Stargate: Galaxy

Project Summary Report

Peter Mitchell
Andrew Krix
Karlos White
Phil Lavender
Kane Stone

11/11/2011

Table of Contents

1 Tables and Figures

1.1 Tables

1.2 Figures

2 Introduction

This document is intended to provide an overview of the project Stargate: Galaxy. It has been a large project and there has been a lot of documentation and a large system developed so this report will summarise the important points. Further information about the game and the development may be found in the other documentation as will be indicated at points in this document.

Stargate: Galaxy is a game that allows explorative freedom in an enormous galaxy set in the universe of the TV franchise Stargate. The game provides the player with a number of ships they may obtain, control, level up, and equip more weapons to. They must engage in combat with four different factions that appear within the game as seen in the TV series: the Wraith, the Goa'uld, the Ori, and the Replicators. Each race has their own large ships that they will try their best to destroy the player with. To facilitate this combat the player is given control over the amount of power they wish to allocate to their shield, weapons, and engines to give them control to play as they want. By destroying enemy ships they will level up and be able to use resources from wrecks to eventually purchase additional weapons. The eventual goal of the game is to free the galaxy with the assistance of the Asgard who will help the player in the game. There are more than 130 worlds that the player must capture, but there are bosses for each faction that will attempt to prevent the player from this along with all of the other ships too. It is hoped that players enjoy the freedom given to them to play how they want to.

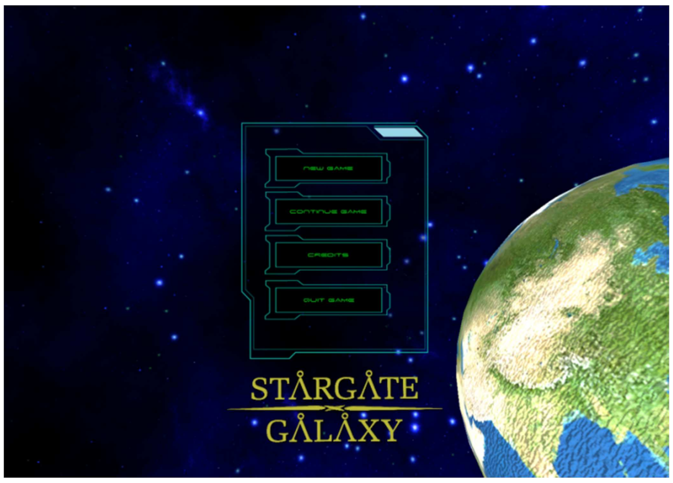


Figure 1 : The main menu

As part of this document the project shall be overviewed, some of the large amount of research that was done summarised, a description of the process of development of the product, and a section about the project management of the project. The appendices include the meeting minutes, agendas and other information has been included on the provided disc. They have not been included directly with this document to save printing many hundreds of pages off. The provided disc also includes a copy of all the documentation completed, the project source files, and the project executable. It is hoped that the product that has been developed in enjoyable to play.

3 Project Overview

3.1 Project Title

Stargate: Galaxy

3.2 Project Purpose

The purpose of the project was to develop a working software system with the associated development lifecycle in a team environment. The purpose for choosing this particular project was that there was a general interest in the Stargate franchise within the group and a discovered common interest in developing a game based on the show. The target audience of the game is fans of Stargate, but it has been designed so anyone who likes flying a spaceship around destroying ships should appreciate the game. More than anything else though it was designed to be a project that would provide a great learning experience for those involved as it required a lot of research into areas that team members had not worked in before. Then the researched material has to be applied to produce a working game and the associated materials.

3.3 Project Description

The idea behind the project was to create and document the creation of a game created on the Unity game development platform. The game, Stargate: Galaxy, has been designed as a free roam space combat style game in the third person view. Main ideas behind the game play were taken from experiences from other similar game styles. One game in particular that had a lot of influence mainly due to its own success was Eve Online by Crowd Control Productions. The lore or story behind the game was loosely taken from the successful Stargate franchise. The story line, or lack thereof, does not specifically follow the franchise but the ships and races along with the general specification of each were. The game itself has a variety of specific features that were decided upon while initially planning the project.

The game was designed to be a free roam galaxy so to say, where the player is able to go anywhere they want with limited restrictions. Mainly the idea was that there are no scripted events or forced paths in the game. The player is able to choose how they want to play and where their story begins in the vast and wonderful universe that is Stargate: Galaxy. Additionally the player has full control over the control of their ship. Players control the power distribution of their ship between shields, weapons and engines all of which offer different awards for increasing the power consumption of each. There is also full control over which weapons are firing when and especially what the weapons are firing at. Such micro-management is a major feature of the game as it forces the player to pay attention to the smaller details of playing the game.

Battles that take place throughout Stargate: Galaxy are huge and ever continuing. Even if the player has no role in the engagement the battle continues causing the galaxy to be ever changing. The game world consists of multiple star systems each of which may or may not be controlled by another race. While there are star systems that are neutral to begin with the artificial intelligence of the non-player characters enables each race to move and take neighbouring star systems. It is also possible that while a race still exists somewhere in the game world, they are able to repopulate and continue expanding. This design means that no two galaxies will ever be the same even when left completely alone. The player can enter and exit each of these star systems as well as battles by use of a hyperspace travelling feature as well as normal movement.

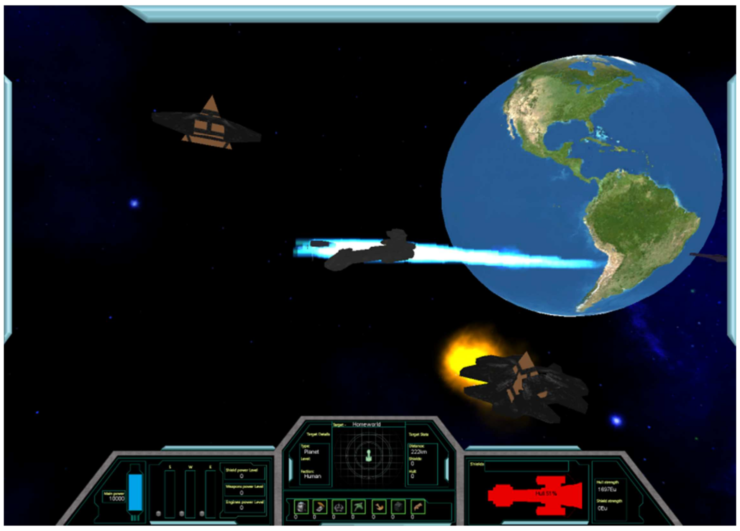


Figure 2 : A battle occuring between Asgard, the Goa'uld, and the player

After a battle in which the player is involved, the player is given experience which is used to increase the level of the player's ship as well as the opportunity to gain loot from destroyed enemies. The increase in ship level improves the performance of the ship enabling it to take one stronger enemies and gain more loot. Loot itself is put towards the trading system implemented which allows the player to choose specific improvements though a storefront type trade system. The trade system was designed in such a way that the player could not purchase upgrades without having to first earn the right to have them. This is done according to the price of each upgrade as well as the specific requirements of each item wise. Though the system it is possible to purchase both new weapons for the current player's ship as well as new ships entirely.

These main points describe the core game play behind Stargate: Galaxy and define the type of game that it is. Of course there are other features that make Stargate: Galaxy the game that it is and they will be described in the following report.

4 Summary of Research

The following sections detail briefly an outline of some of the research that was undertaken while developing Stargate: Galaxy and completing the project. There was a lot of research completed in the areas of:

- Stargate and how it could be used for this game
- How the assets could be developed to represent those seen in the shows without too much overloading of work
- Using Unity as a development platform
- The developing of interfaces in Unity
- Implementing particular game play elements in Unity

4.1 Existing knowledge used in this project

Entering into this project there were a number of areas of prior experience that have been used to our advantage; in particular the completion of the topic software engineering 1 and software engineering 3. Peter, Kane, and Phil had all completed both of these topics. And Karlos had completed software engineering 1 and was completing software engineering 2 this semester. Much of the basis for the documentation that was completed was based on the processes learned during those topics.

The project management topic was completed by Karlos, Phil, Kane, and Peter in the break just before the commencement of this topic where we worked as a team there too. So that is where are lot of the project management approaches that were applied to this project particularly in terms of the approach to documentation of the planning came from.

Andy of course being a digital media student was well practiced at developing various art assets that were used on the project. And there were a lot of general programming knowledge abilities across the team from the last few years in particular.

4.2 Stargate and Eve: The two influences supporting the story and gameplay design

Stargate

There were a number of influences supporting the story and gameplay design for Stargate: Galaxy. Choosing an already established franchise such had a large impact on the story design for the game, it allowed for the inclusion of the races which the franchise had established the background for and gave the development team a framework to which elements could be added. The established background influenced the story for the game because in the Stargate television series events such as the defeat of the Ori have already occurred, this means that during the creation of the story the development team had to create an explanation as to why all of the races were in one galaxy and in conflict with each other. Due to the use of “alternate realities” in the Stargate Series such as the alternate Earth in the Stargate SG1 episode “There But For The Grace Of God”, the concept of alternate realities allowed the development team to create a game universe in which all of the adversaries featured in the Stargate franchise were still actively working against Earth. In the alternate reality of Stargate: Galaxy the Goa’uld and the Ori are still actively fighting Earth, the replicators have not been destroyed and the Wraith were able to make their way to the Milky Way from Pegasus.

