Planetarius

COMS 30400: The Games Project

By

Static Peak

E. Christodoulou
T. Camp
G. Ginghina
J. O'Connor
T. Michiels
C. Mann

Department of Computer Science
University of Bristol

May 9, 2016

Contents

1 Abstract

We introduce you to Planetarius. A multiplayer shooter where you choose to side either with the greedy Pirates, or the nefarious Super Corp. Throw yourself into battle, blasting your enemies with lasers to collect the precious resource known as Elefthismium. War wages over several planets, from the lush forest planet of Napa, the pirate homeworld of Erimos, to the SuperCorp homeworld Polis. You can join over a network with your friends and fight alongside or against them for supremacy. [1]

Players use their mobile devices to control their players using a twin stick system; one for movement and one for firing projectiles. The aim is to collect and retain as many resources from the planet as possible before the round timer has elapsed, upon which players are shown statistics of the round, and then taken to the next planet. The team with the most planets claimed are declared the victors in the race for the resources.

Players only use their devices to control their players so that all interactivity is directed to one, large screen upon which all players are visualised. This design choice was used primarily to maximise the enhancement that the Planetarium environment brings to the game. We have been gearing our development towards the Planetarium as it is a unique opportunity to showcase a new style of interaction and teamwork gameplay.

2 Novelty Component

The novelty component in our game can be summarised simply as a full multiplayer game visualised inside a highly immersive Planetarium environment, where players engage in battle in a cartoon-like world. The @Bristol Planetarium does not have many fully interactive sessions to showcase, and in particular they have never had a large scale multiplayer game, which are the type of new shows they are eager to promote in the future. We believe we have filled this gap for them and could take this forward as the start of a full scale network infrastructure for interactivity in the dome.

Developing a full network solution, which includes integrating with the Planetarium network has proved challenging both technically and collaboratively with the Planetarium staff. We have worked hard to ensure that all corners have been covered, and that players will enjoy and benefit the most from the unique environment the dome brings. Due to lack of time, we were unable to achieve 3D reliably, however detailed later in the report is a succinct explanation of how this could be added.

3 Project Overview

3.1 Team Introduction: Static Peak

We are a team of 6 Computer Scientists from with a wide skillset, from 3D modelling and graphics to game play or networking enthusiasts. We came together as people who like playing video games and were excited to get a glimpse into the gaming industry in order to create a unique game, whilst discovering a wide range of technologies we had not had the chance to work with before. We were eager to apply our computer science skills to the development of the game, to create something that is technically challenging as well as fun.

3.2 Game Concept

The concept of our game was to allow multiple players to easily connect and fly into a battlefield where all players are visualised on one large screen. The players fly around the surface of a planet to collect and retain precious resources, while defending themselves with projectiles. Players use their mobile devices to control their movement and shooting directions. After every round the team with most resources wins the planet. After several rounds the team with most planets claimed wins the game.

3.3 Tools and External Contributions Utilized

3.3.1 Unity Game Engine

After researching the various open source game engines available to us we settled with Unitys game engine, as it provided us with the tools we needed to achieve our vision of the game. Firstly it is an established Game Engine, used by hundreds of thousand of people. Secondly, the engine's portability was very important to us since we needed to easily transfer our game to Android, Windows and Linux. Also, prior to our team, other developers had successfully ported their projects to the Planetarium using Unity, so this further supported the justification of the engine. Unity's graphics engine's platform provides a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware; and if none are compatible, fall back to an alternative shader that may sacrifice features for performance.

3.3.2 Unity Cluster Renderer

Unity Cluster Renderer is a beta feature of Unity that is used to perfectly synchronize two machines such that they display the exact same image. The system works by frame-locking objects in the game and manipulating objects based on timings, input and random seeds. We reached out to Unity in February to utilise this software as it would be useful for stereo vision. Once we received the software however, we found that it would not work with an existing network layer, and we could not use it. Nevertheless, it provided useful inside into how to implement the algorithm for 3D ourselves.

3.3.3 Intel Next Unit of Computing (NUC)

Considering the game was intended to be heavily network based, it made sense to make use of a dedicated server machine. We were provided an Intel NUC box that runs Ubuntu, used to run the Linux server component. The box simply runs an access point that broadcasts an SSID (given as "Planetarium"). The box bridges wired and wireless and assumes that the wired component has access to a DHCP server. This was chosen so that we could run a network either in the Planetarium or simply next to our laptops.

3.3.4 Planetarium Facilities

A major part of our project involved the @Bristols 3D Planetarium, as one of our main goals was to investigate the 3D capability, creating a truly immersive experience. The planetarium provided us with the opportunity to

port our project to its 4K Dome, and further enhance its atmosphere and immerse the players in an environment that completely surrounds them. The planetarium also boasts complete surround sound on top of this. We had 4 hour sessions every one or two weeks to test out our new ideas.

