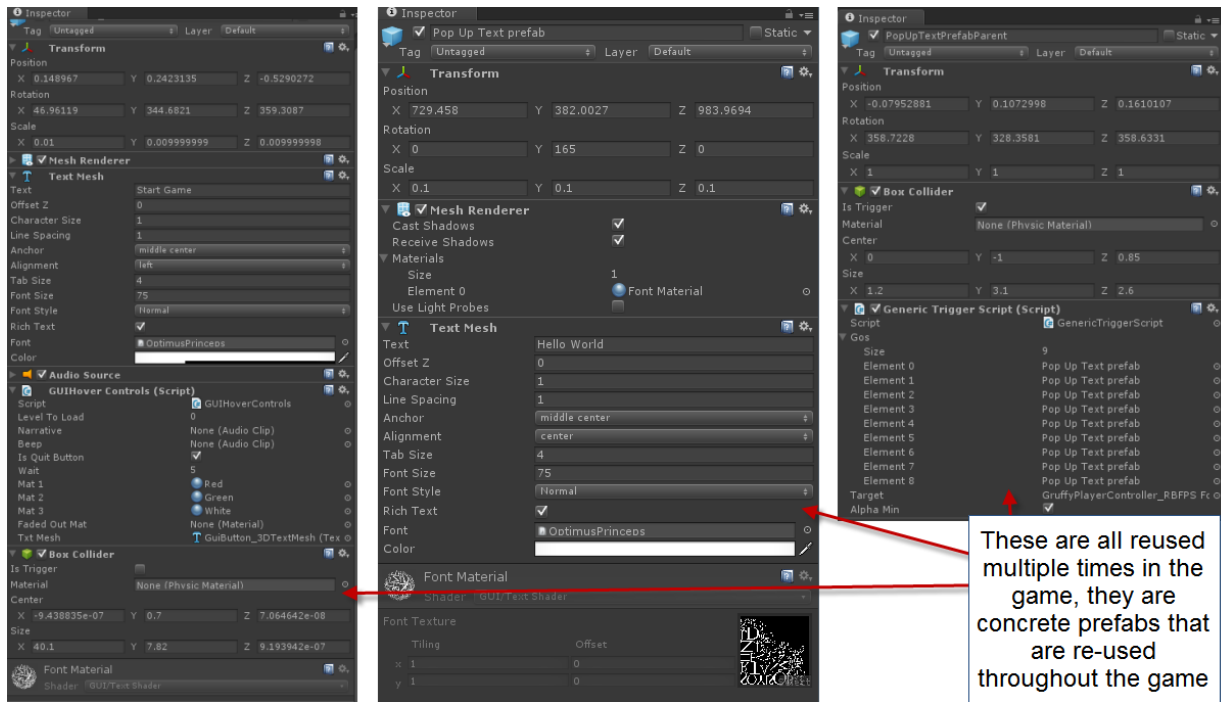
Unity does exactly this operation as a matter of course to any publically specified variables of any class in a project.

- Apply scripted reusable elements to game objects with similar functionality or usage but may differ in its resultant outcome.
- Prefab game object to ensure a base prefab for adjusting to specific needs of game objects in the build.

This method of building prefabs moves to demonstrate utilization of “base” prefabs in the same manner as a derived class might inherit fundamentals from its base (parent). .

Prefabs are developers key to designing the system right at the tangible visual world end of things and they most often follow the singleton pattern mentioned above. They have succinct operations applied independently from the game object’s component and this can be rolled out as a template by prefab. (in this case, the left side represents the 3dTextStart Screen and end Credits controls, the middle one represents all the 3d text logic and mechanics in game, the right hand side is the middles parent prefab that houses the script to operate on the child text prefabs assigned in the inspector.

These last two singleton prefabs might appear like a factory, and in a convoluted way they become such when applied to the game space as a tangible thing that is manufactured all over the game area. But, they are not as they are not reliant upon each other and can provide separate functionality in their own rights, needless to say: The middle prefab still functions perfectly well without its parent holder and the parent can operate on any set of 3d text meshes, not only prefabbed text mesh components alone. – they are not related.



We can see the use of the same prefab multiple times in the hierarchy.

These are themselves prefabbed into a parental prefab.

If you note the screenshot's details it can be further ascertained that **"PopUpTextPrefab"** is in fact a child of **"PopUpTextPrefabParent"** which is a child of the **"ObjectParent"** prefab that hold all the necessary components to fulfil the action of triggering interaction concerning that game element (In-game this is dealing with 3d text mesh pop ups around the area)

As a caveat to be continually aware of, it was discovered that prefabbing in this manner demands highly normalised prefabs with their core functionality solidly laid out, as any attempt to apply solid changes to at a later stage can potentially disrupt other aspects of the build that may have been marked completed.

6.1.2 Singleton Design Pattern in the code

Because the singleton pattern is a common pattern found amongst unity components as the Unity way often denotes a single operational script that acts out a set of self-contained operations upon the instance calling the script or by a message passing system to or from another scripted object.

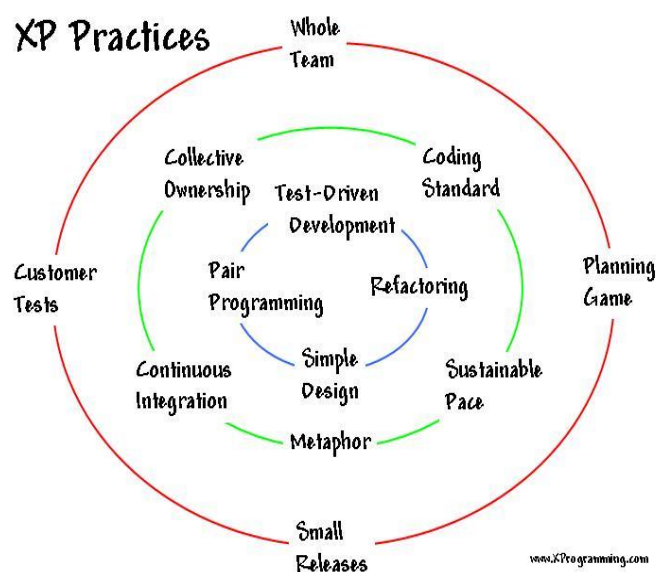
An example of this in the prototype is found inside the “End Credits” class. Which simply deals with handling a move from one level to another based on that components trigger conditionally matching the argument.

The End Credits class is located in the code section of this document’s appendices in Appendix D under the same name but here is an example of it for clarity.

```
using UnityEngine;
using System.Collections;
public class EndCreditsScript : MonoBehaviour
{
    [SerializeField]
    Private String level;
    void OnTriggerEnter(Collider col)
    {
        if(col.gameObject.tag == "Player")
        {
            Application.LoadLevel (level); /// end credits scene
        }
    }
}
```

Very short, very simple, does one succinct, concrete, thing as a self-contained process and can be attached to multiple objects without

5.1.3 Conformance to XP



The project was not team based, it was a single effort and as such the XP cycles in the image above denoted areas where compromise was undergone to secure faster development turnaround.

The following practices were adhered to in the final prototype deliverable:

Coding Standard – The standard set was to ensure C# was used throughout, this was wholly achieved as the Image effects package in Unity comes as JavaScripts. For the next iteration, All JavaScripts will be removed and refactored into code under the C# language, where it has already not been done.

Customer Tests – Phase testing was carried out under incremental release tests that were essential for feedback based improvements.

Small Releases – This alongside the above saw the product released over three iterations in the first instance and towards the end saw 5 additional builds that dealt with minor changes such as in-game typos or new conflicts arisen from code refactoring.

Simple Design & Build - To design and build a 3D experience (through delivery of games) was found a difficult process but, Through continual feedback by testers, based around the current status of the build at the time, various elements to the design and build were indeed simple to achieve.

The project managed to be quite a heavy load in the early stages due, in no small part to the challenge of creating an experience such as this as bespoke as possible to ensure the work was my own at every possible turn. It could be argued here that fully leveraging third party assets would have possibly increased the preparatory time that the project underwent at the initial stages, mostly due to taking the reference shots, and researching historical and folklore value to the area. Not to mention the wealth of preparative knowledge needed to create a successful prototype in Unity 3D (though many have before, so it should be possible was the mantra).

Refactoring – Glad to report this was a success where possible and all bespoke code can be found in C# except in the Image effects and Particle third party plugin, though technically the particle effect is under ownership as it was paid for and no identification in the code specifies any different, however, in the interest of honesty, that code is not mine and is in line for refactoring completely to C#.

With respect to refactoring the designs at any stage in the development, these seemed to change in an organic way. Through initial testing with colleagues and friends, the game grew from having a single objective - experiencing the environment, to having, multiple objectives that appeared essential to keeping interests alive.

The game had a final casual set of challenges to undertake, though arbitrary, and the environment could still be enjoyed without the pursuit of either the collectable bones or areas to get information from.

Collective Ownership – The code found in the back of the documentation in Appendix D, based upon all parties being in agreement should be released as an open source available package, meaning the code is subject to distribution and changes without copyright infringements on the final source. However, the object code produced when compiling the game will be under ownership by either or both, Plymouth University and this projects author; myself, Gareth wright.

As the project was a sole undertaking, respect to collective ownership based on collaboration, no others can be identified as such, though the project mentor and supervisor, Dr Dan Livingstone played significant role in the direction of project delivery through interim meetings where possible and appointed such.

5.1.4 Brief Project Scope summarisation

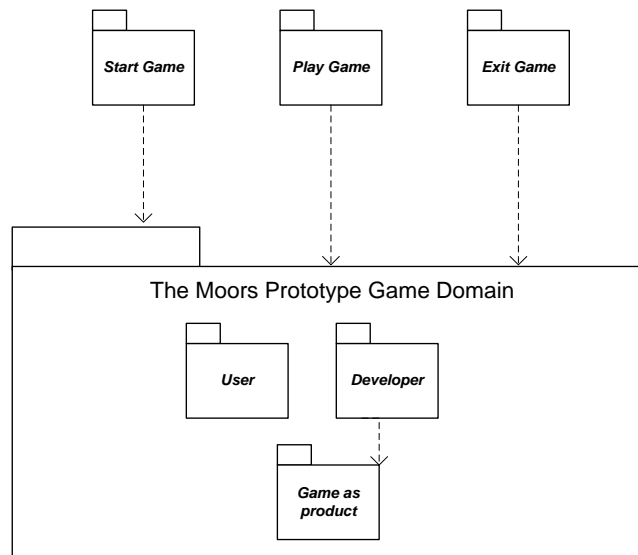
As was composited from previous research laid out in the second and first project deliverables, the scope saw many iterative changes during the cycles of development. Most notably, the direction to change the theme from a heavier biased sub-horror genre game to the experience driven prototype seen in the game today. As time moved forward improvements were undertaken to increase the prototypes appeal across a broader range of user, and decoupling it from the chilly experience of discovering Kitty Jay's bones as the sole purpose to playing through the "game".

The final deliverable sits well with the changing scopes, and can be seen to do so in that it has achieved what it was set out to do, and though evolved from the original earlier concepts to what it is now, still delivers a proof of concept based on the original ideas laid out in the PID and any subsequent documents.

5.2 Project Documentation Routes

The below package design represents the system at the highest level. It is discussed alongside other documentation in the second project deliverable report and found as such in the appendices of this document.

5.2.1 Package Diagram



The package diagram is by no means definitive, in that it does not define the intricacies of the system but rather provides an abstraction of the overall product. From this point, the system can be drilled down to develop stories based that forward themselves to become requirements.

5.2.2 User Stories

User stories were gathered from various interested parties at the very start of the project timeline.

Below shows the final user story set that delivered that saw the final prototype delivered thereafter.

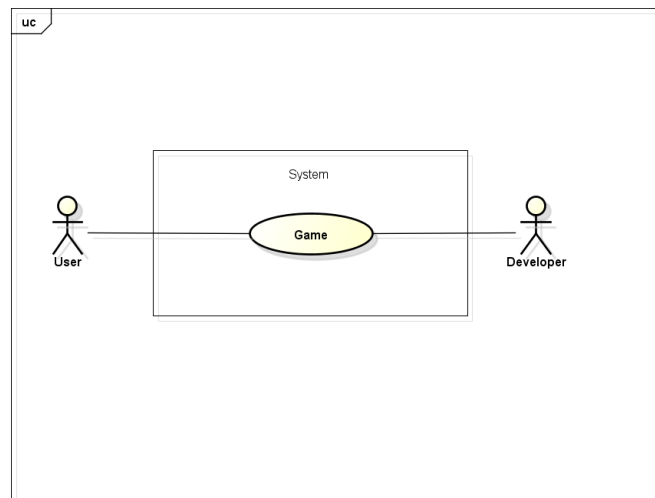
Final release plan– Beta 0.1

As a user I thought the day night cycle detracted from the game some weird events when the sun goes down and the moon comes up, it seems like a rushed afterthought and maybe it could be better	Conformance <ul style="list-style-type: none">• Removing Day Night Cycle for reimplementation after project to increase efforts to create better cycle mechanics for the day night switchover, this includes the potential to use shaders to swap cube maps that would suit the cycle better for both day and night
As a user I wasn't sure about whether I collected something or not as it took me a while to see the collected stuff at the top of the screen	Conformance <ul style="list-style-type: none">• Sound added to each area item to visit in the game• Each sound is subtle but unique to the type of collectable• As such, this has also been applied to the bone items in game.
As a user, when I started playing I didn't know the controls and I missed the secret introduction on the start screen.	Conformance <ul style="list-style-type: none">• A fixed auto scrolling GUI intro text is displayed upon area entry.• This identifies the premise and game controls immediately alongside instruction to pause the game where the options provide both a review of controls and settings to adjust fog and gamma corrections too.
As a user, on the sign post it wasn't obvious what those images were for	Conformance <ul style="list-style-type: none">• Added 3d text meshes to each element on the sign post to clarify what needs to be done
As a user it was not clear to me that I was blocked in , I thought I could explore all over but that wasn't possible	Conformance A fixed auto scrolling GUI intro text is displayed upon area entry

5.2.3 Use cases

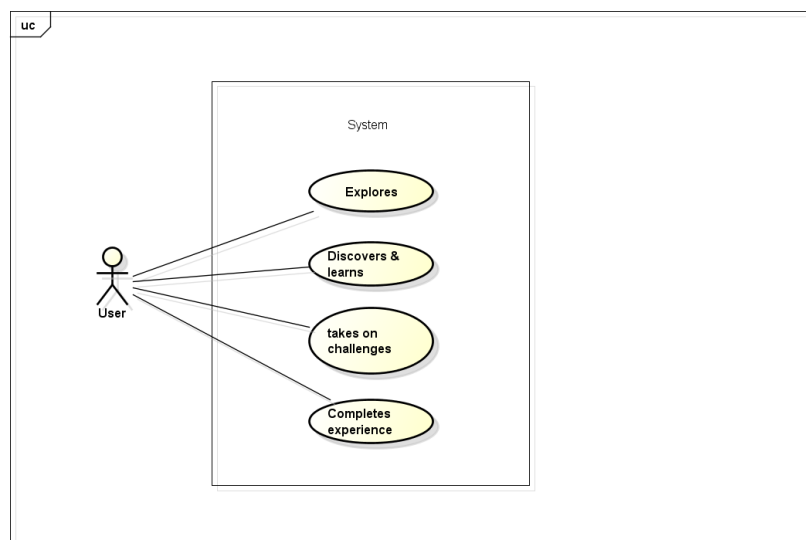
Use cases were developed to lay out client and developer needs at the start of the project and will continue to be the foundation for additional aspects added to the prototype at any later stages.