The use of the Stargate franchise as a setting also gave the game a direction as one of the shows major themes is exploration, this influenced the story and gameplay aspects. In terms of the gameplay direction this influenced the initial state of the galaxy with most of the systems having unknown owners, the player discovers where the enemy forces are and which planets are safe by exploring the galaxy, as the player explores the galaxy they open up new areas to travel which reinforces the theme of exploration. Another of the influences that Stargate had upon the gameplay design was in the creation of the player’s ship and the enemy ships that the player would encounter throughout the game, the players ship would have access to all of the weapons that the human forces would have access to in the Stargate series. Enemy ships would have different statistics based upon their abilities in the TV series, for example the Wraith have no shield technology. The differences between the ships would also be used to define the characteristics and abilities in the game.

Another of the influences that the Stargate franchise had on both the story and gameplay design was the space battles. In the Stargate franchise large space battles such as in the episodes “Camelot”[3] and “Be All My Sins Remember’d” have significant impacts on the story and provide amazing visual set pieces. These battles provided the inspiration for the combat sections of Stargate: Galaxy, the development team wanted to recreate the feeling of the battles as partially chaotic with the amount of fire being exchanged by the vessels and also have them be fun and engaging for the viewer. Using the various space battles that occur in the Stargate franchise as references the development team looked at the designs of the ships and made decisions which related to the difficulty that the ships would pose to the player, for example in the episode “Camelot” the Ori ships easily defeat the fleet that is arrayed against them, using this as a reference the team designed the difficulty of the game so that the races encountered would have various strengths and weaknesses based upon the abilities seen in the show and that certain races such as the Ori would be far more difficult to defeat than the Goa'uld.

The development team used the designs of the ships that exist within the Stargate franchise as the basis for the ships within the game. For example the players first ship is Prometheus which was also the first battlecruiser built by the humans in the series, the design of the enemy ships was also used as the basis for the design of their ships within the game, the development team also designed the enemy ships to have the same weapons that they did in the show, for example the Wraith Hive Ship is armed with energy weapons while the human ships begin with access to railguns and rockets. Information about the size of the vessels was used to scale the in game models so that they were appropriately sized. For example the Daedalus class cruiser is estimated to be around 200m long (Stargate Wiki, 2005) whilst the Goa'uld Ha'tak is estimated to have a length of nearly 700m (Stargate Wiki, 2005). These values were used to scale the models so that they were of an appropriate size when scaled to one another; however, the planets were significantly smaller than they would appear in the show in order to avoid making the player feel too small in regards to the universe around them.

Eve

Eve Online by Icelandic video game developer Crowd Control Productions (CCP) is an open-ended Massively Multiplayer Online Game (MMOG). The entire game world is set in space between some 7,500 different star systems where anything can happen and every decision changes the game. This style of space exploration and combat where the player is given the ability to explore entire systems and engage in massive battles against unparallel odds influenced Stargate Galaxy's game play. How the player interacts with the game world was also based off Eve Online. The controls in Eve Online are traditionally bound to both the mouse and or the keyboard. Everything can be done using the mouse including movement and action controls, however in Eve Online, it is possible to interchange some controls and use keyboard shortcuts instead. The movement controls revolve around the use of the right mouse button to select target locations to either move to or interact with if some game object exits at said location. This method of movement and target selection was simplified and ported into Stargate Galaxy where a player used the mouse to interact with the game world and the keyboard to utilize the graphical user interface provided. This style was decided upon as movement and especially target selection in a three dimensional space is not practical with just the keyboard and so a first person shooter style would over complicated the game play and detract from the micro-management of the ship.

The graphical user interface of Eve Online also provided Stargate Galaxy with some necessary feature design ideas to make the game play possible. Two of the main graphical user interface sections that were used as a template design were the Ship Control Panel and the Overview Scanner. The Ship Control Panel in Eve Online makes available information such as the status of the shield, armour and capacitor as well as actions such as enable weapons, autopilot and speed control. Similar was used in Stargate Galaxy but the weapons were separated into another section to reduce clutter and due to the extra controls needed. The Overview Scanner also gave birth to another in Stargate Galaxy however this was also separated with some of the details such as the selected ship being moved to the Ship Control Panel. In Stargate Galaxy the Overview Scanner section only shows the game objects in the immediate vicinity of the player and their distance. Eve Online has also deeper roots leading into features such as piracy, diplomacy and industry but these did not influence Stargate Galaxy as it would have detracted from the Stargate aspect of the game.

4.3 Research and design of the model based art assets

To design the various models needed in this project, each ship would need to be roughly the same size, shape and the texture as they appear in the Stargate TV series. To do this, all the information used will be in some way sourced from the Stargate TV series, either if it's directly given by the show or calculated from a video frame that has been taken out of an episode. To help make producing the ships easier, the research will be done in step. Each step is focusing primarily on one area of detail at a time as to make it easier to find what is desired.

The first step in researching the ships was finding out the various dimensions and features of each ship. There are a number of Stargate wiki websites that were quite helpful for this as they had already found the main features and dimensions of the majority of the ships. Although the different wikis had the majority of features for the ships in Stargate, they did not always have the all the details needed, as some ships have been listed without any size measurements. To get the remaining details needed, we used forums and discussions boards to find discussions between Stargate fans about the dimensions of the remaining ships. Using the discussion boards did have its problems though; there was quite often conflicting sizes with people giving their own estimated size. These estimated sizes were often produced by comparing the ship in question with another ship that's in the same frame which could also cause problems if the size of the comparison ship was also wrong or if one ship was closer than the other.

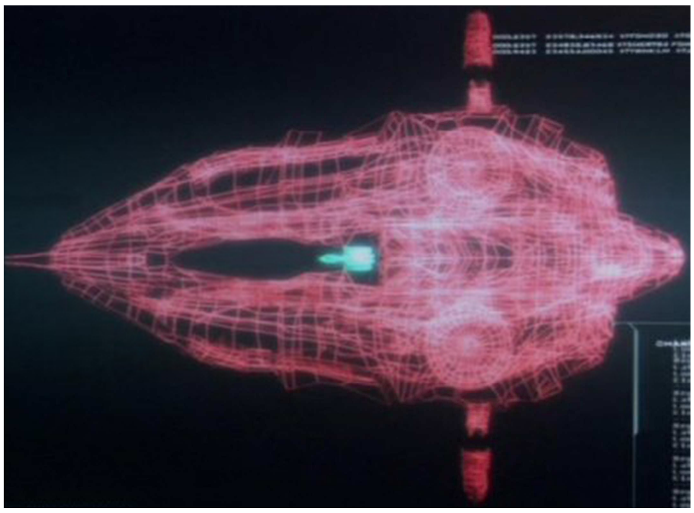


Figure 3: Daedalus compared to the Wraith Hive ship

This technique also had an advantage, during the research I found an image from the show had a direct comparison of the Daedalus, shown in blue, and the Wraith Hive ship, in red. Since during research for the Daedalus, it's known to be about 225m long and from that it's possible to estimate how big to make the Wraith Hive ship without knowing its actual size. Using this image as a basis, we can measure roughly how many pixels the Daedalus is compared to the size of the Wraith Hive ship.

The second step in researching the ships was to find images that clearly displayed the ship and the various components of it. This was done over a series of searches that each was primarily looking for different details of the desired ship. The first search was looking for images or concept sketches that gave a good, clear view of the ships general shape. The second search focused on finding clear images that would provide detail on any of the major features on the ship. Finally, the third search was to find ships with an even amount of lighting across its hull as to help provide an understanding of what the final texture will need to look like. The Asgard Mothership and the Wraith Hive ship had the most problems at this stage because it was extremely difficult to find images that showed the details clearly and did not have object obstructing the view of the ship. Unclear images were a general problem that was caused by images taken from the TV series because the show wasn't recorded in high definition. This was overcome by spending more time researching and to have more images that covered sections of the ship instead of broad views.