3.3.5 Music Composers

The Faculty of Engineering cooperated with the Music Department of the University of Bristol and after presenting our game ideas in front of them, we ended up co-operating with two excellent composers, Edward Brown and Matthew Jones. We proceeded with explaining our game's universe and concept to them, in order to fully immerse them into our idea and the result was amazing. They provided us with some magnificent clips for our needs. Specifically, they composed clips for every different planet, the Pirate and Super Corp theme and menu themes.

3.3.6 git

In order to keep track of our development and have a smooth collaboration, git has proved to be particularly useful, allowing for distributed revision control and source code management, as well as bug tracking, feature requests, task management. We have also utilised git in order to make releases after achieving a milestone as well as providing a backup for when things go wrong. This has allowed for a scrum-based approach, basing our development on sprints, often software integration and regular releases.

3.4 Oculus Rift

In order for an easier porting of our game onto the planetarium, we utilised DK2 Oculus Rift for development purposes. This has allowed us to simulate the fisheye screen in the planetarium and get a better idea of how the game would be displayed onto it without physically being at the planetarium. This saved the team a lot of time since we didnt have to spent time needlessly during the Planetarium sessions implementing micro modifications to the user interface and instead focus on the more important aspects of the project like testing networking and gameplay.

3.5 Android and iOS devices

Our game was intended to be playable on android and iOS devices. Due to the Unity engine providing built-in support for cross platform gaming this step was simplified for us. The difficult task was distributing the packages correctly. iOS does not allow packages to be distributed unless they has been signed with an accredited developer license which costs a sum of money. Most of the devices we support are therefore Android, as there is an option to install unknown software.

3.6 Microsoft Controllers

We were also able to add in support for Xbox controllers on Windows clients. This allowed further accessibility to the game even when an individual cannot use their phone, although it is still more convenient to have a mobile as this is how the game has been targeted.

4 Software Manual

4.1 Running the Game

To run the game the individual must first have the server, observer and player executables. The server will run dedicated on a chosen device so that observers and players can connect to it. The observers (usually launched on Windows) and players (usually on mobile devices, although can also run on windows) can enter the IP (if required) to connect to the correct machine running the dedicated server.

Ideally the individual will connect the observer first so that players can see their team choices updated as they choose a team on their personal devices. Once all the players have joined the observer can click launch at which point all devices transition into a running state and the game launches into round one. From this point the game requires no more management and plays out for 3 rounds.

4.2 Adding New Game Functionality

To add new functionality to the game it is ideal that the developer uses Unity Engine 5.4.3+ to be able to build for Windows, Android and Linux. The code for the game is separated into several folders relating to the game component required. For development related to gameplay flow the developer should first start by exploring the Start, Lobby and Round scenes. All scenes contain component scripts attached to objects which control the flow of the game. The scripts attached to these objects are descendants of MonoBehaviour which means they exist in the context of a scene and relate to an object in the scene.

To change more core aspects of the game such as networking or rounds, the user can navigate the folder structure which is separated by functionality, and then by purpose. For example there is a Network folder which contains many sub folders relating to player object synchronisation, socket connection information etc.

Moreover, most of our features were implemented such that adding on would be a matter of simply inserting a new line of code. For example, the round system is based on having an array of all of the scenes we would want to play in order they must appear, calculating the transitions, round stats and gameplay would automatically be updated. Thus, in order to insert a new round or a new planet, it would be a matter of adding the scene name into the array of scenes in the round manager and this would enable further rounds.

4.3 Software Maintenance During Development

Software maintenance was achieved using git as source control and GitHub as the hosting medium. Throughout the project a new feature, fix, or experimental code would require a new branch on the remote medium. When the branch had fulfilled its original goal it would be reviewed by the team to ensure that all team members were up to date with the new changes on this branch. When the team was happy with the final changes the branch is then merged into the default development branch. When we were happy with the state of the development branch we would make a release to the master branch, which is the latest playable version of the game.

5 Technical Content

5.1 Unity Preamble

It is useful to first introduce the way Unity handles objects to fully understand how problems have been solved throughout the project. Every object in the game contains a Transform. A Transform simply contains **position**, **rotation** and **scale**, which are all 3-dimensional vectors. It is also possible to query the relative **up**, **right**, and **forward** direction vectors, used commonly in 3D graphics.

The networking layer is easily explained as sending and receiving on channels. There are two main channels: reliable and unreliable. The reliable channel ensures that the packets reach their destination through acknowledgement protocols. The unreliable channel, as it sounds, does not guarantee that the packet arrives at the destination.

5.2 Planetarium Challenges

Since February we have been having frequent sessions in the Planetarium usually for 4 hours in the evening. The Planetarium houses two main systems for projecting images onto the dome; Digistar and Vioso. We were in charge of using Vioso which uses two servers, Master and Slave. The purpose of having two servers is for the potential for 3D which the Vioso system achieves by simply overlaying the the dome projection from Master and Slave.

The projected image comes from two 4k projectors placed at the front and rear of the dome, which blend along the middle to the dome to produce a single hemispherical environment. The images are accompanied by a 7.1 surround sound audio system providing a uniquely immersive experience for the audience.