The below use case shows the expected of the prototype development. Using these, it could be established from the off, the core elements expected from the prototype would be a game product.

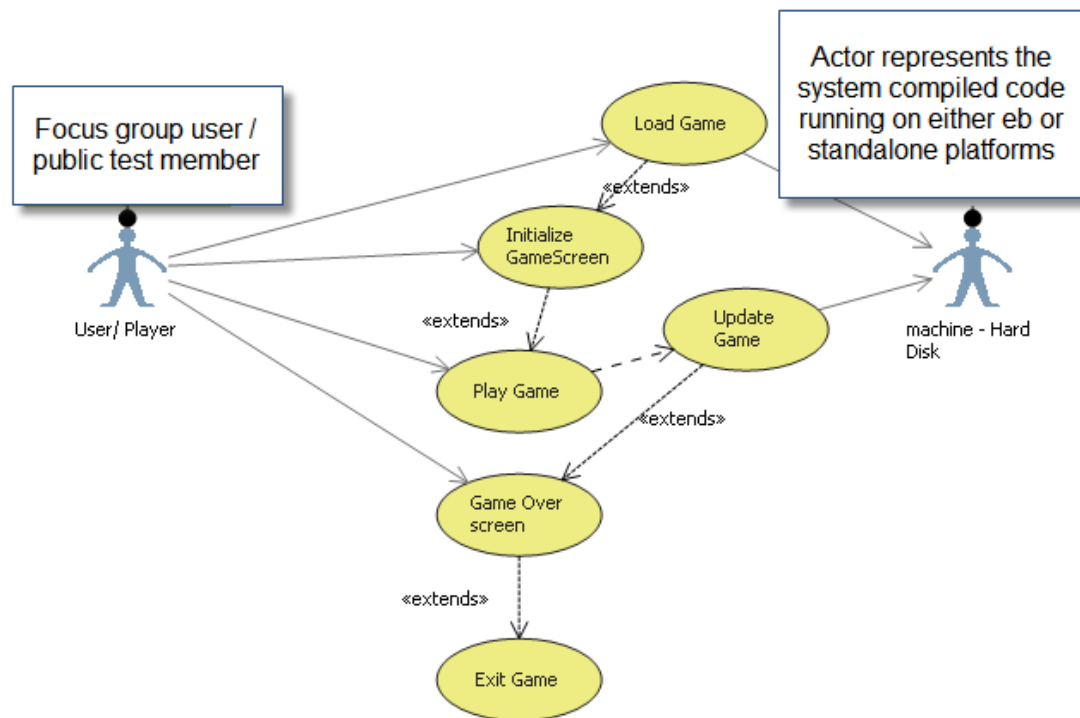


The below use case simply identifies the needs of the client from the system.

This is often accompanied on the opposite side showing the system or developer denoting what aspects they are involved with to bring the system to fruition. Here though, the developer has been omitted as all areas are covered by the sole developer of the project and the use case better serves to focus on the client recipients needs rather than developer responsibilities in this case, due solely due to that fact it is being developed as a one man army attempt, there are no other real key responsibilities assigned outside of the developer.



This below Use case exemplified the system delivery and key roles from both user and prototype (system)



Use cases were an effective tool, though at the initial stages it could be forgiven to think that they may not be necessary, but it is the concept that carries through to later use cases that can greater simplify complex or eve not so complex system responsibilities, which would further assist in keeping track of status as a comparator later down the development line.

5.2.1 Class Definitions and Diagrams

Visual Studio 2010 tools were used to document the systems class definitions and diagrams.

Below is an example of what they look like but the bulk body can be found in the Appendices section under **Appendix C** where class definitions and diagrams showing class attributes and operations are there to show the documented system.

Name	Type	Modifier	Summary
Methods			
OnTriggerEnter	void	private	Raises the trigger enter event.
OnTriggerExit	void	private	Raises the trigger exit event.
Start	void	private	Start this instance.
<add method>			
Properties			
<add property>			
Fields			
destination	Teleporter	public	The destination.
teleported	bool	private	The teleported.
<add field>			

Teleporter
Class
→ MonoBehaviour

Fields

- destination
- teleported

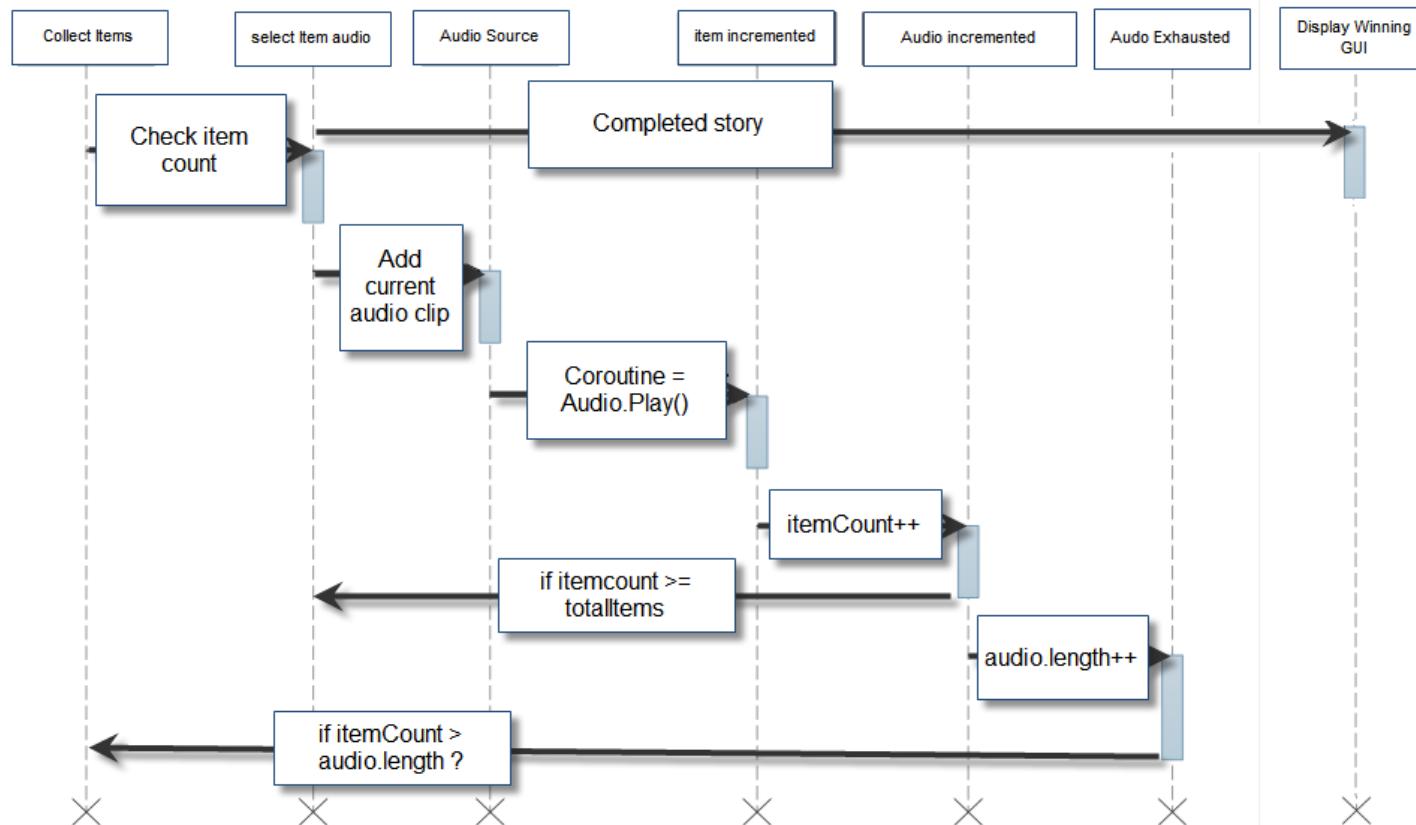
Methods

- OnTriggerEnter
- OnTriggerExit
- Start

Due to modern IDE documentation tools in Visual Studio 2010, this has been chosen as it is legible to read and the class diagrams adhere to the principle of identifying Fields and attributes as can be found in hand drawn UML class diagrams.

5.2.2 Sequence Diagrams

These can be found in the back of the Documents Appendices section under Appendix

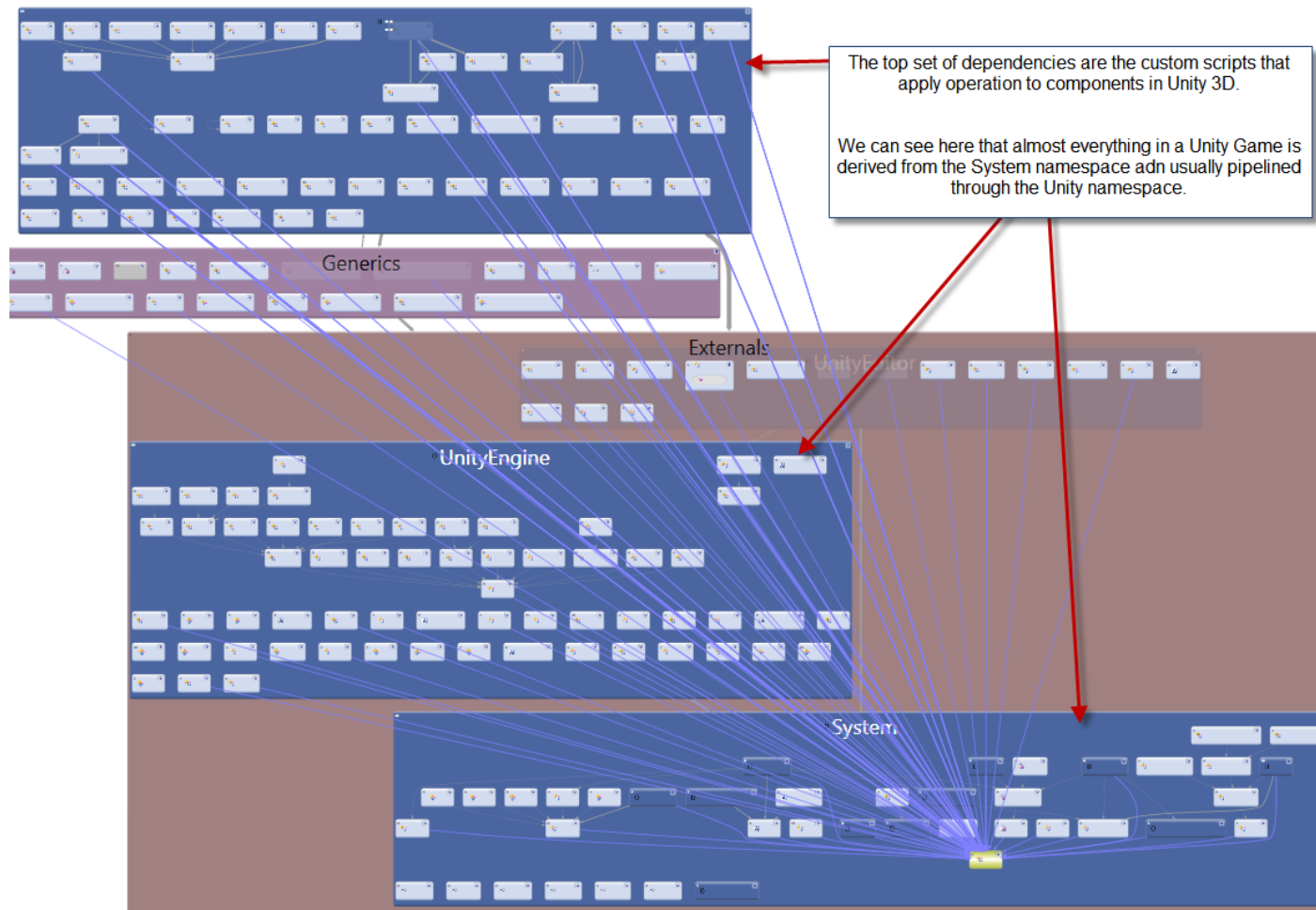


Above is an example of the sequence diagrams that were created to show a system state sequence of events either firing or returning a conditional argument of some sort

The above sequence represents the sequence run when an item is collected. This particular sequence pertains to the bone items in the prototype and actually falls inline to most sequences the game carries out due to the Unity way, bringing forth component oriented strategy to constructing game objects; it often denotes a single operational sequence per component, though tis could be further broken down into further individual method operations.

There further Sequence diagrams available at the back of this assignment in the Appendices under Appendix

5.2.3 Class Dependencies – An Overview

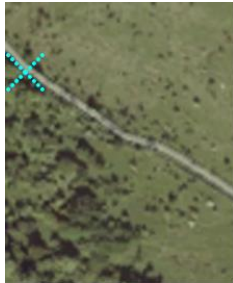


In the above we can see the connections made by derived classes calling to base elements in both the System and unity engine namespaces. Mostly it can be denoted that everything in Unity, as is found in C# (.NET), everything comes to life at the System namespace and is derived from "Object". A larger version is available at the back of the document in the Appendices under Appendix.....




6 Game Design

6.1 Level design

Due to the nature of the location being both a single level and real, the design carried out was taken from an overhead shot created in Global Mapper™ - a commercial tool used to obtain geospatial data and topological map of overhead views of Earth.