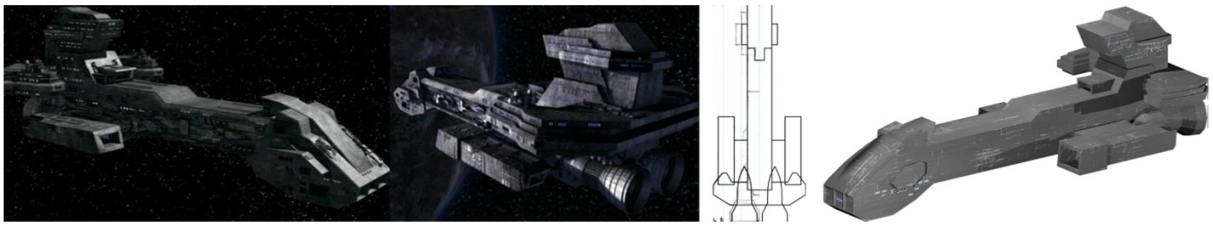


Figure 4: Stages of design for the Prometheus

This illustrates the stages of design needed to help produce the models. The two images on the left are images of the Prometheus taken from the Stargate show as found during the research. There are lots of different images used when sketching the Prometheus and each are primarily showing different angles as to get a better understanding of how the ship has been designed. Then the sketch in the middle has the general shape of the model and any major features that are to be included. The sketch is to make sure all the proportions are right and that everything known has been researched before attempting to model it. Later on, these reference images would also be used as a good basis to develop the general look of the texture. The last image on the right shows the Prometheus after being completely modelled and textured.

Occasionally, there would need to be a third step in the research that would instead focus on video clips instead of still images. While video clips will be harder to get any fine detail that is needed for sketching, it will provide more detail about depth and whether or not there are any animated parts, these details would not be possible using images. As this step was done on a couple of ships and after the majority of sketching, there weren't any major problems. The most noticeable ship to need this step was the Replicator Cruiser which has a centre piece that slowly spins; relying purely on images for the details would have never shown this small detail.

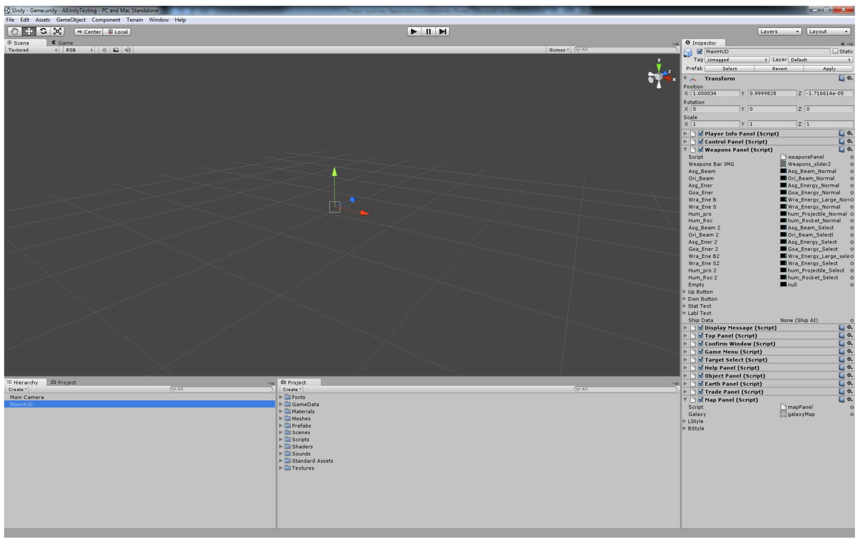
The final step was to continue researching the ship's hull in order to make the textures for the ships that have been modelled. These textures were needed to provide any fine details that were I thought will be noticeable from within the game. In addition to researching the ship's hull, any existing free to use textures and tutorials for creating spaceship hull textures were also researched. While there were plenty of images of the ships, there wasn't any useful information on the internet about creating our own spaceship textures for use in a game. Because of this and no useful pre-existing textures could be found, the creation of our own textures relied on the images from the show to provide us the details to make ourselves.

By using these steps to research the Stargate ships, we found all the details we needed to reproduce the ships ourselves for use in our game. However, using a TV series as a basis for sketching does have its problems because the show was broadcasted in standard definition, it's very difficult to find good sized images that are still clear. Getting as much information and images on the ships as possible before starting the actual modelling will help streamline the modelling and texturing process as we never needed to go back and find more on the ships.

4.4 Using Unity as a development platform

Unity 3D was used as the primary development environment and linked with Microsoft Visual Studio as an integrated scripting platform. Unity 3D provides developers with an enormous number of different features that allow the development of 2D and 3D games although it is mainly intended for 3D. The purpose of this section is to introduce Unity and explain some of the high level concepts that were researched and used in developing the overall product.

The Unity website does not provide a very large collection of free tutorials. The tutorials that they provide on their website relate more to a specific few types of concepts rather than an introduction to the system as a whole. In searching for a means to learn Unity it was discovered that 3D Buzz, a video tutorial training site, provided a massive number of tutorials on many of the introductory topics. These videos were found to be invaluable as they cover from a ground up approach introducing all of the concepts and then provide examples of how each element worked. Initially looking at Unity the interface has many different regions and operations and this site provided an overview that allowed understanding of these features without complication.



MainHUD
Static
Tag: Untagged
Layer: Default
Prefab: Select | Revert | Apply

Transform
Position: X 1.000034 Y 0.9999828 Z -1.716614e-05
Rotation: X 0 Y 0 Z 0
Scale: X 1 Y 1 Z 1

Player Info Panel (Script)
Control Panel (Script)
Weapons Panel (Script)

Script	weaponsPanel
Weapons Bar IMG	Weapons_slider2
Asg_Beam	Asg_Beam_Normal
Ori_Beam	Ori_Beam_Normal
Asg_Ener	Asg_Energy_Normal
Goa_Ener	Goa_Energy_Normal
Wra_Ene B	Wra_Energy_Large_Norı
Wra_Ene S	Wra_Energy_Normal
Hum_pro	hum_Projectile_Normal
Hum_Roc	hum_Rocket_Normal
Asg_Beam 2	Asg_Beam_Select
Ori_Beam 2	Ori_Beam_Selectl
Asg_Ener 2	Asg_Energy_Select
Goa_Ener 2	Goa_Energy_Select
Wra_Ene B2	Wra_Energy_Large_sele
Wra_Ene S2	Wra_Energy_Select
Hum_pro 2	hum_Projectile_Select
Hum_Roc 2	hum_Rocket_Select
Empty	null
Up Button	
Dwn Button	
Stat Text	
Labl Text	
Ship Data	None (Ship AI)

Display Message (Script)
Top Panel (Script)
Confirm Window (Script)
Game Menu (Script)
Target Select (Script)
Help Panel (Script)
Object Panel (Script)
Earth Panel (Script)
Trade Panel (Script)
Map Panel (Script)

Script	mapPanel
Galaxy	galaxyMap
LStyle	
BStyle	

Main Camera
MainHUD

Fonts
GameData
Materials
Meshes
Prefabs
Scenes
Scripts
Shaders
Sounds
Standard Assets
Textures

Figure 5 : The Unity Interface

The Figure seen above shows the main interface that is presented by Unity 3D and where most of the work is done apart from scripting. The main view area seen in the centre can be changed between a free control view of the scene as it is in the above, and an active game view. This allows viewing of the game as it appears from the camera while still being able to go and check elements are working properly from alternate angles. The lists in boxes seen on the bottom show by default all of the elements that are in the current scene and a complete collection of the project assets in the other. The project assets can be organised into collections of folders to provide an easy way to locate assets when needed. The top bar provides the main menus for inserting new objects and adding additional components to assets and manipulating the positions within the scene. The other panel on the right is one of the most important features; it is the inspector panel and provides the full content details about the currently selected object. This is where all the configuration of objects is done and configuration may be modified actively during running the game; thus also allowing within the environment to rapidly test out features. This aided with unit testing as it saved a lot of custom writing of input functions to then trawl through the output. It allowed for direct entry and response in some cases.

3D Buzz’s first fundamentals videos are the most important to understand; so to summarise them. Games developed in Unity at their highest level are considered to be projects. Each project consists of a number of scenes. At any single time in a game there will be a single scene being shown. Each scene contains a collection of game objects and game objects may have a hierarchy themselves of game objects and components. They may also have assets such as materials with textures, meshes, and scripts attached as components to game objects. These elements provide the interactivity and visualization of what needs to appear in the games. Objects that need to be used regularly such as enemies can be made what are called prefabs so that instances may be instantiated of the prefab to duplicate the existing object. This allows for easy “spawning” of elements within a scene to populate without needing to go into the game and place every single object before the game starts.

Packages in Unity allow the importing and exporting of partial amounts of content. This was used heavily in development as rather than having to hand over the entire project to other team members patches could be distributed including just the changed elements. Packages allow selection of what is desired from the project asset library and then can automatically include any dependent items to save searching through the entire library of game assets. Then when importing the packages the user can be selective about what gets imported.

Unity projects when built for deployment can be exported to a variety of sources. Unity allows for publishing to: web (via a browser plugin), windows x86 and x64, Mac iOS, Android, Xbox 360, PS3, and Wii. Using the free windows version only exporting to the web and windows versions is supported. The pro version of Unity allows expanded exporting to the various other environments though. The pro version of Unity also allows use of a host of extra more advanced features for further customisation of games. The features available in the free edition are robust and useful for making games though. Unity automatically allows for customisation of display settings and graphical quality making it easy to tailor games to fit many different computer configurations. Interfaces for example though do need to be specially customised to allow for different resolutions, but that is out of the scope of this particular overview.