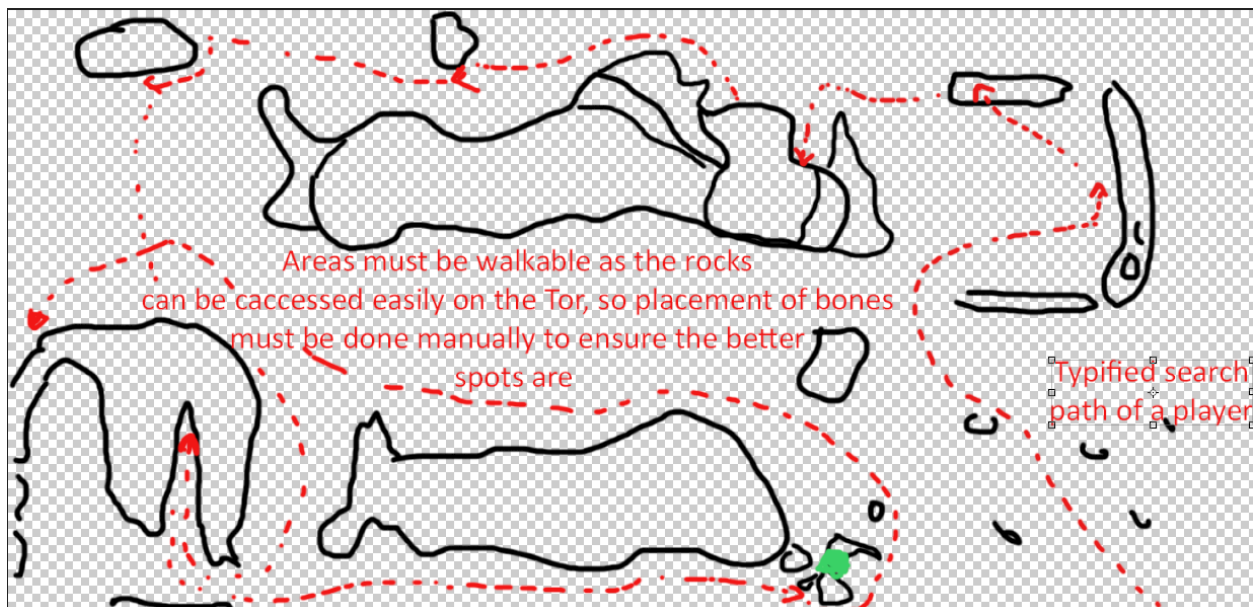


Kitty Jays Grave Area

-  Bone placement
-  Geo Caches
-  Areas to visit



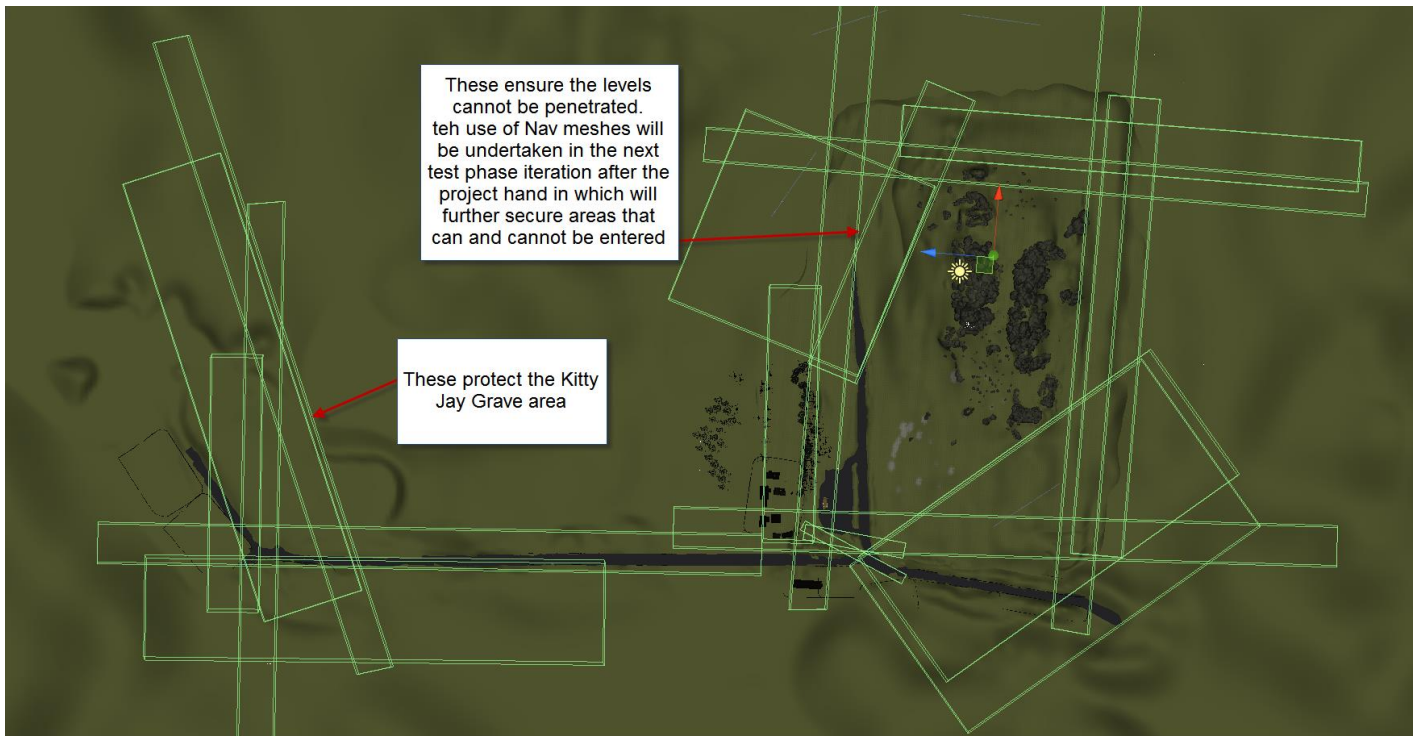
Hound Tor Area



Hound Tor area mean average typified player denoted from user testing sessions where 10 people were asked to play the game in a quiet dark room with headphones and left to their own devices. As a bonus, this inadvertently set the average in game time too, resulting in an average play time of between 15 – 30 minutes. This is discussed later in the conclusion to this document.

Further level designs were implemented to ensure areas could not be entered into. As such, these areas were contained through the use of Game Object component colliders which were linked to a prefabbed text mesh that fired itself active when the player gets too close to the barrier zone.

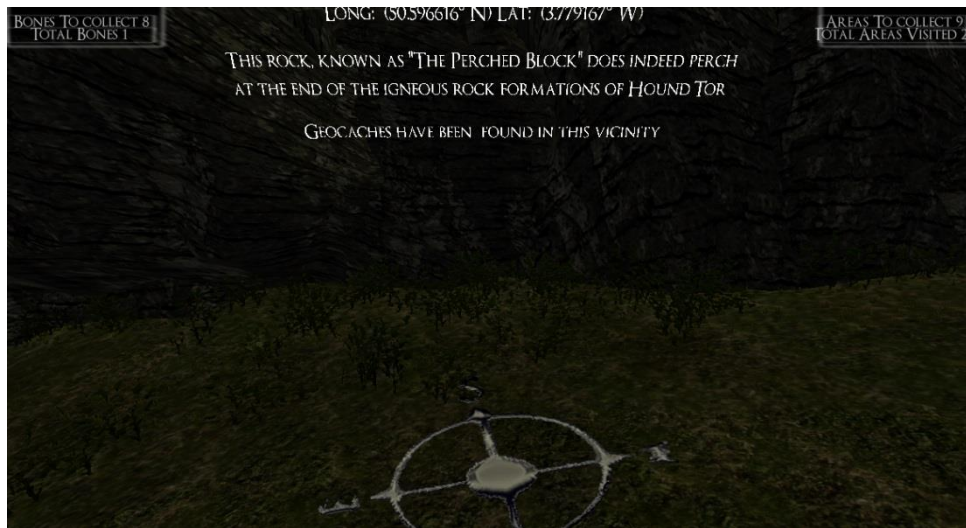
Below is an image describing this visually inside Unity's editor.



Furthermore, the image below depicts how we see this effected in-game.



The following screenshots example some areas in game where the items are placed according to the level design specified in the above image overlay of the hound tor area



This image represents the area of information found at the perched rock



This image represents the Geo cache found near the same area as above



These bones are in the game at one of the blue X spots by the first large rock set "Downward Bound"

5.2 UI design

6.2.1 Start Screen UI



Concept



Resultant outcome

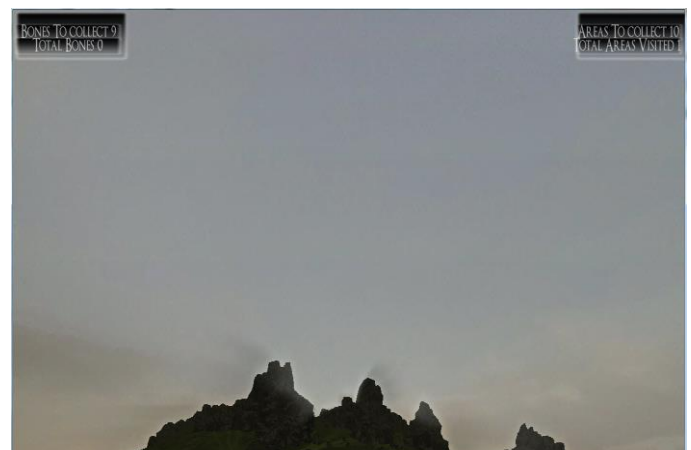
As can be seen in the image found below, the UI controls are specified for the user to familiarize themselves with and by which, denotes how the user interfaces with the game at the start screen

6.2.2 In-Game UI

Below demonstrates the in-game UI, as seen by the user

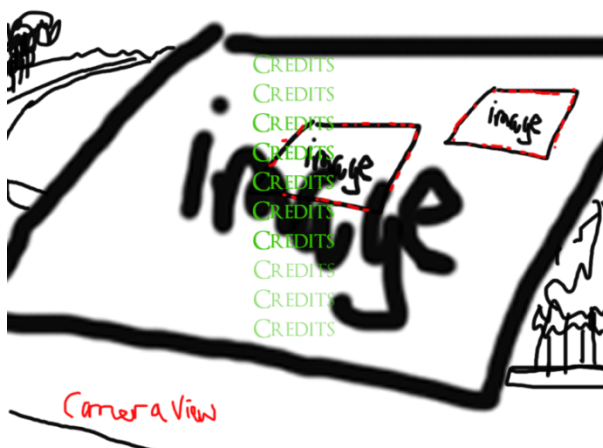


Concept



Resultant outcome

6.2.3 Game over UI

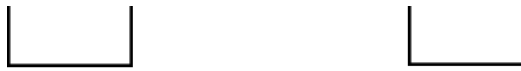


Concept



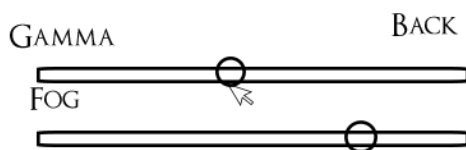
Resultant outcome

6.2.4 In Game options

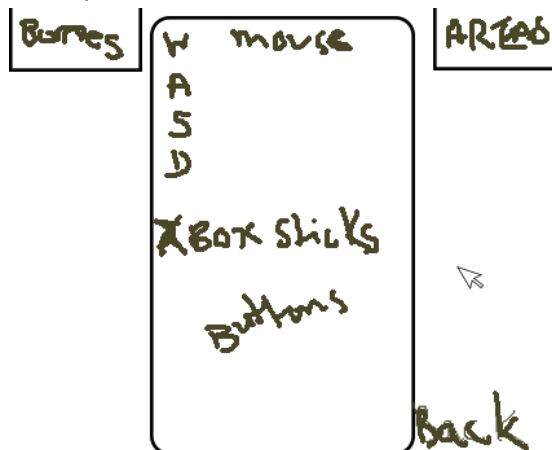


Back
Options
Controls

Concept



Concept



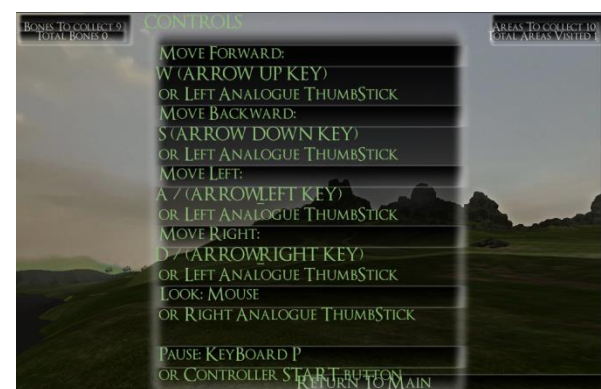
Concept



Resultant outcome



Resultant outcome



Resultant outcome

6.3 Elements of Design used to create Game Assets

As briefly identified in the earlier section, Section 4: Technologies there is further discussion related to the products used in the project

6.3.1 Autodesk and Blender

Using a composite of software from Autodesk to find the happiest medium between them, experience concluded that it is the opinion of this author that the 3DS Max suite from Autodesk is by far a superior package to handle every modelling aspect overall. However, it is considerably complex for a beginner to modelling and UV mapping anything. A much more suited software package was found in Maya from Autodesk, where this felt like it was built for game model and environment development as its sole purpose. Further to this down the line Blender 2.6 was discovered. It's a free modelling software built open source and has Unity plugins available to assist in importing animations to Unity from blender models/scenes. It is again, like many modelling software, finicky to initially get anything achieved and relies upon heavy use of keyboard shortcuts to navigate it with success. It is clear that these programs are involved and ask for some deep commitments to get little decent results in the first instance.

Two software programs of the note, that certainly helped assist this project's workflow, with respect to modelling software was both the commercial software from Autodesk called MudBox and the free ZBrush creators ,beginner, modelling program known as Sculptris Alpha. – Both MudBox and Sculptris are like the “Tonka Toy” version of their bigger modelling software siblings with a lot of the finicky optimizations techniques and strict UV mapping rules found in each of the previously mentioned programs, catered for you with these to some extent.

There are screenshots in **Appendix E**, identifying the above software packages.

6.3.2 Adobe Photoshop CS5 (All Artwork & art editing– hand drawn scans and digital)

Adobe Photoshop is viewed as a fore frontal, industry standard software for digital image and art editing bitmaps, textures and now 3D.

Due to previous experience with Photoshop prior to the project, this choice of software was best suited to the task ahead, namely:

- General handling of exporting correct file formats for saving, additionally, PSD is a preferred image option in Unity 3D's engine, identifying it as an easy consideration for professional digital artists to quickly present their images interactively.
- Reduction of images sizes to conform with the game's dimensions.
- Additional image enhancements, including deeper image details like shading and colouring.
- Optimisation for lower image file sizes
- Alpha transparencies on images using .png or .psd with Unity 3D

In Appendix E of this assignment, one can find a screenshot denoting the user interface and an annotated explanation of its use.

5.3.3 FL Studio, Ableton Live 8, Line 6 –GearBox Software

(Audio mixing and editing)

- All audio was mixed and edited between Ableton Live 8 and FL Studio 10 with the use of recorded audio and VST's (Virtual Studio Technologies) for the sound effects that take place during the game and the in-game ambient music.
- The game's OST was recorded and layered in the DAW FL Studio 10 (Digital Audio Workstation Fruity Loops Studio V10) to create the soundtrack to the game.

- Leveraging previous experience in these software packages allowed one to write the song score on the fly and record each affected guitar layer into Ableton Live 8 for cutting and looping and on to FL Studio to composite a finished soundtrack and other in-game audio textures.

Appendix E of this assignment, one can find a screenshot denoting the FL Studio user interface and an annotated explanation of its use alongside one for Ableton Live 8 too.

6.3.4 Breakdown of Scene Artwork and Models Created

Kitty`s Bones

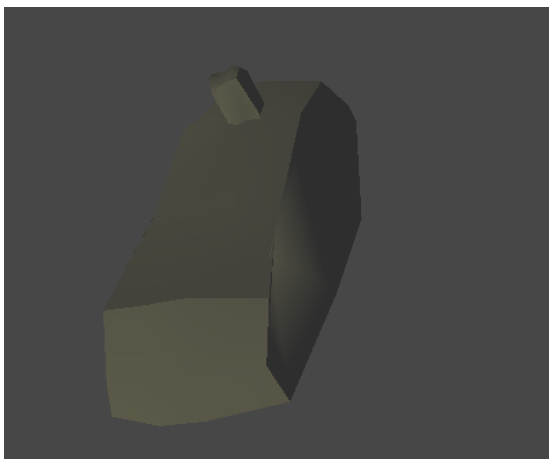


Modelling it

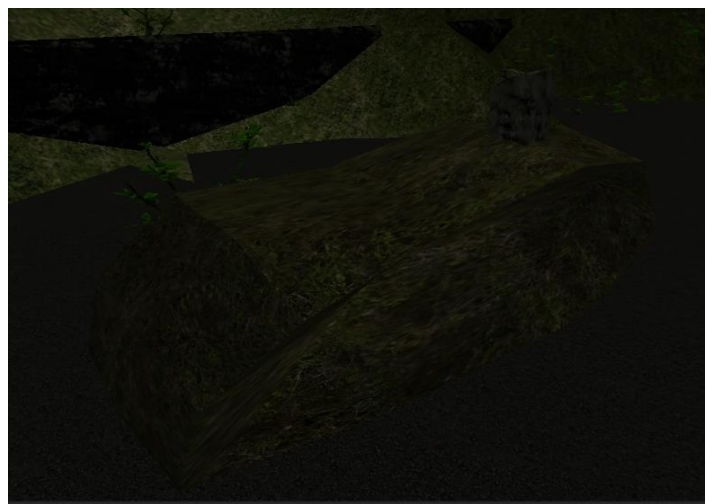
Resultant outcome

The above image is indicative of the rest of the models parts for the skeleton and as can be seen the resultant outcome was quite satisfactory. It was created from an online tutorial which was followed to create the whole skeleton. Then it was realised the skeleton had to be separated which caused numerous problems in 3DSMax as the model was made heavily connected to each part by parenting as the tutorial was really designed for animations as the finalised result. At this juncture, I only needed the model and felt another time would be appropriate for animations in modelling.

Kitty`s Grave

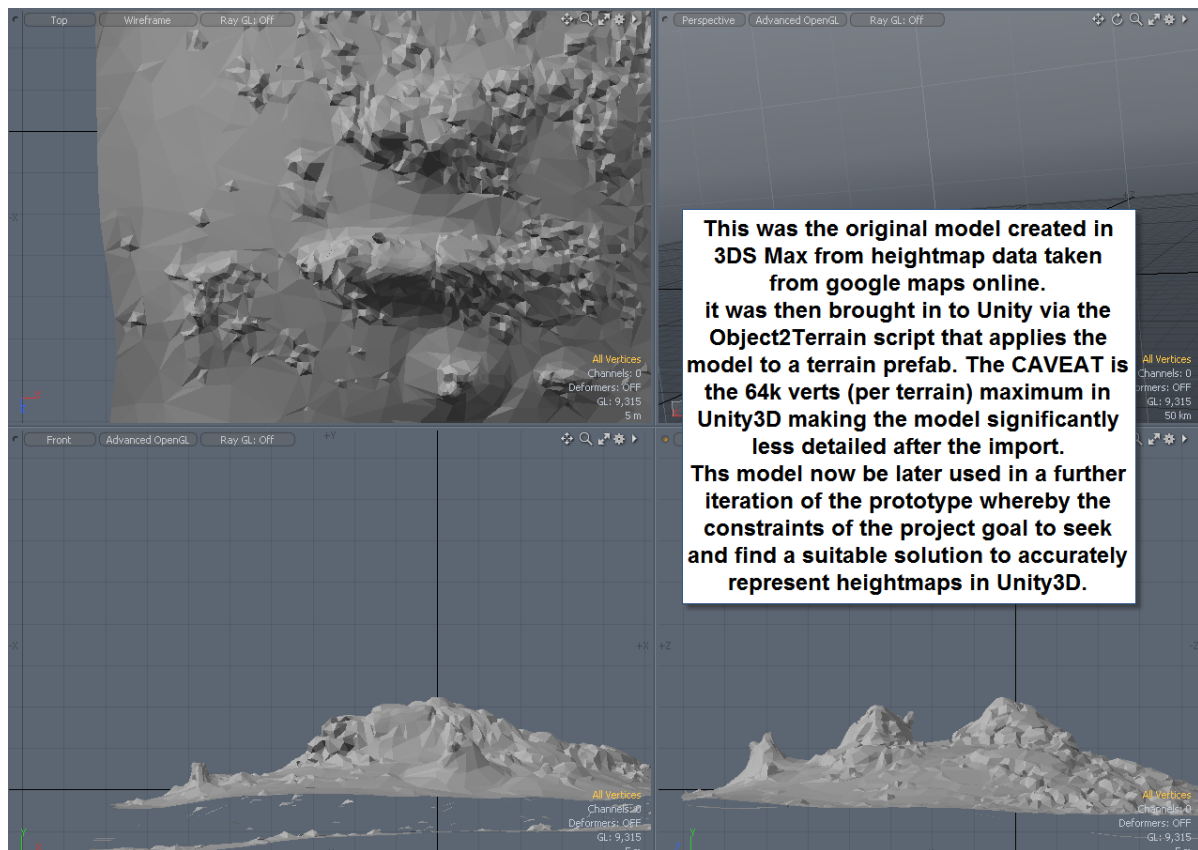


Modelling it



Resultant outcome

Houndtor Rock Scene



The alternative was as stated in the image above, bring in rock models to the environment and make do with what Unity could best reproduce through its terrain component.

Using reference photography caught first hand by visiting the area on several occasions over daytimes and evenings, including all night at one point to see how it could be represented in the game for the second iteration.

Below has identified some of the reference shots taken there which went forward along with others to supply enough reference that it could be modelled, well attempted as such.



Kitty`s Crossroads

This area was a last minute thrown together aspect that suffered due to time constraints and such.

However, it was in the final prototype deliverable so it should be counted, below is the Crossroads original image and as it stands in the game opposite.



Referencing image



Resultant outcome

Particles

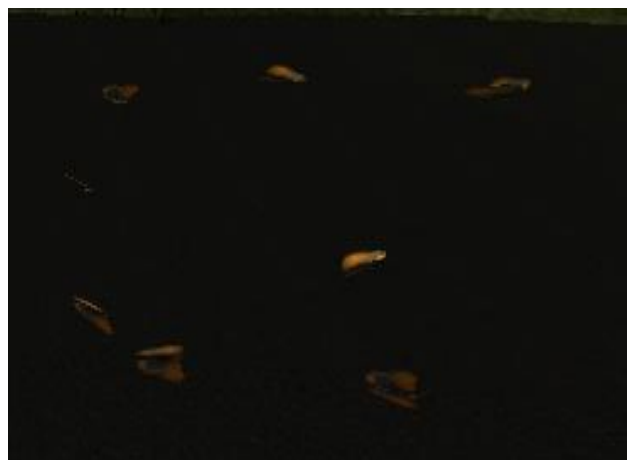
The game leveraged a particle effect afforded at the Unity Asset Store for \$1.00 giving the owner sole rights to use it. However, in the interest of credit where it is due the author, they been credited for it in acknowledgements section at the start of this document.

The particle effect was a simple one that predominantly gave me a time saver as it would mean modelling more meshes and further texturing and texture editing to get the desired effect, so for the price of one dollar the particle

effect below is in place at entrance to the game area and was an attempt to add some level of visual to the in-game entrance.



Leveraged Particle effect



Leveraged Butterfly effect

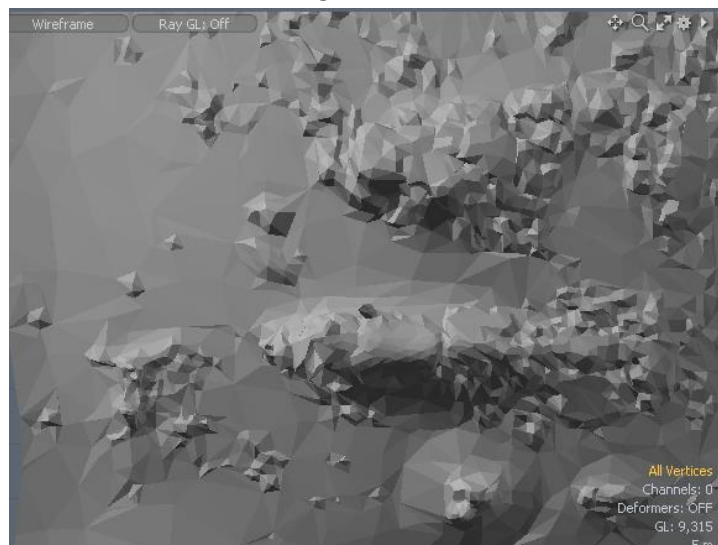
Unfortunately at the time of purchasing this, it was not noticed the scripts were JavaScript based so they instantly broke the C# only rule, alongside the Pro Image effects leveraged in Unity a swell this concluded the rule to now be defunct, though the goal would still be an aim for a later iterations.

7 Implementations

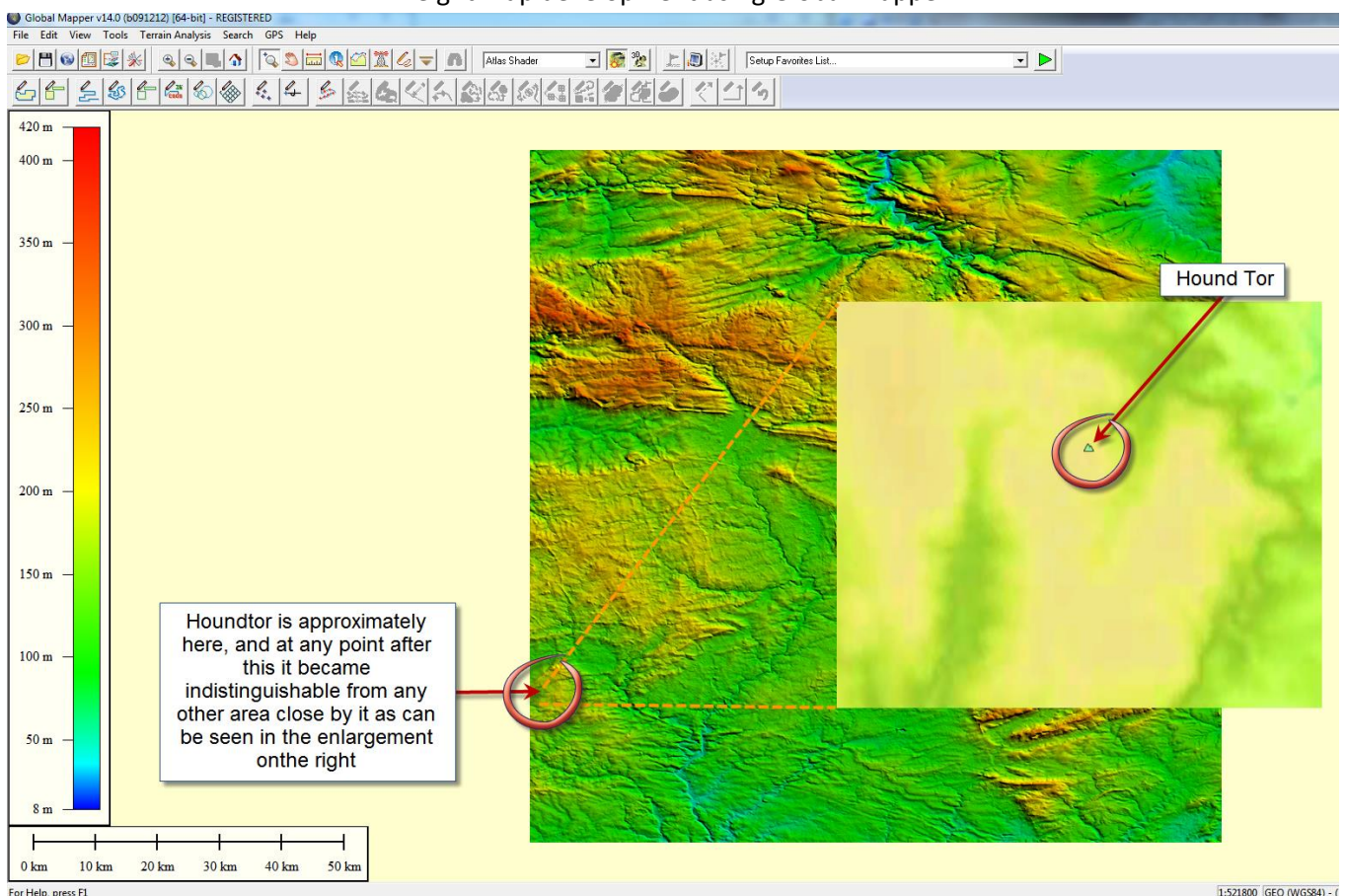
7.1 Development Phase 1

The first development phase saw much achieved by way of research into the language and environment to be used to complete upcoming project tasks like modelling and developing well in Unity3D. During this time the Terrain was created and iterated through several different modelling programs to decipher the simplest program to use for the task to completion of the models for the next phase.

Original model

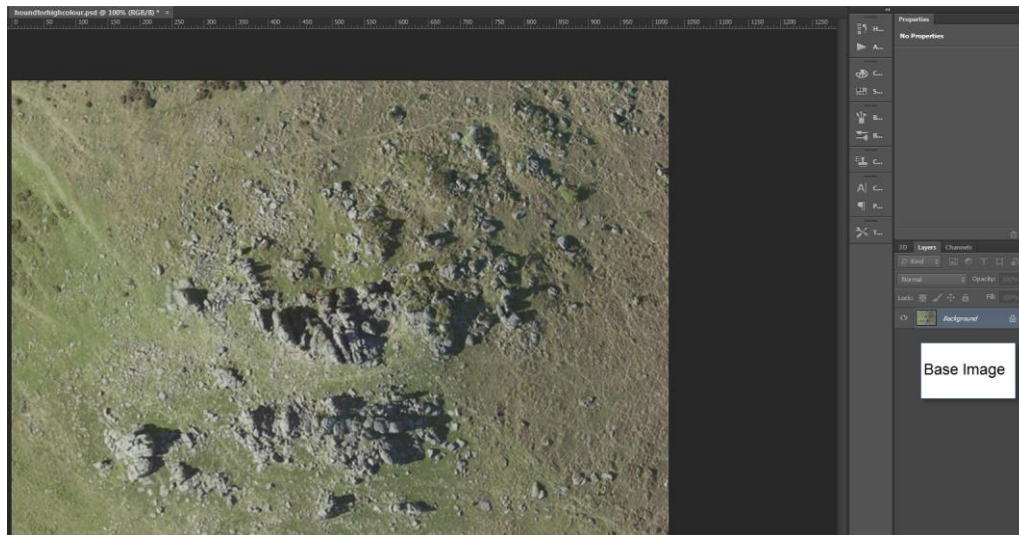


Height map development using Global Mapper

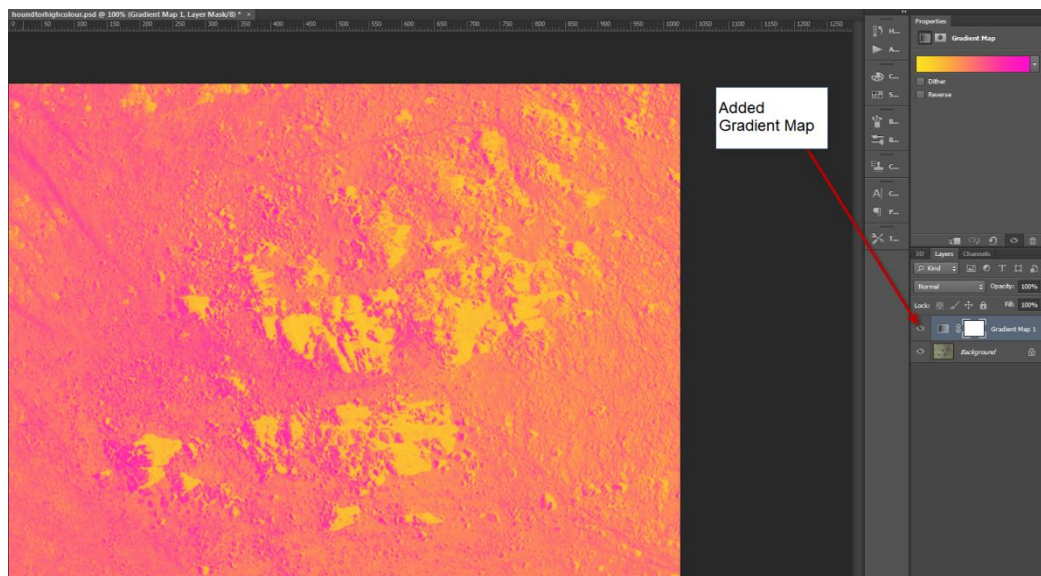


As can be denoted from the image, achieving a real location based height map was out of the question due to scaling over anything else, with a minimum 10, 000 KM² recommended for using this program. Hound Tor only manages to be under a 1000 square feet and only as a height elevation of 414 metres. The alternative then was to create a height map manually.