The tutorials available on 3D Buzz help, but the Unity documentation that is provided on their main website provides detailed information on the uses of individual components provided you know what you are looking for. The application even provides links directly to the web from within Unity to these help documentation pages. Unity Answers the official question answering discussion board for Unity is an invaluable resource too. Many individuals have asked questions that related to similar tasks that most would encounter troubles with during development. Often this is how the specific functions and approaches could be found for development.

Unity was for the most part an easy to use development platform that provided a wealth of tools to make games developed with it appear visually stunning with a lot of functionality. This summary was just an introduction to Unity and what it is capable of on a very high level. The following sections of information provide further detail on the research that went into interface development and features provided by Unity that were used within the development of Stargate: Galaxy.

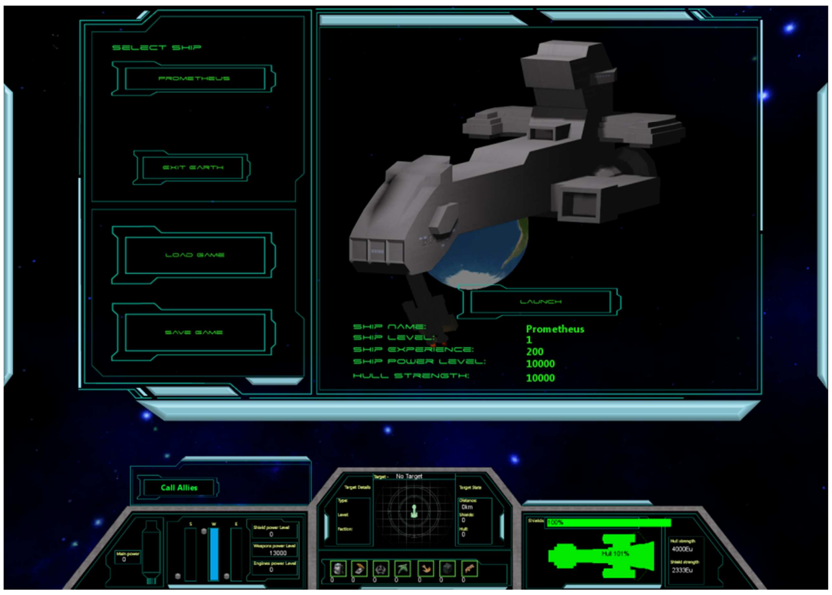
4.5 Interface development in Unity

There were two main elements involved in the development of the Graphical User Interface (GUI), first was graphical second functional. Unity was chosen as the desired development environment because of its ability to handle all elements of the proposed 3D game. It was assumed that the API supplied by Unity would handle all the functional requirements for the Stargate game but no previous knowledge of the API was known. Adobe Illustrator was chosen for the development of the graphical side of the GUI but there was also no previous experience in using this environment.

The GUI was constructed in four steps, plan, design, create and test. The planning stage involved listing all the function requirements of the GUI, from ships controls to ship information as well as menu items loading screens selection boxes. The general styles and ideas for all the assets created were taken from the Stargate TV series mainly Stargate Atlantis. The designs used for computer displays in the shows were adapted for use in the GUI. The storey board was the first step in demonstrating the general concepts of each GUI element. After all the ships functions were listed and all required elements documented the graphic design started.

Before work could begin on creating the GUI elements digitally all required objects were designed by hand. Once all elements were created it was possible to create digital copies to be used in the game. Adobe Illustrator was used to create all of the graphical assets, to be able to create what was needed a proper understanding of the development environment was needed. Websites such as Adobe Illustrator Tutorials(Adobe 2011), the most important tool used to draw all the elements was the pen tool. A tutorial video by Tutevids(tutevid 2007) was used to gain a full understanding of how to properly use this tool. After extensive research into most of the features available in the adobe illustrator environment each of the GUI elements was created and imported into the Unity gaming environment.

As stated, it was only assumed that unity would be able to handle all the requirements of the designed GUI no actual knowledge of the environment was previously known. Extensive research into the gaming environment was needed before it was possible to create such a complex GUI. The Unity script reference guide(Unity 2011) was the most important source of information for learning the API. The guide shows all the classes and all functions for the environment with examples of each and short descriptions. The Unity community forums(Comunity 2011) was also a very helpful site for seeking answers to problems and issues with anything in the GUI environment. The language used to code all the elements was C#, while there was a basic understanding of the language before beginning this project many of the requirements required a much more in-depth understanding of more complex classes and methods, the Microsoft development Library(Microsoft 2011) is a similar resource to that of the unity script reverence page. All classes and methods are heavily documented with examples and descriptions how to use each class. When developing the code for the GUI these references were constantly used through each step, whenever a new feature was needed or a new issue arose the community would have an answer for almost any problem or question. During programming the GUI was constantly tested and corrected, each time a new element was created or altered it was thoroughly tested. The GUI design was also changed as new requirements were needed and as the scope changed elements left out.



SHIP NAME: Prometheus
SHIP LEVEL: 1
SHIP EXPERIENCE: 200
SHIP POWER LEVEL: 10000
HULL STRENGTH: 10000

Figure 6: The Earth Menu

The first and most important thing to learn for developing a GUI in unity is the OnGUI method. This is the only method unity will recognise GUI commands in, if a GUI call is used in any other method it will return a compile error. When first learning how to load the GUI images multiple methods were used such as directly drawing a texture, loading a texture within a box or within a button. After reading over forums of people asking similar questions it was found the best method for drawing GUI elements for this project was to use GUI groups. This allowed for relative positioning of all controls within that particular element such as the bottom control panel. Unity was also able to directly draw slider bars which were the preferred method for controlling the ships energy resources. Other important objects were buttons, text labels selection lists and windows. As there are many ways to do things within unity allot of thought and research went into finding what would be most suitable for the specific requirements needed for this project. Another example was the buttons, for displaying textures within a button element it is possible to directly draw the image within the button or display text, because of the need to display both an image and text an alternative method was required. From reading through the forums it was found that a style can be added to all GUI elements including buttons which allowed for both text and an image on a button.

When starting this project there was little or no understanding of the graphical or functional development environments. To have been able to produce this fully functional and graphically pleasing GUI a large amount of research was conducted into all areas of GUI production. Knowledge within the Unity environment, Illustrator environment and C# language has greatly expanded since the beginning of this project.

4.6 Gameplay elements in Unity

Unity features a wide range of inbuilt elements that assist with handing game objects in three dimensional spaces. There are also features that allow for simpler and greater control over the properties of game objects and the entire game in general. Below is a list of the elements from the unity documentation that were used during the development of Stargate Universe.

Scene

http://unity3d.com/support/documentation/Manual/Creating%20Scenes.html

A scene may be seen as a level or individual screen of a game and includes all the game objects and components.

Physics

http://unity3d.com/support/documentation/Manual/Physics.html

Unity utilizes the NVIDIA PhysX physics engine to allow for more life-like behaviour of objects.

Rigid Body

http://unity3d.com/support/documentation/Components/class-Rigidbody.html

For an object to act according to physics, a rigid body must be attached. This allows for properties of each game object to be defined such as the mass or if the object is affected by gravity. A game object that has a rigid body is not normally moved according to the transform but instead by added forces or torque which provides a more realistic movement. A rigid body can be added though the Components -> Physics -> Rigidbody menu.

Colliders

http://unity3d.com/support/documentation/ScriptReference/Collider.html

A collider can be attached to a game object so that collisions can be detected. Only collisions between two game objects with colliders attached will be detected. When a collision occurs three trigger messages will be sent for each of the objects: On Trigger Enter, On Trigger Stay and On Trigger Exit. It is possible to override each trigger message however if a rigid body is attached to the objects then the default triggers will act according to the physics engine. A Collider can be added through the Component -> Physics menu. There are two primary types of colliders: Mesh and Primitive Colliders. Primitive Colliders are a specific geometric shape and include box, sphere, capsule and wheel shapes. A Mesh Collider is automatically generated using the mesh already attached to imported models however is only useful for objects that do not move as two mesh colliders cannot interact with each other.

Layers

http://unity3d.com/support/documentation/Components/Layers.html

Game Objects can be placed on separate layers in order to make ignoring specific game objects easier. Layers are used in both layer masks and culling masks, the latter is so cameras are able to only render parts of each scene. A new layer can be created though the Project Settings -> Tags menu.

Layermask

http://unity3d.com/support/documentation/ScriptReference/LayerMask.html

A layer mask can be used to selectively remove layers from calculations thus ignoring specific game objects. This is done by using a bit shift according to the layer(s) that should not be ignored; 0 represents an ignored layer while 1 represents a valid layer.

Ray

http://unity3d.com/support/documentation/ScriptReference/Ray.html

A ray is an infinite, and by default invisible, line that originates at a specific point and travels in a defined direction. It is possible to return a point at any distance along the ray.