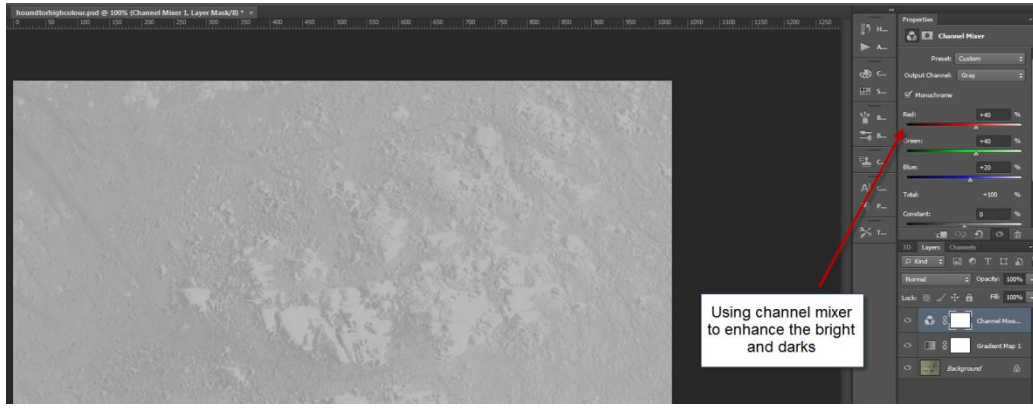
The following workflow identifies the underlying process of manual height map creation, using Global Mapper for the screenshot of aerial hound Tor and this was brought into Photoshop CS 5



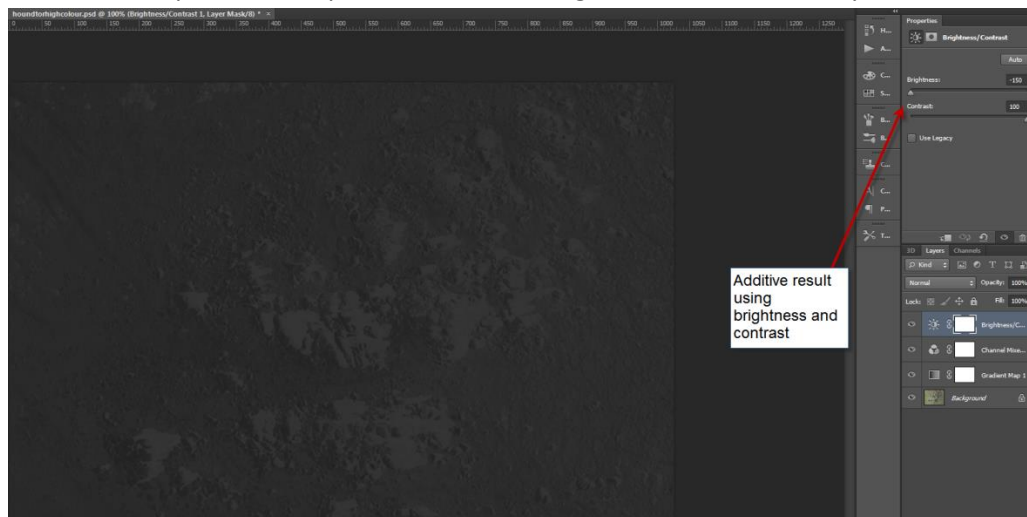
Following onward, The image received a gradient Overlay to differentiate the highlight and the low light areas of the top down hound Tor.



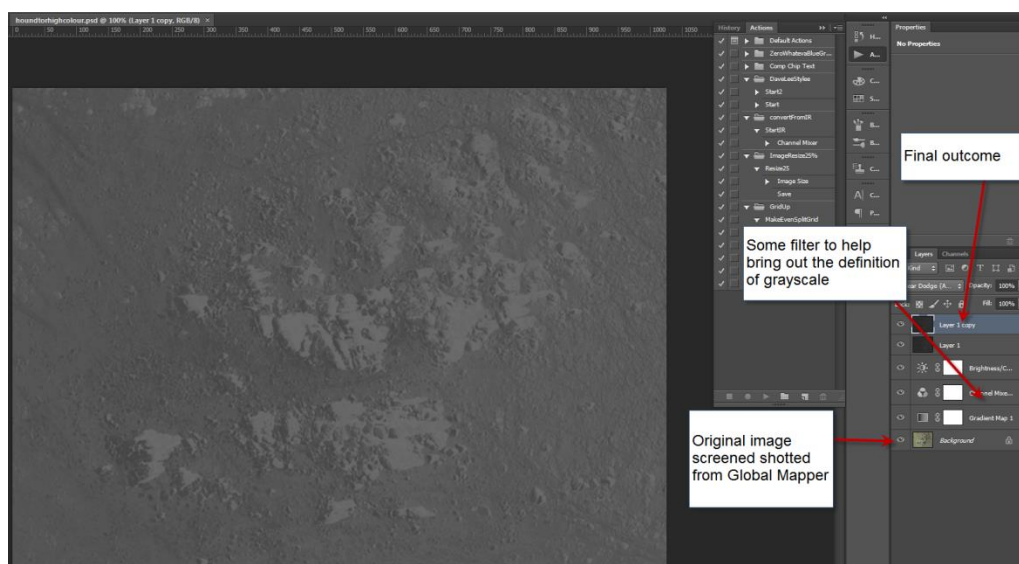
It then received a channel mixer and made greyscale



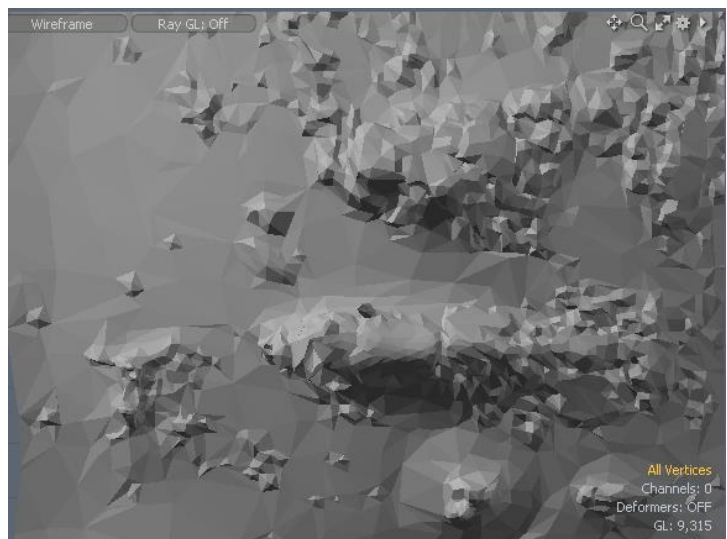
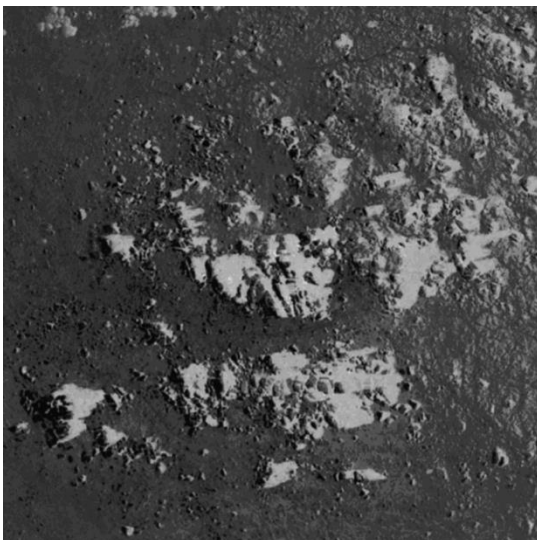
The penultimate part was to add a brightness and contrast layer to



This was the final result as a composite which moved forward to become what can be found in the subsequent image.



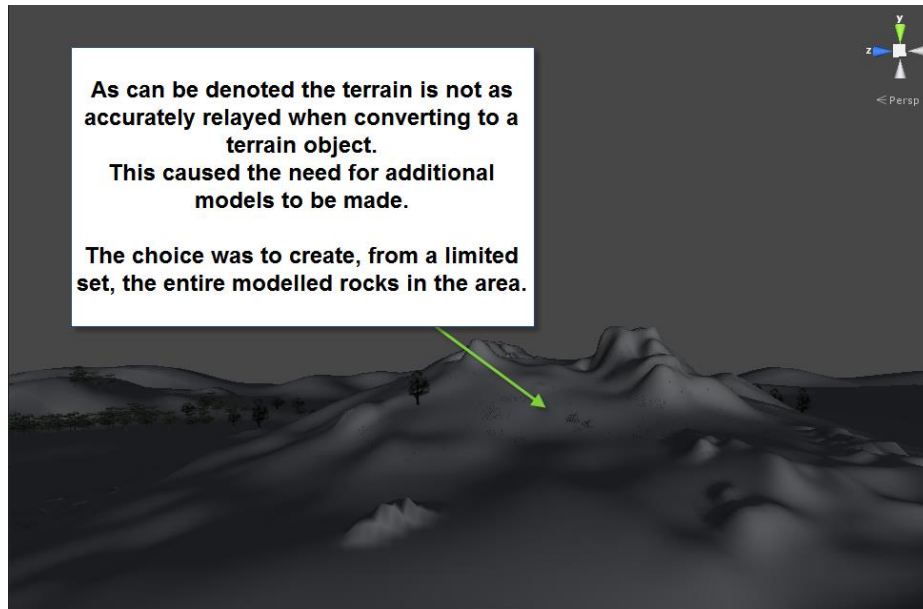
The image above was edited with brightness and contrasting layer additions until it resulted in the below left. From here, the height map was now ready to be transferred over to 3DS Max and Blender respectively for both building up the map and reducing the vertices where possible in Blender, resulting in the image found above this explanation and opposite for comparison.



Below is a screenshot showing the initial import of the model into Unity using Object2Terrain to convert it into a terrain object.

Developer, Eric Haines (Eric5h5), made the original JavaScript Objec2Terrain script that was converted into C# and which helped achieve the terrain height import a bit better, but the result were still poor based around Unity vertices count restrictions.

Final Acceptance Import



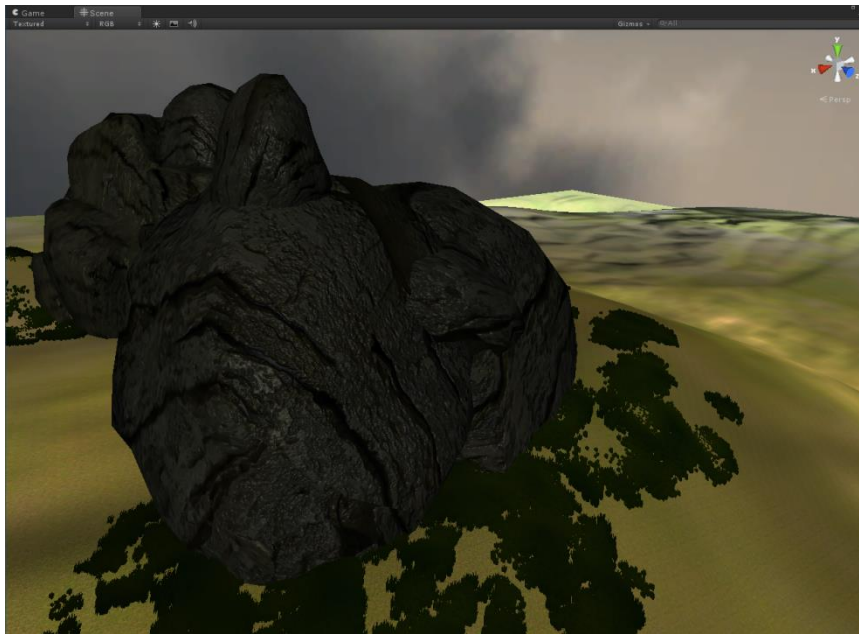
As can be denoted the accuracy from the original model made in 3dsMax from height map data did not carry through due to the 64, 000 verts maximum allowance for a Unity Terrain. The decision to move forward and use own modelled rocks to carry out the task of adding the rock formation`s character to the terrain. This was carried out in phase 2. However, overleaf is another screenshot that had been established to identify the creation of the height map that was used to create the model prior to importing what you can see in the image below into Unity.

7.2 Development Phase 2

After the decision to create the rocks, which would develop both modelling and experience in modelling packages

The result were laid out as follows and as can be seen went through multiple changes and optimisations before it got to the final set laid out at the end.

Initial rocks in Unity were nice but did not sit right with the broken lined rock of Hound Tor so further improvements were attempted



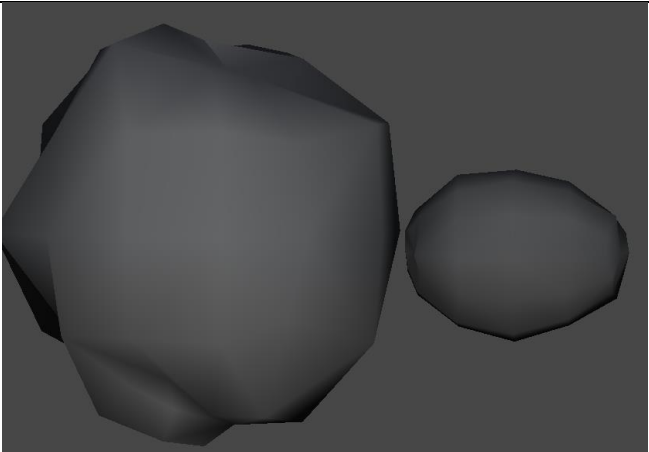
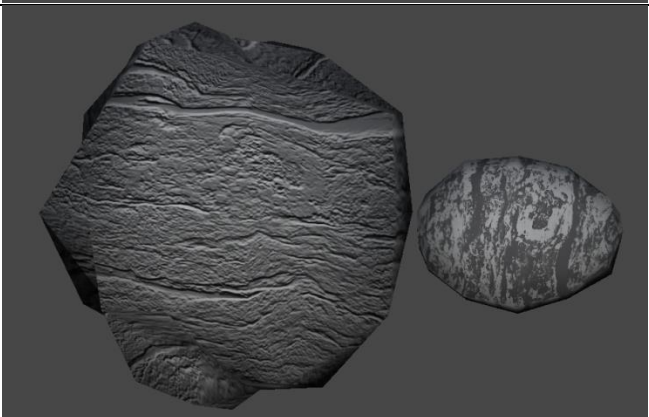
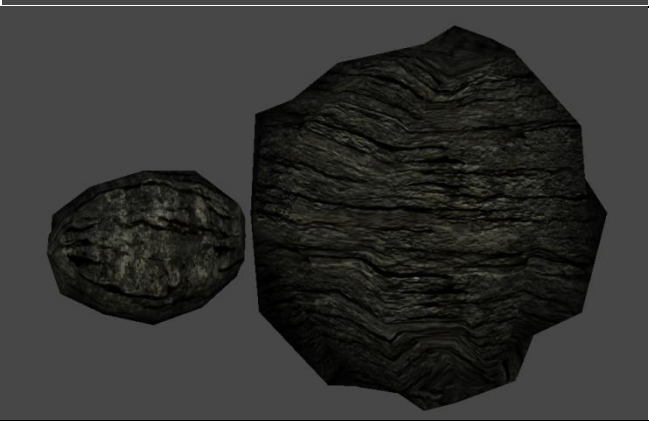
This next set were better from a distance but were not satisfactory up close so again were discarded for a better creative attempt.


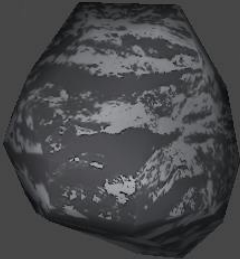




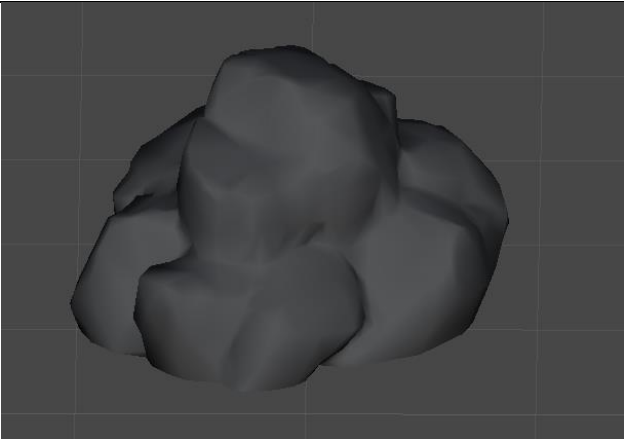
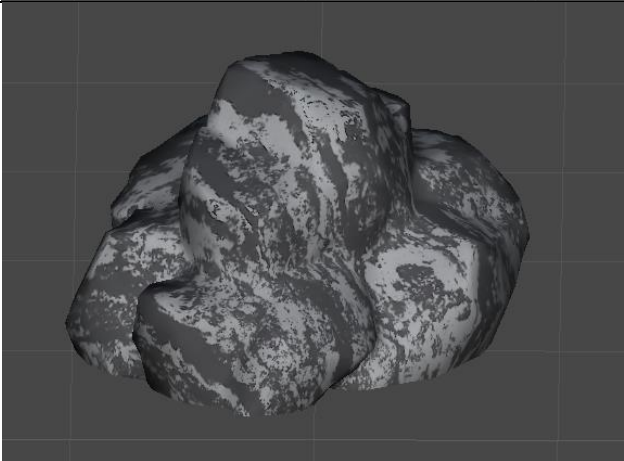
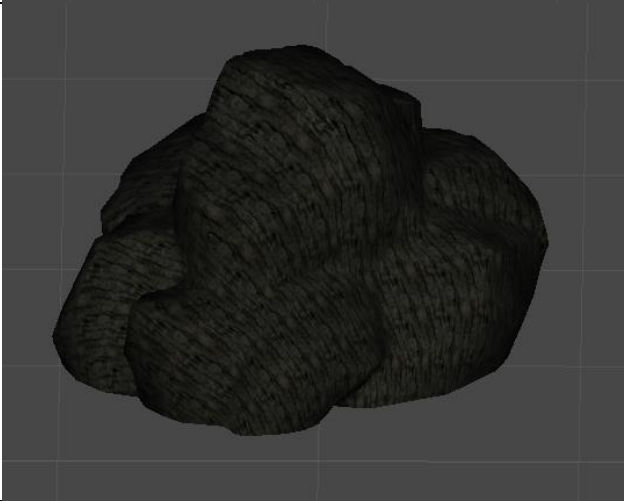
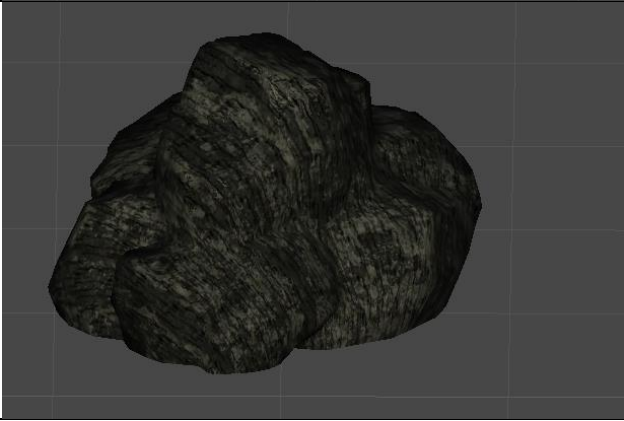
It was at this third attempt to create a rock set that could feasibly recreate the Hound Tor that finally started to yield satisfactory results, with respect to providing a convincing environment as part of the projects proof of concept outline.


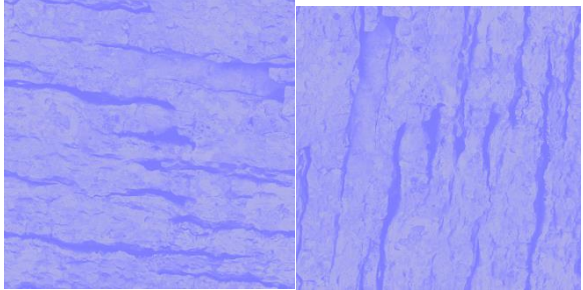

The rocks are below and are shown in the form of un-textured, normal textured and then fully textured.

These rocks, alongside some trial and error Adobe Photoshop tiling techniques for the textures, composited the final rocks types found in the game. These are not the exhaustive list but they cover approximately 50% and the remaining 50 % are mostly composites of the same rock model only enlarged or smaller with higher or lower res textures.

<p>Not textured</p>	
<p>Normal textured</p>	
<p>Fully textured with both texture and Normal with a bumped diffuse shader.</p> <p>The textures were taken first hand at Hound Tor rock and were close up and macro shots of the rocks which formed a good starting template to develop the tiled textures from with the resulting outcome opposite.</p>	

Not textured			
Normal textured			
Textured diffuse only shader			
Again, fully textured with both texture and Normal with a bumped diffuse shader			

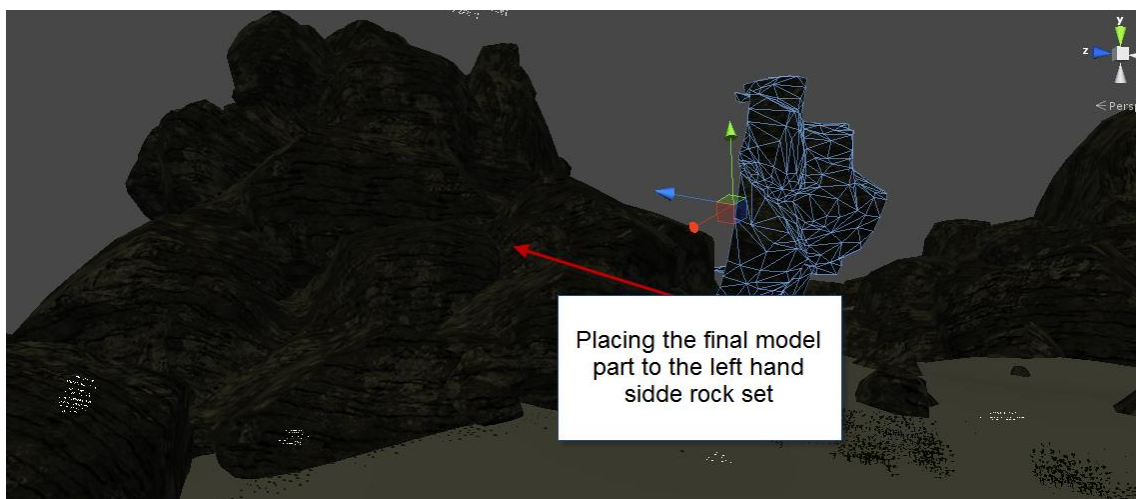
<p>Not textured</p>	
<p>Normal textured</p>	
<p>Textured diffuse only shader</p> <p>We can see here as is examples in the previous image set also, that the rock does not come off great without the addition of the bumpmap diffuse shader offered as part of Unity 3D shader utilities.</p>	
<p>Fully textured with both texture and Normal with a bumped diffuse shader.</p> <p>These rock were finally beginning to look somewhere close finally.</p>	

<p>The completed textures for the Rocks are laid out in the opposite column and are the result of my own attempts to learn tile-able texturing for 3D models. They came out okay but as with most involved processes could benefit from some real time to explore modelling software more to get better at this.</p> <p>There were two resolution types made after the texture was made.</p> <p>One at 100ppi and the other at 150 ppi alongside being a power of 2 texture at 1024 X 1024 making it the required tile-able dimensions.</p>	<p>Final Created Tile-able texture</p>  <p>Normals</p>  <p>Alpha</p> 
---	--

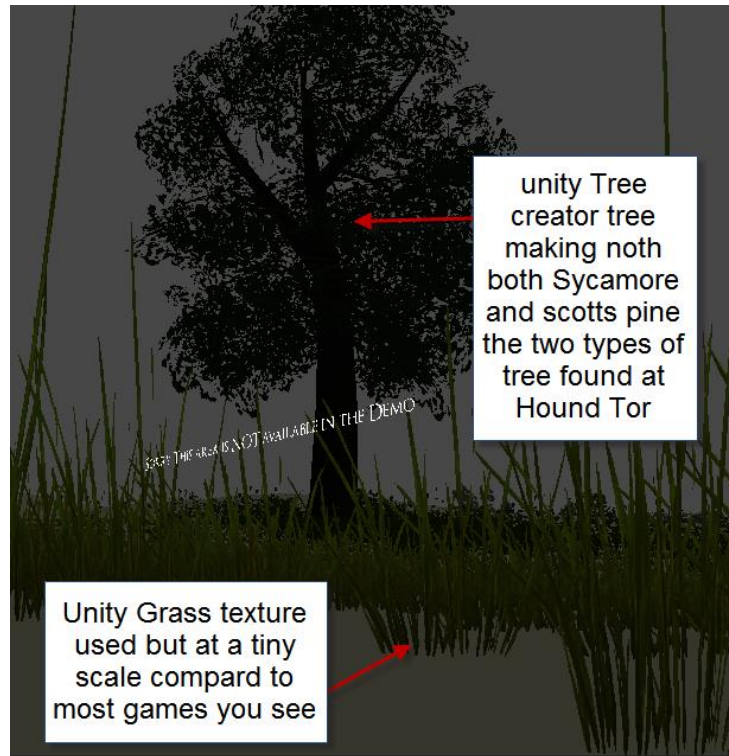
At this juncture the development was still being undertaken in the Community edition of Unity 3D version 4

Now the rocks were satisfactory it was time to add to the environment and start building them to resemble Hound Tor better.

The below screenshot aims to provide satisfactory evidence of this in action.



It is also pertinent to mention the Tree Creator tool in Unity. Inside the IDE can be created a number of tree type by simply joining nodes together in the Editor we can create trees though some practice is needed to perfect, as was found and evidenced by the attempt shown in the image below.



7.3 Development Phase 3

At this stage of the development, a lot had been created in terms of assets for the game but much was needed to be done on making the objects, now created satisfactorily, sit well and accurately representational inside the game environment.

In order to keep an equilibrium of both asset and project development with respect to coding any parts of the prototype game a decision was made to take away what had been learnt in the modelling packages and store it safely until needed so concentration could be focused into getting the game elements discussed in the level designs working.

There were three main elements to the code side that needed to be completed in order that the proof of concept could be achieved. They are laid out below, as before in previous project documents.

The experience revolves around both a story of legend and information historically accurate to Hound Tor.

The game has potential in the tourism industry as a cross interactive pre visit experience to entice people there so the game objectives had to be mandatorily casual, as in there be no time limit or emphasis on hurrying up and getting the level completed.

With the above in mind the bone model elements needed a way of being collected and stored, alongside the unveiling of a story fed achievement through collection.

The script then required an all access store that could be access system wide to effect separate components to fire some audio off when a bones is struck.

To achieve this a static variable was declared in a class named “ItemController”: it was this class that would take in updates to its itemCount which is available to the whole system being static, and based upon that count reaching a new number, effect a call to play the correct audio stored in another game object/ component in the game.

The complex part was mostly in the audio handling as each time a bones is collected the next logical audio plays.

In order to achieve this an algorithm had to be made that deciphered the current change in the array and played the audio accordingly whilst ensuring it did not jump ahead or play the same one over again.

Below is the code that was finally deemed successful.

```
void Update ()
{
    item = ItemController.itemCount;
    if (canPlay && item != prevItem)
        PlayCurrentClip();
}
```

Which calls this respectively....

```
private void PlayCurrentClip()
{
    if (item >= audioClip.Length)
        item = audioClip.Length - 1;
    else if (item < 0)
    {
        canPlay = false;
        return;
    }
    if (item == prevItem || audioClip[item] == null)
        return;
}
```

The co routine is then called to deal with starting the audio.....

```
StartCoroutine(PlaySound()
);
```

Here it checks to find out if the audio array item is the same one or that it is not empty and if it passes...

The co routine is fired and the audio plays until yielded, then the state is returned to checking for a change in the static itemCount.