Raycast

http://unity3d.com/support/documentation/ScriptReference/Physics.Raycast.html

http://unity3d.com/support/documentation/ScriptReference/Collider.Raycast.html

A Raycast allows for the detection of a collision between a ray and a collider. If a game object is hit by the ray the Raycast returns true and the colliding game object will be stored in a variable to be used as a reference if need be. Physics Raycast allows for the ray to collide with all game objects in the scene but layers can be ignored using a Layermask. Collider Raycast will only allow the ray to collide with game objects of the same collider type, ignoring all others.

Prefabs

http://unity3d.com/support/documentation/Manual/Prefabs.html

A prefab is a method of making any game object reusable. A prefab can be used in any scene and can be duplicated any amount of times. Changes to a prefab will cause the change to be seen in each and every instance of that prefab being used. All attributes and attached scripts or components on the game object are also included in the prefab. New prefabs can be created using the Assets -> Create -> Prefab menu and then having the Game Objet dragged and dropped into the newly created Prefab object.

Camera

http://unity3d.com/support/documentation/Components/class-Camera.html

The camera is the component through which the scene is viewed. A camera has various properties in order to provide full control over what and how something is rendered. Cameras also support scripts that provided further control over how the camera acts. Unity allows for an unlimited amount of cameras in each scene, each of which can be scripted to perform different tasks or render different objects.

Audio Clip

http://unity3d.com/support/documentation/Components/class-AudioClip.html

An Audio Clip is a component that allows for audio files of formats .aif, .wav, .mp3 and .ogg to be imported into unity. The properties provided though using Audio Clips allow for support of features such as three dimensional sound, gapless looping and specification of loading settings such as compressed in memory or streaming. Each Audio Clip is played back though an Audio Source.

Audio Source

http://unity3d.com/support/documentation/Components/class-AudioSource.html

If three dimensional sound on the Audio Clip attached to the Audio Source is enabled the Audio Source will play back the clip according to its position in the game world. This also allows for additional features such as defining the falloff curve according to distance from source and applying the Doppler affect for moving Audio Sources. An Audio Source can be created from the Component -> Audio -> Audio Source menu when an empty game object is selected. An Audio Clip then needs to be assigned to the Audio Source.

Skybox

http://unity3d.com/support/documentation/Components/class-Skybox.html

A skybox is a set of six images that are used to either create a sphere or box around the game world. The skybox can never be reached but is always persistent and can be seen from every angle inside the game world. A skybox can be created through the Component -> Rendering -> Skybox menu and the images need to be added to their respective locations: front, back, left, right, up, down.

Light

http://unity3d.com/support/documentation/Components/class-Light.html

Lighting can be used to simulate all types of lighting such as natural light from sources like the sun or fire effects or from unnatural sources such as directional flood lights or florescent lighting. There are three main types of lighting: point lights that illuminate all directions, directional lights that illuminate in an infinite direction and spot lights that illuminate in a cone shape for a specific distance. Properties allow for features such as glows, flares, halos and shadows to be specified for each light source.

Materials and Shaders

http://unity3d.com/support/documentation/Manual/Materials.html

Materials and Shaders cannot work properly without one another. A Shader defines how an object is rendered and any render settings applicable to that object. Meanwhile a Material defined which texture and colour to be used for the rendering. Creating a new material though the Assets -> Create -> Material menu also creates the Shader on the same game object. Through the Shader properties, it is possible to make the rendering self illuminated or even reflective among various other settings that can be used to slightly alter the game objects rendering.

Transform

http://unity3d.com/support/documentation/Components/class-Transform.html

Transform is a component that is given to each game object and is used to determine that game objects position, rotation and scale. The Transform can be used to move and rotate an object manually if the game object does not have a rigid body attached.

Scripting Overview

http://unity3d.com/support/documentation/ScriptReference/index.html

Unity uses scripts that can be attached to any game object or component and can be written in JavaScript, Boo or C#. Each script can use different functions but generally use the predefined Update, Late Update and Start methods available to each script.

Particle System

http://unity3d.com/support/documentation/Manual/Particle%20Systems.html

Particle Systems are used to render a two dimensional graphic in respect to three dimensional space. Textures attached to Particle Systems are rendered multiple times according to the preferences set for each component of the Particle System. A Particle System can be created using the Components -> Particles -> Particle System menu. Each Particle System has three components that allow for control over how the particles are rendered and how they act in the environment: Particle Emitter, Particle Animator and Particle Renderer.

Particle Emitter:

There are two Particle Emitters that may be used determine how the particles spawn. An Ellipsoid Particle Emitter spawns particles inside a sphere while a Mesh Particle Emitter spawns particles around a mesh for more complicated shapes. For both types of Particle Emitters properties regarding size, lifetime, amount, speed, velocity and more advanced features can be altered to achieve a desired effect.

Particle Animator:

A Particle Animator allows for the dynamic movement and colour alterations of particles by applying colouring, force, rotation and growth properties for each Particle System.

Particle Renderer:

The actual material used for the particle is defined in the Particle Renderer where how exactly the material is rendered for each particle.

A Particle Collider may also be added so the particles are able to interact with other game objects that also have a collider. Particle Colliders act the same as normal colliders except have different properties such as bounce factor, loss of energy and minimum velocity for each particle.

These many different types of components that make up core functionality of game objects in Unity were all important areas of research. Further information may be found about these features at the provided links. Unity answers is also particularly useful for when there are individuals who have asked questions similar to what you a developer need to accomplish..

5 Design and Development

5.1 Initial Concept

The original proposal for the game was designed and written a couple of weeks prior to the commencement of the topic after a group had been formed who could work on it together. This document may be found in its original form on the provided disc along with the rest of the documentation for more detailed comparison to see the stages of re-scoping the project.

The original concept was centred on the player as a marine from Earth. The game was set in the universe as seen in the Stargate TV franchise with a focus on those races that were encountered within the Milky Way galaxy including the Goa'uld and Replicators. The Asgard too, playing a part as an ally to assist the player in their travels. The intent was that the player could traverse the ground of planets where the player could encounter alien races and discover information and parts that could be used for the main part of the game. The main part of the game was intended to be the space combat and the travel between worlds via Stargates a sub-plot to allow the player a more expansive universe to explore. From the initial documentation the outline of the scope was given as follows.

- Space combat
  - Including more detailed ship management with hull breaches, air management, power management and other ship related activities/issues.
- Transportation to and from pre-created worlds. Facilitated by Ship or Stargate transport.
  - Including the acquiring of technology and learning of more Stargates that may be found out in the unknown galaxy beyond. Only a few pre-created partial worlds may be included to give the feel of the potential of the full scale game.
- Artificial intelligence that is capable of making strategic and diplomatic decisions.

The initial concept also outlined a lot of high level requirements about each of the key areas. Mostly this expanded on the details of intended scope with a more specific list. Some of the high level points of interest in the initial concept that were introduced as possible gameplay elements included:

- A space environment that gave a sense of enormity, with random encounters, enemy bases, random planet returns, asteroids, and crashing into planets as a possibility.
- Ships were expected to give control to the player over: shield management with multiple areas of shield to manage, weapons management with different types of weapons, life support particularly having to close hull breaches, power management coordinating all the different systems, communications and sensors with weapon targeting controls, engine management, and repair management.
- Weapons/Shields/Hulls were expected to have varying advantages and disadvantages when in different combinations. The weapons were expected to miss potentially, and ships also have an arsenal of fighters that they could also deploy.

- A means of landing on planets using ships.
- Planet surfaces would have control of a character from first person view with collectable information, a stargate to transfer between worlds, ship dock and other buildings around to provide detail.
- Diplomacy between races to allow actions such as an alliance, cease fires, requesting assistance, and other diplomatic actions.
- Trade to acquire information, parts, or ships by communication with the Asgard.
- Artificial Intelligence that was capable of a lot of different actions.

As may be seen from this initial concept list of requirements the original concept was a lot more than what would have been accomplishable with such a small group. Many of these features were cut out as is discussed in the following section.

5.2 Final Concept

The final concept used for Stargate: Galaxy is a very trimmed down version of the original concept. The original concept was found to be far too large not only in details but in the wide scope that was defined. Originally there was to be both ground and space combat where the player could not only control a ship but also a human marine. Due to the fact that this would have basically been two separate games merged as one it was decided to remove the ground combat part. In the final concept the play only controls a ship in space and is only able to participate in battles in space between various other ships. However, this posed a problem of a lack of types of enemies that may be encountered. Oringinally only the races from the Milky Way galaxy were to be included. Those races being Human, Goa'uld, Replicator and Asgard, however, it was decided that two more races should be added for variety, the Wraith and Ori.

In regards to the space combat and ship management little actually changed. The player is still provided with full control of their ship; however, some of the minor control features were removed. Such features include multiple areas of shield management, life support, having to deal with hull breaches and finally being able to deploy fighters. The decision to remove these features was decided upon due to the time and skill set restriction placed upon the development group. Smaller fighters would have not only taken more time to script but also to design the models and textures. The other features were simply too in depth and would have detracted from the rest of the game. Another feature that was dulled down was the diplomacy between races. While there is diplomacy in regards to having friendly and hostile races. There is no way to alter the state of the diplomacy and calling for assistance is done on a dice roll basis rather than if they like you enough.