This was a great success in the code and Unity handles it well. The full code can be found in the Appendices section under Appendix D

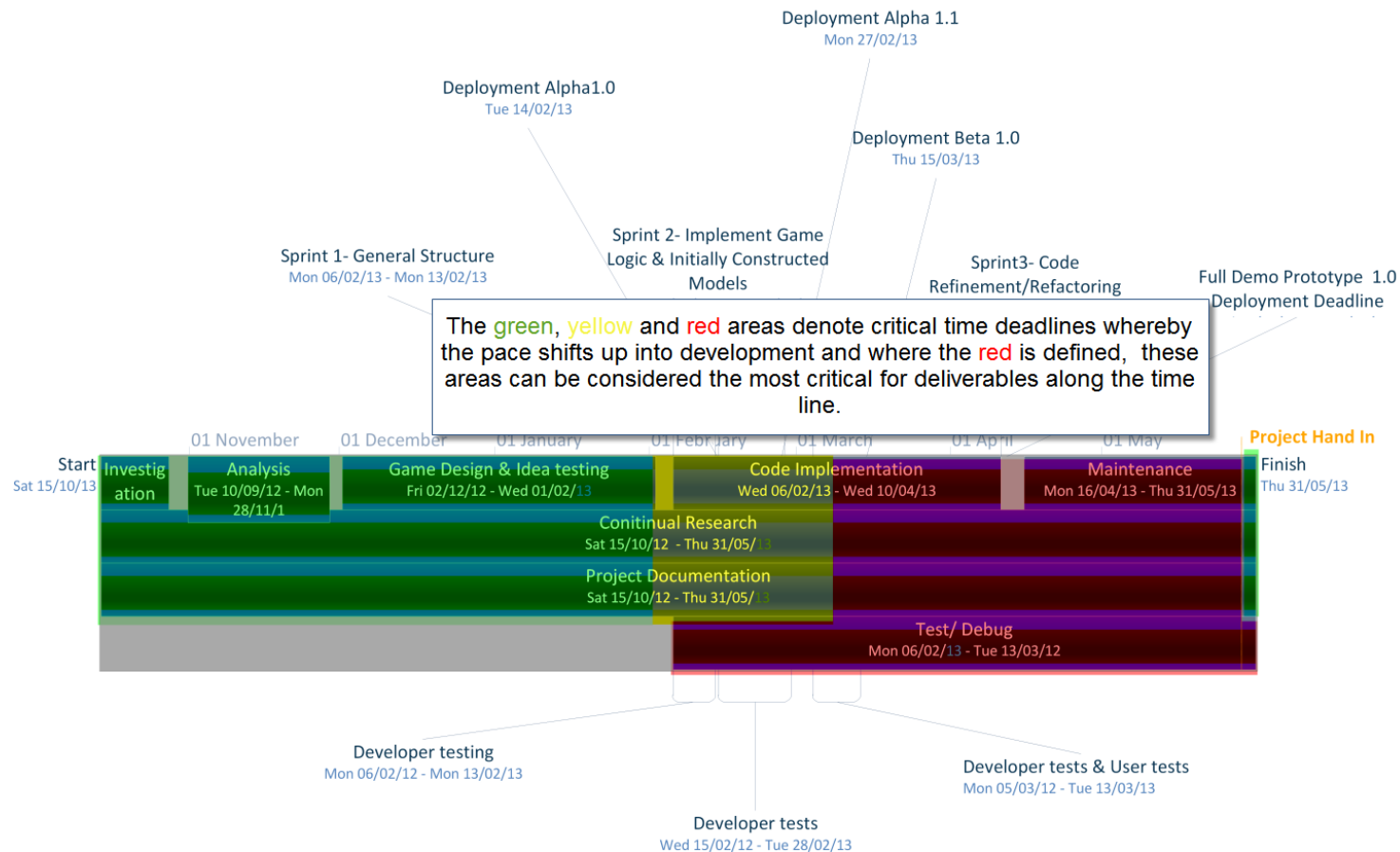
At this juncture the development was being undertaken in the educational Pro edition of Unity 3D

Further developments saw the creation of a camera flight coordinator system coded into placed to so both a relevant and somewhat

8 Project Management

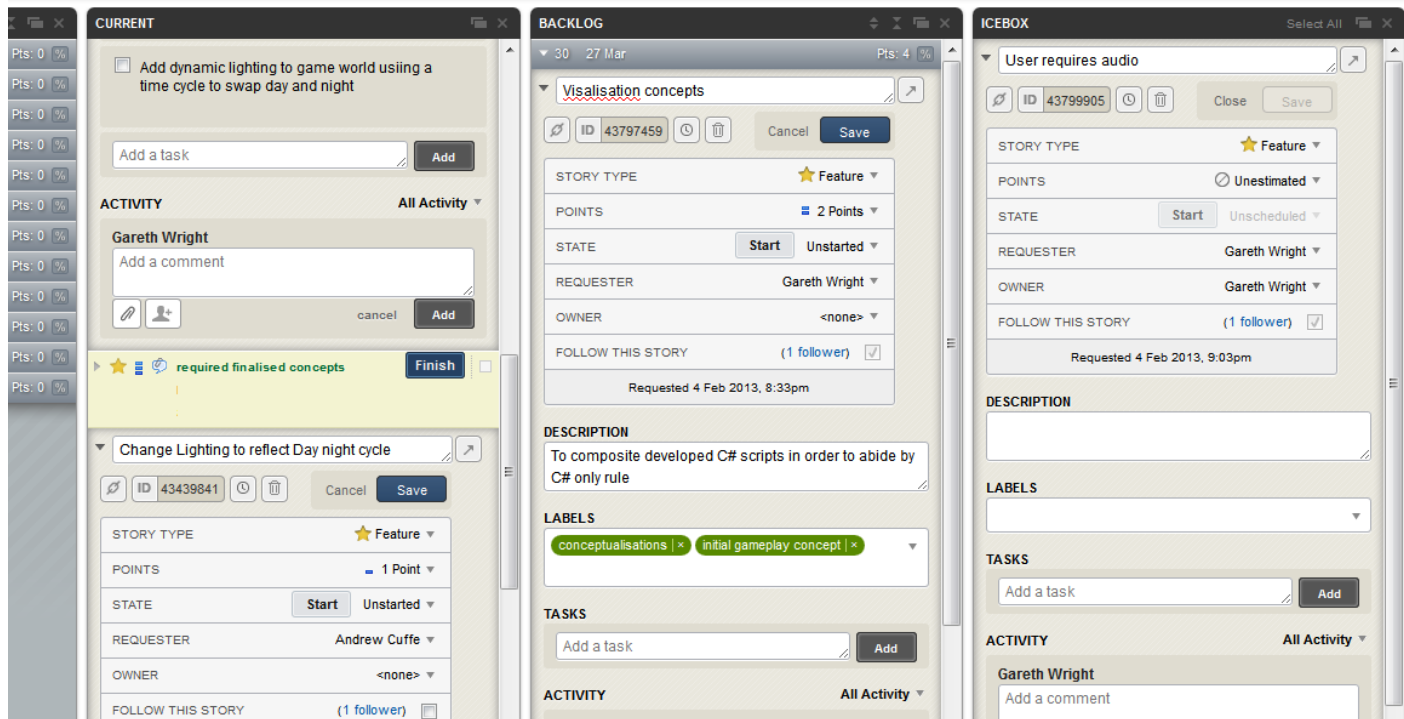
The chart below shows the timeline, which was adhered to mostly, though not without times where it felt the scheduling was a little tight at the end for a one man development and one thing to take from it would be better spread of responsibilities at the different stages. As it was a one man project, it has to be stated that times

occurred where coding drifted from one section to another that should have been a later stage implementation (flying camera system, for example). This should have been a later development but felt good to program and producing the prototype at this point for the final deliverable would not have felt or looked quite as appealing as an introductory screen (it also fed through to become part of the exit game scenario too). All this is identifying is the point that sometimes it is easy to get side-tracked under developments, especially when making some headway with the code, and can have some positive results (like the flying camera system) but they must be closely monitored so as not to go beyond specifications at iteration stages in case something is later identified that was not taken care of when it could have been so, earlier.



Below is offered a Gantt chart depicting the life cycle of the development over time.

Initially, the project utilised “**Pivotal Tracker**” an online tool that can be used to develop a project by specifying ice box requests and shifting them through to a finished plot at the end, the process fit nicely with the idea of using SCRUM tactics to deliver items to the project but unfortunately, and it is this developers own fault, the trial period was signed up for on first joining Pivotal back in February 2013.



9 Product Test

The continued testing phase results are found in the Appendices at the back of this document under Appendix..... The final Test tables are below, and show both developer & user final test results respectively

9.1 Final Developer Approval Tests

<i>Test required</i>	<i>Test deliverable</i>	<i>Passed</i>	<i>Result</i>
Game structure Functioning correctly	Ensure initial game structure class loads and game screen states are functional	PASS	<ul style="list-style-type: none"> Code produces no errors Prefabbed items are secured Casual objectives functioning well
Player Controls (Keys) Player Controls (Game Pad)	Ensure movement is closely mimicked to realistic human movement and ensure game time updates have no lag	PASS	<ul style="list-style-type: none"> Both Keyboard & Controller are functioning (GUI controls are mouse based still)
Item collections	Ensure items are collected and inventoried by player	PASS	<ul style="list-style-type: none"> All item are collected and the GUI updates accordingly
Game audio (OST)	Ensure game music working correctly and game sound effects function dynamically with collision interaction and	PASS	Game audio working correctly <ul style="list-style-type: none"> Story audio works on “itemCount” increments Ambient music was kept to a low impact volume and sounds good 3D birds and nature audio attached to tree objects which dynamically increase in volume Area visited signifier in Bones collected signifier in All GUI controls button back and forward audio signifiers in
Re-implementation of Day Night time Cycle	Reintroduced to game ?	FAIL	<ul style="list-style-type: none"> The time cycle to create day and night has been moved forward to a date beyond this first initial release, found commonly in many independent developer releases, the cycle shall return as it was not satisfactory at all in feedback tests
All models and textures correct?	See if all models are in place during play through.	CONDITION	<ul style="list-style-type: none"> All models are correct and the texturing works somewhat but it’s an area that may never get satisfied. but to ensure all models are passable and in place, they should be tested by other users for feedback, which will be addressed in the next iteration The leveraged model used to describe the snack van has only single US texturing applied and will also need

			readdressing for the next iteration
Outer-Game (compile time) Errors/Exceptions/Warnings	Ensure no errors, exceptions or warnings occur during one full play of game	PASS	<ul style="list-style-type: none"> No errors were found or created at compile time for the final iteration prototype deployment As of 18/05/2013
Inner-Game (runtime) Errors/Exceptions/Warnings	Ensure no errors, exceptions or warnings do not cause malfunction to the game experience during loading and complete play through to close	PASS	<ul style="list-style-type: none"> No errors were located in game or in any other section during runtime for the final iteration deployment As of 18/05/2013
Approve for release	Iterate and refine any issues found prior to release	PASS	<ul style="list-style-type: none"> Approved for Project Hand In As of 19/05/2013

9.1 Final User Approval Tests

Test required	Test deliverable	Passed	Result																		
Visual appeal / look of the game	Viewport look of game	PASS	Feedback from user test quotes: <ul style="list-style-type: none">• ‘The view was nice but small from me on the web version, but I could still see what’s going on’ (J.Webb, 2013)• ‘That’s great’ (Rosier, 2013)• ‘The score things at the top are bigger on the web player version, but it all looks nice’ (Wright, 2013)• ‘light shafts are nice’ (Fensom, 2013)																		
Controls	<div>(Controller/ Mouse) work as expected Both are mutually exclusive (except where stated below)and can be swapped during play if both connected</div> <table><tr><td>Control :</td><td>Type A</td><td>Type B</td></tr><tr><td>Start and GUI</td><td>Handed to Mouse</td><td>Handed to Mouse</td></tr><tr><td>Look</td><td>Mouse</td><td>Right Analogue</td></tr><tr><td>Forward</td><td>Keyboard : W</td><td>Left Analogue</td></tr><tr><td>Backward</td><td>Keyboard : S</td><td>Left Analogue</td></tr><tr><td>Left</td><td>Keyboard :</td><td>Left</td></tr></table>	Control :	Type A	Type B	Start and GUI	Handed to Mouse	Handed to Mouse	Look	Mouse	Right Analogue	Forward	Keyboard : W	Left Analogue	Backward	Keyboard : S	Left Analogue	Left	Keyboard :	Left	PASS	Feedback from user test quotes: <ul style="list-style-type: none">• ‘yep all good mate’ (J.Webb, 2013)• It’ll take me some time to get used to it but they seemed okay ’ (Wright, 2013)• ‘buttons responded well and sticks seemed like most FPS games’ (Gill, 2013)• ‘I think they were okay
Control :	Type A	Type B																			
Start and GUI	Handed to Mouse	Handed to Mouse																			
Look	Mouse	Right Analogue																			
Forward	Keyboard : W	Left Analogue																			
Backward	Keyboard : S	Left Analogue																			
Left	Keyboard :	Left																			

		A	Analogue		<p>but don't really bring anything to the experience either as mouse and keyboard or controller'</p> <p>(King, 2013)</p>
	Right	Keyboard : D	Left Analogue		
	Pause Menu	Keyboard : P	Button "Start"		
	Confirmation	Keyboard : R	Button "A"		

Navigate stages of screen to get in to game	Ensure each game screen can be navigated intuitively	CONDITION	<p>Summarised Feedback from user tests:</p> <p>All user tests confirmed navigation was simple however, it was mentioned in various feedback that the 3d Text on the start screen sometimes took a couple of clicks – this would seem not to affect the game but indicates that the area that listens for the mouse on these may need adjusting before another release</p> <ul style="list-style-type: none"> Overall the results were positive but the test cannot pass due to the inefficiency of the button handling
Game play experience Feedback for any changes to be carried out in future development and maintenance stages.	Ensure any feedback is delivered to developer	PASS	<p>Feedback from user quotes:</p> <ul style="list-style-type: none"> 'Cant wait to see the day night things you said about ' 'Will there be more stories to come? 'weather like Tomb raider ?' 'I think the narratives could do with a spruce up' <p>The above is correct, this developer's voice is not for long narratives in a game play experience</p>
GUI REWARD Text fired on meeting "Total Bones"	Did it pop up ?	PASS	

item quota			
GUI REWARD Text fired on meeting "Total Visited" area quota	Did it pop up?	PASS	
Removable from key or gamepad press	Were both popups removable?	PASS	"R" key and "A" buttons reported as working correctly.
Overall experience in general , to decide whether public testing should begin	Is this at a stage that it can be released for public review as an Alpha stage experimental prototype	PASS	This was approved by the group Who made this decision by filling out the Google form handed to them after playing
Approve for release	Feedback any further issues found prior to release	PASS	Overall, consensus was to see what the public thought outside of the closed focus group and immediate peers. This moved the decision to address any CONDITIONS arisen in the final test table to be made before the next release of the prototype

10 Final Critical Evaluation

Delivery of Working Game Mechanics

Working game mechanics was a goal of this prototype and the foundation mechanics was achieved in order to prove the concept of the virtual experience.

There is a minor issue with the Start screen 3D text mesh not immediately responding, but this was found due to the collider placed on the text mesh being perhaps a little too small, this will require adjusting slightly to ensure separation from the other menu options and should rectify the problem.

Also, the flight coordinator mechanics may be in need of some further optimization or it may be a simple increase in the time to do a full loop, accessible through the inspector of the Unity 3D Editor, meaning the problem would be rectified without a need to optimize the code at this point.

Aside from the negligible elements previously mentioned, the game mechanics are working well in that collectable items, areas, and audible story clips are played accordingly (based solely on the bone collection in this iteration)

The 2D UI and navigation is feasibly well constructed and functions excellently with the mouse. However, the XBOX Controller does not yet take over or have mutually exclusive control alongside the mouse when paused, which means players sat relaxed and enjoying the experience on a large screen may have to move back to the mouse to navigate the pause menu, possibly detracting the casual exploration value of the game play.

Delivery of Project Final Hand Prototype

Iterations of the development was released to closed testers alongside the development phase moves, though the last version had been “close to the bone” with a final better polished release appearing just days prior to the project hand in due to unforeseen circumstances.

The prototype is delivered as a part of this document online and as part of the physical hand in prerequisites.

Delivery of Gameplay & Design Documentation

All code was documented through XP in the form of code commenting and can be found in the **Appendices** section under **Appendix D**.

System and further code designs and definitions were documented using UML tools like user stories, use cases, package and sequence diagrams, and had utilised online tracker software to create evolved user stories to deliver requirements and an abstracted view of the system in general.

Conclusively, these three goals were met.

Delivery of Demonstrable product available for download over two tested platforms

Both a Web Player and Standalone versions were uploaded for users to download and test.

They both came with a specific blog that accompanies the visualisation/game.