The features added to the game due to specifying space combat only also include the use of a pair of Supergates rather than Stargates and the ability to move more freely between star systems. Not much more was added as the scope was already large enough to easily take a small group half a year to make.

5.3 Design Overview

5.3.1 Overview

Over the course of developing Stargate: Galaxy a number of design changes were initiated and also the approach to the development. As may be seen from the previous two sections the scope was modified heavily during the early stages of development. After the early stages the general scope was held the same and only elements that needed to be taken out due to time and resolutions to ambiguities were modified. We were careful on the whole to not scope creep and therefore lose any hope of ever completing the project. The details of the actual design and implementation are split between the System Requirement Specification and System Design Documents. These two documents together define most of the processes and structures used. It was a deliberate decision not to detail everything in further detail than has been given as full documentation would have served no purpose more than to be a paper weight. Spending too much time on documentation would have slowed down the implementation phase of the development even more than it was due to complications. A summary of the actual project management and how the time was managed may be found in section 5 of this document.

The purpose of this overview of the design is not to just reiterate everything from the System Design Document and other documentation, but will draw on the information that is there and explain the process of iterative design that was used akin to a spiral model. The design process was primarily iterative as it was a continual learning experience becoming used to Unity. The software engineering topics in particular assisted with the processes of thought that went into how the system would be structured and modifications were made to the design when it was decided certain changes needed to be made for it all to work. For this project it may be considered that there were 5 stages that occurred for the design. For each of these a brief summary is supplied and how the design was influenced in each stage.

5.3.2 Iterations of Design

5.3.3 First Stage – The SRS

For the game Stargate: Galaxy, the development of the Software Requirements Specification document was not only a list of requirements that needed to be included in the software product. It facilitated a collection of designed series of functional requirements for the structure and process of the overall system. In particular the activity diagrams that may be found in the appendix of the SRS covered functionally how the system would process. Since within Unity objects are defined and most of the development is functionally orientated toward objects and their goals, these processes heavily aided the first steps of the design. The requirements defined the need for particular game objects such as the planets, ships, projectiles, and other elements and how they would need to interact in the larger system.

During this initial stage it was known what general data was required, that there would need to be some form of controller object that would manage the scenes, and the lists of assets that would need developing. Right from as soon as possible the assets were being developed for the ship models that would player the main visual role within the game as these would take the longest to develop. Originally a larger set of ships were to be included with fighter ships that would fly out of the larger ships and engage enemy fighters and the target ship. This was later scraped as a design element as it was deemed to have pragmatic and time oriented issues. Another race that was originally to be included during this earlier stage was the ancients and for them to have their own ships too. Ideas like this were removed late during the SRS stage as having only one digital media student restricted the quantity of work possible.

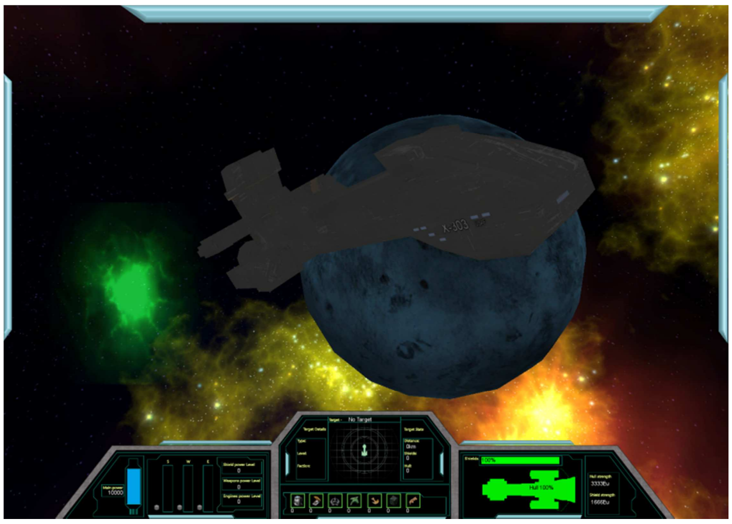


Figure 7: The Prometheus exiting hyperspace

5.3.3.1 Second Stage – Getting to Know Unity

After conceptualising how the software product could be developed the next significant stage was everyone introducing themselves to Unity. This included a lot of the actual research. This stage could perhaps be considered the most critical of the stages as it was when Unity was properly tested out and learnt about. In addition to learning the general information about how the editor was laid out, we each experimented with the system to find out what was possible and how easy it was to implement different types of functionality. One couple of the outputs of this phase may be seen in captured video that Peter released on his youtube channel.

The videos being:

First shot captured of AI in unity:

http://www.youtube.com/watch?v=JNjNsBkX00s

Unity Hyperspace Window First Effect Test:

http://www.youtube.com/watch?v=Vm836K3iNLM

The first video demonstrates an initial test that included demonstration of:

- Placing meshes into Unity
- Attaching those meshes with correct materials to Game Objects
- Using lighting to show up the meshes
- Some basic scripting that controls an object and makes it travel between a pair of objects endlessly

It was a particularly important initial test as it provided information that indicated it would be possible to use the approach as was being designed.

The second video is of lesser importance, but is still significant. The scene contained a planet that demonstrated bump mapping and a first particle effect for the hyperspace window. There were many elements in the final product that used particle effects so this first attempt at making effects was important in furthering the process. By the time these two videos were completed enough had been learnt about Unity to properly design and construct the system.

5.3.3.2 Third Stage – Design and Implementation

The third stage involved much of the time in meetings resolving ambiguities. The meetings prior to this stage had of course included handling of ambiguities; they were more centred on exactly what would be in the project scope wise. The data required for each component and the list of game objects with their properties was drafted during this stage and filed into the initial draft of the system design document. The particular phase was short lived as there was a major ambiguity where the lead designer had thought a piece of information had been decided on differently. They had thought that the decided approach was that access would only be to a single planet at a time. This meant that the design was lacking as what was actually intended was that all the planets in each system would be available to travel to. And that they player would still need to jump between different systems to visit planets in other systems.

5.3.3.3 Fourth Stage – Revisions to Design and Continued Implementation

After the major design change had been corrected. It then allowed for a system where multiple planets could exist concurrently in the game world each with their own AI ships associated. This structure was fundamental to how the objects could exist in the game and is how the design was kept. Only minor changes to specifics of functionality really changed from this point and elements were continually added to increase the content of the product.

Before anything was actually put together the high level objects, data storage mechanisms, required scenes, and other important elements had all been decided on. Everyone knew the goals that they had to achieve at each stage and worked tirelessly to produce high quality work. The two major elements of the game, the interface and the 3D world, were developed separately till it was no longer possible to just test the software elements apart. There was a lot of the main unit testing conducted during this stage prior to the integration of the interface and the 3D world. The integration was very successful when it eventually occurred and it was fulfilling to see the work coming together.

Iterations of the builds continued more frequently as patches to content needed to be applied. Generally fixing the issues in the game that were presented were related to typos or incorrectly written functions and null pointers. Unity's debugging system as briefly discussed in the relevant research section did not provide as helpful as would have been desired debugger. It was manageable and issues were discovered and corrected in a timely manner.

Another video that was released during the later stages of this phase was the following video that demonstrated projectiles being fired for the first time.

http://www.youtube.com/watch?v=Uq1e1q4aE_E

Projectiles were added into the game at a late stage as the movement and AI mechanics were considered more important as a first order of business. The code was already written earlier for projectiles to work, it was just that the projectiles needed their own special objects to work so had been delayed. Ideally they could have been developed more alongside the rest of the ship content during earlier development.

During this stage population of the database was completed. The database was quite a large feat. There are many thousands of lines of code that generate the contents of the database dynamically as a fresh copy when required. This dynamic generation approach to the population of the database was important as manual population would have required filling in over 4000 attributes with appropriate data. This entire population process was aided by the writing of code that generated code itself for the automatic population. The database population code may be found within the DataLoader.cs class in assets under scripts in the Game Controller scripts folder.

5.3.3.4 Fifth Stage – Tying Everything Together

The final stage did not contain any real significant design changes as the product was mostly complete. It was most related to modifications to visual elements to ensure they were as originally intended. So polishing and bug fixing took up a lot of time in preparation for submission of the product.

5.3.4 The Design

The Figure that follows below demonstrates a basic level representation of the system. The HUD is displayed by using information from the game. The game loads its data with the data loader and uses the structures defined by the Data Manager. The Game Controller maintains an AI Controller for each planet. So each AI Controller manages the collection of ships, wrecks, and a planet in space. Additionally ships fire projectiles. For the full details please see sections of the System Design Document related to each of the specified components. Also see the design document for the broader and more detailed definitions of the design.

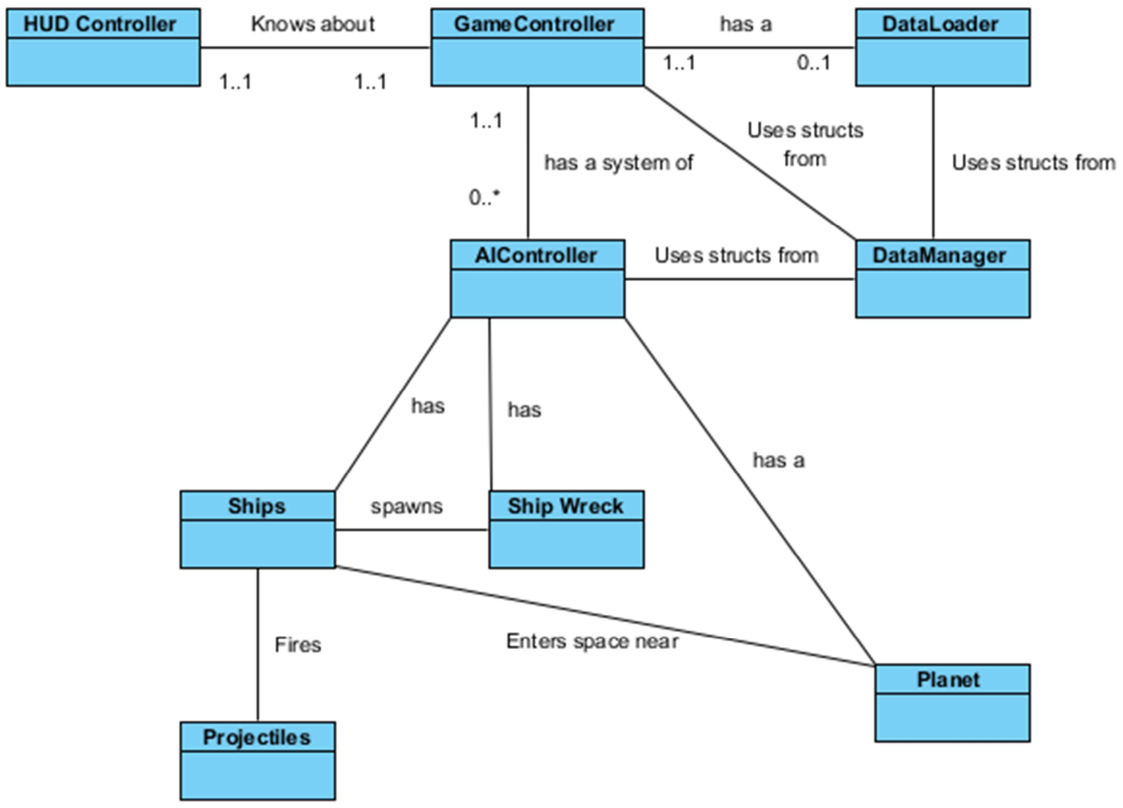


Figure 8 : The Design

5.4 Use of 3rd party assets

Stargate Galaxy, as in all games, needed certain assets in order to pull everything together as a complete product. Due to the skill set of the group producing Stargate Galaxy, it was not possible to have all assets produced from scratch for the game. This meant that certain assets had to be outsourced from free repositories that release assets under creative commons or similar licences. For Stargate Galaxy specifically, the assets that were outsourced were all the music tracks and planet textures as well as some of the audio files used.

Music

Due to the skill set possessed but the group it was not possible to have original music compositions. Fortunately, composer and producer Jared Kraft graciously allowed for the use of his own compositions in Stargate Galaxy.

Licence: See attached email.

Source: http://jaredkraft.com/

Ambush – The Fused – Jared Kraft

Eternal – The Fused – Jared Kraft

The Dance – The Fused – Jared Kraft

Not A Friendly Place – The Fused – Jared Kraft

Orphans Cry – The Fused – Jared Kraft

Pulse – Jared Kraft

Sound

Some sound effects needed to be outsourced, the list below specifies what files were outsourced, from where and who as well as under what licence the asset is being used under. Please note that most compositions have been altered and are not used in their original form but credit is given where credit is due.

Nathan Shadow

Licence: http://creativecommons.org/licenses/sampling+/1.0/

Source: http://www.freesound.org/people/nathanshadow/

Thruster_Level_III.aif

Thruster_Level_II.aif

Thruster_Level_I.aif

Inferno

Licence: http://creativecommons.org/licenses/sampling+/1.0/

Source: http://www.freesound.org/people/inferno

Hvylas.wav - http://www.freesound.org/people/inferno/sounds/18382/

Hvrl.wav - http://www.freesound.org/people/inferno/sounds/18380/

Smalllas.wav - http://www.freesound.org/people/inferno/sounds/18397/

Slighlas.wav - http://www.freesound.org/people/inferno/sounds/18396/

Jamius

Licence: http://creativecommons.org/licenses/sampling+/1.0/

Source: http://www.freesound.org/people/Jamius

BigLaser.wav

Frank Bry – The Recordist/Creative Sound Design.

Licence: Royalty Free Downloadable Sound Effects Collections.

Source: http://www.therecordist.com/free-sfx

Missile_SE.mp3

Missile_Rip_By2_SE.mp3

Laser_Burst_Gun.mp3

Explosion_Missile_Impact_SE.mp3

Laser_Beam_Crystal_SE.mp3

Planet Textures

All planet textures were outsourced from various locations where all the textures have been released under creative commons or by expression of artist are free to use for non-commercial purposes.

James Hastings-Trew – Planetary Pixel Emporium

Licence: http://planetpixelemporium.com/planets.html

Source: http://planetpixelemporium.com/index.php

All available textures.

Seth Pritchard

License: "All textures here may be used for non-commercial purposes so long as their soruce is cited."

Source: http://www.celestiamotherlode.net/catalog/show_creator_details.php?creator_id=211

82 Eridani - Upbringer Universe.

Craig Strapple

Lisence: http://creativecommons.org/licenses/by-nc-sa/3.0/

Source: http://www.celestiamotherlode.net/catalog/show_creator_details.php?creator_id=209

Castra.

Imperium Atlantis.

Milosz21

Licsense: http://creativecommons.org/licenses/by-nc-sa/3.0

Source: http://www.celestiamotherlode.net/catalog/show_creator_details.php?creator_id=172

Avatar_System.

Daemon_System.

Hameed

Licsense: “Free to use for any purpose, no permission required.”

Source: http://hameed.deviantart.com/art/Planet-texture-maps-01-116701701

Planet Texture Maps 01.

5.5 Testing

The Stargate: Galaxy testing team consisted of members of the development team who conducted the tests based on a number of test cases which the development team created based on the SRS and System Design Documents, along with Unit and Integration Testing. The development team did not have time to complete the integration tests, or complete code walkthroughs. These tests would have helped to ensure that the code was better optimized. The tests that were completed allowed the development team to cover large sections of the program and ensure that the main components worked as intended.

5.6 Success of the product

A group leader was chosen unanimously on day one, this person had the skills needed to manage the rest of the group as well as having the respect from all the other team members. Every one fell into their roles relatively quickly and we all agreed on the amount of work we wanted to put in. The success of this project came from a group of people that all wanted to achieve the same high level of work. The main reason why the group worked so well was because of a leader who effectively delegated all work to the correct team members, was able to keep everyone on task while still retaining respect from the other members. While there were some heated discussions every one allowed others to voice their opinions and everything brought to the table was discussed.

When the group was first formed we discussed the creation of a Stargate based game, there was no shortage for ideas. The group envisioned a game that included epic large scale space battles and well as land combat with the ability to transition between space and surface combat, in depth faction reputation systems, advanced trade economy just to list a few. A good part of the first few weeks after forming the group involved discussions on what was possible to include given the group's skills and time frame allowed. Features were constantly added and removed as the group decided on what we wanted to included, what we could include and what we should include. While many of us had our own ideas on what should and should not be included as a group we were able to compromise and all agreed on what we would create. As a group we agreed to limit the game to a space environment only with a limited trade system and no real faction control, we decided to focus on increasing the space combat and space environment rather than trying to implement both land and space combat. A large deciding factor on what we could and could not do was the abilities of each group member, no one in the group had used Unity (the chosen development environment) so it was not possible to accurately estimate how long each feature would take to create.

Once we decided on the exact features we wanted to develop we documented everything, this allowed the group to know exactly what was needed when it was needed and how it would fit with everything else. During this process the scope was still changing, the more we researched into each item the more accurate we could predict it's time to complete. The documentation provided a vital tool for the development of such a large project it was a reference point for group members to see what each other was doing and how.

While the project was not 100% complete for the presentation the product is still seen as a success. As a group we managed to take on a very large and daunting project break it down into manageable pieces and work as a team to bring all those pieces back together into a working product that exceeded everything we had hoped for. After managing our scope throughout development we were able to produce exactly what we set out to create. The majority of the group went into this project with little or no previous experience with a project of this size and have managed to work around all issues as a group. Each member has gained extensive knowledge from the research into their parts of the project and experience in working with a team of people to take on a large project.