As stated earlier in this project document, the (based on prototyping milestones) final deliverable product is available for download alongside instructional information on how to both download successfully and install it to use available at the blog page.

There is also a README file acting as the preliminary USER MANUAL in the download location plus a fairly comprehensive primer to having a go is also on the blog.

The blog address on the web is available below:

<http://hountorgame.blogspot.co.uk/2013/08/hound-tor-history-legends-virtual.html>

This goal was met. However, thoughts to take forward were based on good and bad methods of marketing the product to the public for testing. This has been a challenge, as it is very difficult to know what to say to encourage interest and indeed, where to go to do it. This would need further discussion with those in the marketing world, as this an area the project omitted to concentrate on prototype development as its main goal

Delivery of own code, where feasible, and stick to a C# only code rule

As can be found throughout this document's appendices, all aspects of the code was originally coded in, or converted to, C#

The only surviving JavaScript's left in were either purchased as a part of the small but effective particle effect, or pertained to some Image effects available to the developer as part of Unity 3D Pro tool kit. Though the deliverable, with respect to the C# only rule, was arguably not delivered by the very nature of any JavaScript's being present, it is still the opinion of this developer that the remaining scripts written in C# form the overall working body of code and ultimately alone carries the mechanics and game logic seen in the prototype. Further to this, the next iteration will have these removed and replaced by refactored C# code.

Although it appears not achieved in the real sense(due to the JavaScript presence), the paradigm has bonded to the prototype well and will continue to grow with any future developments as a symbiotic affair to the development and a good rule of thumb to stick by in general.

11 Conclusion

11.1 Conclusion

This was a hard project to deliver, mostly in due to the scale of new skillsets required to develop a whole product, even in the prototype stages such as this one. In no small part it has been a sharp learning curve and along the way has at times, been difficult to maintain the end in sight.

Mostly, the experience has taught some real lessons in delivery of product using a game engine like Unity.

The caveats that come with Unity3D, alongside the advantages too (namely the speedy prototyping game functionality and quick tangible object positioning in world space) were often a greatly received simplification, but also had caveats like the dreaded prefab. This tool initially appears as an advantage tool but soon becomes dangerous, if the prefab is not well designed before its application into a game scene. Changing these once created is not recommended, especially if they house a large set of components perhaps prefabbed themselves.

Though the above should be noted, it has been the discovery and enjoyment of getting to know Unity 3D as game development IDE which has been a positive influence in achieving the final goal.

This further concretes a passion unaffected by the woes of constant compiler fails, or uncaught runtime errors, the inconsistent tutorials that offer to provide an understanding to a solution or even the solution itself and turn out to be FOS. Most appreciated was the excellent online community at the Unity Answers website whereby many evangelistic members are happy to hold a hand from time to time and help you see find the answer (an important aspect not to be confused with just being given the answer). Needless to say, They are inspired and altruistic people that can only be recommended to anyone attempting the Unity 3D development product. This is why they have an acknowledgment in the beginning of this document alongside the other there.

One real advantage gaining experience in modelling software, something that had been avoided up to this point due to concentrated efforts in learning to develop code. This is something that will have continued development, not in any small part due to iterating the prototype further forward but also as a beginner skillset to hone and keep in the tool belt. Also, though it was difficult to find the correct modelling package to get to into, this inadvertently caused, at a beginner level of course, experience to be attained in Blender, Max, Maya, MudBox, ZBrush,

Height mapping techniques learnt, or conjured, and used were also found to be a valuable aspect in this developer's tool kit. It will be further utilised for potential future recommendation found below.

Documenting the system was at a level satisfactory to a one-man development but, it has always been the downfall of this developer and one area strived toward understanding better all the time. The methods used in this project documentation is proof of improvement to this developer but, as may be the developers curse, can sometimes fall by the wayside to make way for more fruitful accomplishments like the reward of working code, for example.

It is always in understanding that a system absolutely must have clear portrayal to the non-technical minded that always have roles, and often better heads, for the other equally important aspects in any development.

Finally, Coupled to the project, is the inspiring thought that the attainment of time served experience in the Unity 3D environment, where the problem with actually turning code into something that closely resembled an original idea, is no longer such a daunting prospect.

This conclusion ends overleaf with a table showing the skills discovered or developed resultant of this projects undertaking.

Skills learnt or enhanced:	
Unity 3D-significant programming and environment usage skills	Project Documentation Practices and techniques
General 3D tactics and game logic knowledge development	Height Mapping (creation to use)
Multiple Modelling software experience and the modelling simple items	Audio knowledge for games creation
Enhanced appreciation for all aspects of creative development outside the programming	Photography knowledge
General C# knowledge and skills development	Knowledge and understanding of COP (component oriented programming)

11.2 Recommendations for the future development of the project

In the future and in order to get one`s exact way in unity 3D, it would be better to build some extension methods that frequently seem to pop up during code implementations.

By drilling in some extension methods unity could be greatly optimized and serve one`s future needs more fluidly (something this developer is building as you read these final words). Further advantages lie “under the hood” that perhaps suitable API`s can be constructed to better utilise the engine. However, this is not to say, the Engine was a lacking, it was a great tool as it stood but extension classes and methods would improve workflow for a start.

The Prototype will continue development as stated throughout this project. It is the opinion that it is closer to a beginning of something rather than a means to an end.

Much of the code will undoubtedly need refactoring upon review during the next iteration phase to consolidate certain classes that might be found too similar (i.e. CollectItems.cs/ CollectAreas.cs are an example of this) and there is the conversion previously mentioned concerning both image and particle effects.

Not least, the audio is in dire need of a clean and concise narrator. This would inspire better story telling that would only serve to enhance the overall experience of the prototype.

Much of the project ended up focusing on a visualisation / experience driven game play and it predominantly stayed the main focus. Going back to visualisation it can only be advantageous to the prototype to develop this aspect as much as possible but it was advantageous to discover clearly that a small scale model environment is difficult to get real data for, denoting either an up-skill direction in environment modelling by hand or an up-scale by looking to have larger environments

Thus.

A wider scale visualisation of another real location is under research with potential for a development, if feasible.

Perhaps, an entire of Dartmoor could become the grounds for the next setting with more of an adventure pathway, though still in the realms of location visualisation alongside maintaining the informative but fact driven aspects.

13 Bibliography

Adiscon GmbH, (2001) *Why Unicode?*. [Online]

Available at: <http://www.eventreporter.com/common/en/articles/why-unicode.php>

[Accessed 28 March 2012].

Ambler, S. W., n.d. *Introduction to diagrams of UML 2.0*. [Online]

Available at: <http://www.agilemodeling.com/essays/umlDiagrams.htm>

[Accessed 01 12 2011].

Cay S. Horstmann, A. d. P., (n.d.) Untitled <http://alexdp.free.fr/violetumleditor/page.php>. [Online]

Available at: <http://violet.sourceforge.net/>

[Accessed 12 November 2011].

Cooper, J. W., (2003) *C# Design Patterns A Tutorial*. Boston: Pearson Education, Inc.

cplusplus.com, (2012) *Namespaces*. [Online]

Available at: <http://www.cplusplus.com/doc/tutorial/namespaces/>

[Accessed 7 April 2012].

Dean Leffingwell, D. W., 2010. *Agile Software Requirements*. 1st ed. Westford: Addison Wesley.

etayrien, (2006) *Basic Game Engine Structure*. [Online]

Available at: <http://blogs.msdn.com/b/etayrien/archive/2006/12/12/game-engine-structure.aspx>

[Accessed 20 April 2012].

Fensom, D., (2013) *Quoted Feedback*. Exeter: None.

Gamma. E, H. R. J. R. V. J., 1995. *Design Patterns: elements of reusable object-oriented software*.

Reading: Addison-Wesley.

Gill, A., (2013) *Quoted Feedback*. Newton Abbot: None.

[http://www.cs.millersville.edu/~webster/gametechnologytrack/CS475/XNA/misc/CBennettXNA2D3](http://www.cs.millersville.edu/~webster/gametechnologytrack/CS475/XNA/misc/CBennettXNA2D31.pdf)

1.pdf, (2009) *A Simple Introduction to Game Programming With C# and XNA 3.1*. [Online]

Available at:

<http://www.cs.millersville.edu/~webster/gametechnologytrack/CS475/XNA/misc/CBennettXNA2D31.pdf>

[Accessed 5 April 2012].

IBM, (2005) *Explicit initialization with constructors (C++ only)*. [Online]

Available at:

<http://publib.boulder.ibm.com/infocenter/lnxpcomp/v8v101/index.jsp?topic=%2Fcom.ibm.xlcpp8l.doc%2Flanguage%2Fref%2Fcplr387.htm>

[Accessed 14 April 2012].

IBM, (2005) *Initialization of structures and unions*. [Online]
 Available at:
<http://publib.boulder.ibm.com/infocenter/lnxpcmp/v8v101/index.jsp?topic=%2Fcom.ibm.xlcpp8l.doc%2Flanguage%2Fref%2Fstrin.htm>
 [Accessed 14 April 2012].

Interactive Systems Studio, (2013) *GALLERY*. [Online]
 Available at: <http://www.interactivesystemsstudio.com/gallery.html>
 [Accessed 06 August 2013].

J.Webb, (2013) *quoted for feedback*. Newton Abbot: None.

King, M., (2013) *Quoted Feedback*. Newton Abbot: None.

Liberty, J., (2001) *C++ -> C#: What You Need to Know to Move from C++ to C#*. [Online]
 Available at: <http://msdn.microsoft.com/en-us/magazine/cc301520.aspx>
 [Accessed 1 April 2012].

McMullan, A., (2009) *C# FAQ for C++ programmers*. [Online]
 Available at: <http://www.andymcm.com/csharpfaq.htm>
 [Accessed 9 February 2012].

McShaffry, M., (2003) *Game Coding Complete*. Scottsdale: Paraglyph Press, Inc..

Micheal Jesse Chonoles, J. A. S., (2003) *UML 2 FOR DUMMIES*. New York: Wiley Publishing. Inc.

Microsoft, (2012) *.NET Framework 4*. [Online]
 Available at: <http://msdn.microsoft.com/en-us/library/w0x726c2.aspx>
 [Accessed 15 April 2012].

Microsoft, (2012) *Built-In Types Table (C# Reference)*. [Online]
 Available at: msdn.microsoft.com/en-us/library/ya5y69ds.aspx
 [Accessed 13 April 2012].

Microsoft, (2012) *C# Language Specification*. [Online]
 Available at: <http://msdn.microsoft.com/en-us/library/ms228593.aspx>
 [Accessed 26 April 2012].

Microsoft, (2012) *Classes and Structs (Managed)*. [Online]
 Available at: <http://msdn.microsoft.com/en-us/library/6w96b5h7.aspx>
 [Accessed 6 April 2012].

Microsoft, (2012) *Game.Components Property*. [Online]
 Available at: <http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.game.components.aspx>
 [Accessed 22 April 2012].

Microsoft, (2012) *Game.Draw Method*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.game.draw.aspx>
[Accessed 22 April 2012].

Microsoft, (2012) *GameComponentCollection Class*. [Online]
Available at: msdn.microsoft.com/en-us/library/microsoft.xna.framework.gamecomponentcollection.aspx
[Accessed 22 April 2012].

Microsoft, (2012) *GameWindow.ClientBounds Property*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.gamewindow.clientbounds.aspx>
[Accessed 03 March 2012].

Microsoft, (2012) *General Structure of a C# Program (C# Programming Guide)*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/w2a9a9s3.aspx>
[Accessed 1 April 2012].

Microsoft, (2012) *Introduction to the C# Language and the .NET Framework*. [Online]
Available at: <http://msdn.microsoft.com/library/z1zx9t92>
[Accessed 01 April 2012].

Microsoft, (2012) *long (C# Reference)*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/ctetwysk>
[Accessed 1 March 2012].

Microsoft, (2012) *Microsoft.Xna.Framework Namespace*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.aspx>
[Accessed 12 April 2012].

Microsoft, (2012) *Playing a Video*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/dd904198.aspx>
[Accessed 23 April 2012].

Microsoft, (2012) *Structs Tutorial*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/aa288471%28v=vs.71%29.aspx>
[Accessed 2 April 2012].

Microsoft, (2012) *using Directive (C# Reference)*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/sf0df423%28v=vs.90%29.aspx>
[Accessed 02 April 2012].

Microsoft, (2012) *Video Class*. [Online]
Available at: <http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.media.video.aspx>
[Accessed 8 April 2012].

Microsoft, (2012) *VideoPlayer Members*. [Online]
Available at: <http://msdn.microsoft.com/en->

[us/library/microsoft.xna.framework.media.videoplayer_members.aspx](http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.media.videoplayer_members.aspx)

[Accessed 5 April 2012].

Microsoft, (2012) *Windows Phone 7: "How Do I?" Videos*. [Online]

Available at: <http://msdn.microsoft.com/en-us/gg243438.aspx>

[Accessed 15 April 2012].

Microsoft, n.d. [Online].

Mono, no date. *What is Mono*. [Online]

Available at: http://www.mono-project.com/What_is_Mono

[Accessed 06 10 2012].

Nystrom, R., (2011) *Game Programming Patterns / Behaving Patterns*. [Online]

Available at: <http://gameprogrammingpatterns.com/component.html>

[Accessed 2 April 2012].

Rajesh, V., (2001) *Understanding Structures in C#*. [Online]

Available at: [http://www.c-](http://www.c-sharpcorner.com/uploadfile/rajeshvs/structuresincs11112005234341pm/structuresincs.aspx)

[sharpcorner.com/uploadfile/rajeshvs/structuresincs11112005234341pm/structuresincs.aspx](http://www.c-sharpcorner.com/uploadfile/rajeshvs/structuresincs11112005234341pm/structuresincs.aspx)

[Accessed 1 April 2012].

Reed, A., (2011) *Learning XNA 4.0*. Sebastopol: O'Reilly Media Inc.

Rosier, O., (2013) *Quoted feedback*. Newton Abbot: None.

Schuller, D., (2011) *C# Game Programming For Serious Game Creation*. Boston: Course Technology.

Scrum Alliance, n.d. *Scrum Principles*. [Online]

Available at: http://www.scrumalliance.org/pages/scrum_101

[Accessed 30 April 2012].

Sharp, J., (2010) *Microsoft Visual C# 2010- Step by Step*. Washington: Microsoft Press.

silviu11, (2011) *Creating your first Skeleton in 3DS Max*. [Online]

Available at: <http://3dm.com/3d-model/skeleton-tutorial.html>

[Accessed 05 April 2013].

Unity Technologies, (2013) *mecanim*. [Online]

Available at: <http://unity3d.com/unity/mecanim/>

[Accessed 15 January 2013].

Unity Technologies, (2013) *SerializeField*. [Online]

Available at: <http://docs.unity3d.com/Documentation/ScriptReference/SerializeField.html>

[Accessed 06 06 2013].

Walsh, J., (2011) *XNA 4.0 Workshop*. [Online]

Available at: <http://gamedevelopedia.com/post/2011/08/14/XNA-40-Workshop-Week-5.aspx>

[Accessed 1 April 2012].

Wells, D., (2009) *Extreme Programming*:. [Online]
Available at: <http://www.extremeprogramming.org/>
[Accessed 15 April 2012].

Wright, N., (2013) *Quoted feedback*. Newton Abbot: None.

Thankyou for reading.