6 Project Management

6.1 The initial plan and how it differed from the actual project

The project management plan that was written initially covered a lot of detail about: who, what, and when. The content was based on the structure that was taught during the project management topic that ran just prior to the commencement of this project.

The following table summarises the original list of milestones that were set out to be achieved. The original expected duration is shown to be paired with an approximate time of how long the tasks actually took to complete. Most tasks took the expected period of time with some just a little over or under the estimates. The major ones that threw off the expectations were the development of the design and the implementation. The implementation took far longer than expected as there were complications with the scope being the size it was even though it was cut down a lot.

Milestone	**Expected Duration (days)**	**Actual Approximate Duration**
Concept Proposal Complete	17	17
Project Management Plan Complete	17	**20**
Meeting Agenda Complete	1	1
Meeting Minutes Complete	1	1
Meeting Action Plan Complete	1	1
Functional Requirements Document (SRS) Complete	13	**16**
Determine Research Topics Complete	2	2
Research Complete	41	40
Research Summary Complete	25	10
Systems Design Document Complete	13	**40**
Model Assets Complete	40	**70**
Other Assets Complete	40	**70**
Code Complete	34	**70**
Test Procedures Complete	12	12
Test Files and Test Readiness Review Complete	13	Test Readiness not done.
Test Analysis Report Complete	9	10
Issues Resolved from Test Analysis Report Complete	13	2 (no time to complete further)
User Installation Guide Complete	6	3
User Operations Manual Complete	6	3
Post Completion Report	12	12
Deliverables Complete	-	-
Deliverables Submitted	-	-
Presentation preparation Complete	5	2
Presentation Submitted	1	1
Personal report or presentation	20	5

Table 1 : Expected duration vs Actual duration

The project was managed well and was kept on track for what was accomplished. A lot of documentation and product development was involved during the life cycle. The task allocations that were handed out initially were partly reassigned as may be seen in the credits. This was all handled based on a scaled work load for each individual so that there was a reasonably equal work load between the group members.

Communication was largely handled through Skype chat and calls to discuss the implementation and integration of features. A copy of the Skype chat logs has been included on the disc to show how much discussion there was that went into the development of the game. The meetings were important too as they brought everyone into the one place and allowed for often more visually based discussion with use of a whiteboard. The agendas and minutes have been included from all the meetings on the disc as well. The minutes have been left as quite rough lists as it was decided the time it would take to write them out formally would cut into valuable development time.

6.2 Lessons learned from the project

During the project the team learnt a number of different lessons. One of the major lessons that the team learnt in the project was that even though the team reduced the scope of the project significantly from its original concept the project was probably still too large for the amount of people that were in the development team and the time frame that was allocated to the project. Another of the lessons that the development team learnt during the project was the amount of extra learning that was required to create various elements of the game, a major example was the creation of the GUI which took significantly longer than was expected to complete. The team also learnt lessons during the project as they tried to complete the game using a professional approach including weekly group meetings. Another of the major lessons that the development team learnt in the project was to ensure that all members of the project had the same idea of what was being created. An example of this that occurred in the project was the development team had different ideas of how the planets would be arranged in systems, some members of the development team thought that there would be one planet per system while other thought that there would be multiple planets in a system, this issue was resolved during one of the development teams meetings.

6.3 Changes that would be made if project was repeated

Were the project to be completed again the development team would make a number of changes so that the project was completed more smoothly. One of the main changes that the team would make would be to increase the size of the project team; the increase in team size would include at least one more digital artist to assist in the creation of 3D assets. Another of the changes that the team would make would be to begin the implementation phase of the project sooner; this would have allowed the team more time to create the game and would have hopefully allowed testing to begin sooner than it did during the project thus giving a more polished end product. The team would have also endeavoured to finish the SRS sooner than it was completed in the current project as the length of time spent on its creation meant that other areas such as implementation and testing had much shorter time constraints.

7 Conclusions

When the group was first formed we discussed the creation of a Stargate based game, there was no shortage for ideas. The group envisioned a game that included epic large scale space battles and well as land combat with the ability to transition between space and surface combat, in depth faction reputation systems, advanced trade economy just to list a few. As a group we discussed and refined all ideas and reduced the concept list to a manageable project for the amount of people, time and skills available to us. There were many revisions of the concept thought develop, the more we learned about the Unity development environment and how much work some of the features required. By first researching and documenting all that would be required we were able to make informed assumptions on how long each step of the development process would take. More work went into research and documentation then actually creating the game but without having first documented and researched what was needed a project of this scale would not have been possible given the time and skill constraints.

What this project taught us most was the importance of research, team communication and planning. As seen in the initial concept the initial game idea was much larger than what we finally decided to create. At the time it all seemed possible and without proper research and planning we would have gone ahead with those ideas only to realise to late that it would have been impossible to complete. Decisions such as what assets to create and source externally, when to have particular goal achieved and how to test the game were all planed before undertaking and gave us a strict guideline to follow.

The final product created reflects countless hours of research and planning, which resulted in a product that achieves all expected requirements concepts at the level we all set. Even after all the planning the project we set out to create was immense; to have completed it within the given time with such a small group is an achievement on its own.

Bibiliography

3D Buzz (2011), Unity Fundamentals, viewed 10/11/2011, http://www.3dbuzz.com/vbforum/sv_videonav.php?fid=1ec1a7be9c0d4cf9e7a31525250a30ff

Adobe (2011), "Adobe Illustrator Tutorials." viewed 6/11/2011, http://www.adobeillustratortutorials.com/

Comunity, U. (2011), "Unity Comunity Forums." viewed 6/11/2011, http://forum.unity3d.com/forums/25-UnityGUI

Eve Online (2011), “About Eve Online”, viewed 6/11/2011, http://wiki.eveonline.com/en/wiki/About_EVE_Online

Microsoft (2011), "MSDN Library." , viewed 6/11/2011, http://msdn.microsoft.com/en-us/library/

Stargate the Ark of Truth, 2008. [DVD] Robert C. Cooper, Canada: MGM, 20th Century Fox.

Stargate SG-1: Season 1 Episode 20, There But For The Grace Of God, 1998. [DVD] Robert C. Cooper, Canada: MGM.

Stargate SG-1: Season 9 Episode 20, Camelot, 2006. [DVD] Martin Wood, Canada: MGM.

Stargate Atlantis: Season 4 Episode 11, Be All My Sins Remember'd, 2007. [DVD] Andy Mikita, Canada: MGM, 20th Century Fox.

Stargate Wiki. 2005. Daedalus - Stargate Wiki. [ONLINE] Available at :http://stargate.wikia.com/wiki/Daedalus. [Accessed 06 November 11].

Stargate Wiki. 2005. Ha'tak - Stargate Wiki. [ONLINE] Available at: http://stargate.wikia.com/wiki/Hatak. [Accessed 06 November 11].

tutevid (2007), "Adobe Illustrator Tutorial The Pen Tool.", viewed 6/11/2011, http://www.tutvid.com/tutorials/illustrator/tutorials/thePenTool.php

Unity 3D (2011), Tutorials, viewed 10/11/2011, http://unity3d.com/support/resources/tutorials/

Unity Answers (2011), All questions, viewed 10/11/2011, http://answers.unity3d.com/index.html

Unity (2011), "Unity Script Reference.", viewed 6/11/2011, http://unity3d.com/support/documentation/ScriptReference/GUI.html

Appendix A: Accompanying Documents

The accompanying documents that may be found on the supplied disc include:

- Project Request (ORIGINAL concept).pdf
  This document is a copy of the original proposal for the project that was originally supplied at the start of the semester.
- Concept Proposal (MODIFIED concept).pdf
  This document is the revised proposal that was written to more formally identify the goals of the project and what it would accomplish.
- Project Management Plan.pdf
  This document covers the plan that was devised for managing the project including how much time was originally expected to be spent on the various activities.
- System Requirement Specification.pdf
  This document formally outlines the requirements for the product in detail and identifies many of the processes required for the system.
- System Design Document.pdf
  This document covers a moderately detailed overview of the design that was used for implementation of the product.
- Testing Documentation.pdf
  This document includes the primary test cases that were checked.
- User Guide.pdf
  This document covers an overview of how to use the product.
- Project Summary Report.pdf
  A copy of this document that is being read currently.
- Meeting Minutes and Agendas folder
  Contains a dated copy of all minutes and agendas that have been produced during the course of the topic.
- Skype Communication Logs
  The majority of the skype communication logs from the group that was set up have been included. Almost the entire log is communication regarding the product. It is possible that there is off topic communication that has not been removed during checks through.

Other assets that are located on the disc include:

- The complete Unity3D project file collection.
  In particular the scripts may be viewed under the Assets/Scripts/ directory. Each of these scripts is sub divided into other folders and they are named based on what she will be accomplishing.
- A pre-compiled copy of both the x86 and x64 builds of the project to allow execution on both platforms.