Projecting onto the planetarium was one of the major problems that the team had to solve. Two solutions were investigated, a simple shader with one camera, and a 5-camera setup. With the simple shader, one camera is applied with a fisheye correction so that pixels are shifted according to the rotation and distance from the center of the eye. The difficulty with this approach is that along the sides of the eye there is a distortion effect as the camera in most cases only has a 60° field of view. Therefore we had to investigate another solution.

Through research into how others had solved the Fulldome (as it is known) view, we found one open source solution which uses 5 cameras, and projects each camera's viewports onto adjacent meshes, which stitch together to create a full fish-eye effect that has 240° field of view.

One of our main challenge in porting the game onto the dome view was accounting for a different camera view all throughout the project, having to develop for 2 completely different views (dome and normal screen). Thus, in order to allow for an easy switch, we implemented both views into the same project, allowing for these to change using a toggle. This meant that every feature added had to account for both versions of the game. Moreover, it led to certain shaders and textures not being included in our game due to not being suitable on one of the views.

5.3 Networking challenges

One of the major challenges for us was making the game fully networked, allowing for many players to play together for their own mobile devices. We require a fair amount of data to be sent and processed by both the server as well as the player clients and observer clients. Movement, rotation, deaths, kills, scores and game updates all needed to be synchronised and updated for all players. We initially started by using the Unity Networking structures available to us, which allowed for us to establish the basic functionality for the game in good time. Once the basic plan layout had been established, we were able to test the game in a basic form, which was helpful in assessing the current gameplay and how to improve on it.

We realised at this stage two things: firstly that we needed to focus more of the game on the observer view, and that the Unity Networking structures we'd used were far from optimal for our usage. From here we worked on replacing the Unity structures with our own, allowing us to create very specific networking functionality for our game with single message passes between only the required devices. By moving a lot of the computation to the observer displays, we were able to reduce the amount of network traffic significantly. By making the players only visible on the observer view, we reduced the need for all clients to receive constant updates as to player actions, only needing for the observer to be updated with these changes. Everything besides deaths could then be dealt with on the observers, be it interactions with resources, events, or scores. Even events which seemingly would require some network traffic, such as the rotation of player objects, were with some thought and mathematics able to be purely calculated on the observer.

The network uses a star topology. The dedicated server is essential to ensure that data from all clients is acknowledged on a fair basis, which may not be the case if running in parallel with graphics. In addition, it allows for a smooth transition into 3D in the planetarium, as the server can fairly estimate network latency. There are two main scripts that enable synchronizing the position and rotation of objects between server and clients. In our final design, all clients send their positions to the server after some threshold has passed, and the server passes these on to the observing clients.

A final touch to all of the object synchronization is to make the position and rotation updates smooth. This is achieved through linear interpolation between either vectors or quaternions on the observer screens in every updated frame. A notable bug that caused headache was that of a certain case where the delta-time between frames was much higher than usual. The delta-time is used as past of the linear interpolation to indicate the percentage completion of the interpolation. Unfortunately when delta-time is combined with a interpolation rate this caused the value to be greater than 1, which introduced NaN values into the objects transform and caused objects to disappear. It was an extremely illusive bug that was seemingly occuring out of nowhere, however as a team we eliminated it.

5.3.1 Stereo Algorithm (3D)

The algorithm for stereo visualisation of the game is actually very simple. It is partly inspired by the Unity Cluster Renderer technique and the network solution we had already implemented.
Updates which are sent to the observer occur over a network where there is no guarantee of when they will they will arrive, of course we expect that they will take far less than say, 1 second, but another level of execution locking is required to fully synchronize the state of two observers.

This is achieved through pairing position updates with a forecast time, and packet ID at which the observer should execute the actual position update. Initially, the forecast time should be well in advance, but rapidly decrease as the observers receive more updates. This is achieved by the observers sending an acknowledgement response to the server, detailing the **actual** time at which they received the update, as opposed to the forecast time. The server then uses these responses to build a model of the network latency, and adjust the forecast

time appropriately to maximize responsiveness in gameplay.

To fully illustrate the system, two side by side algorithms are given for the server and observer components.

Server Functions

```
def serverReceiveUpdate(u)
    id ⟵ createIdFromPacket(u)
    storePacketInDict(id, u, CurrentTime)
    time ⟵ CurrentTime + getMeanLatency()
    for all observers do
        sendForecastUpdate(u, id, time)
    end
end

def serverReceiveResponse(id, r, time)
    diff ⟵ time − originalTimeFromDict(r)
    updateMeanLatency(diff)
end
```

Observer Functions

```
def observerRecieveForecast(u, id, time)
    obesrverSendResponse(id, u, CurrentTime)
    waitUntill(time)
    executeUpdate(u)
end
```

Where all latency functions simply query a data structure that holds mean and variance latency over the network.

5.4 Interaction

When considering the method of interaction on the client side, many factors had to be taken into consideration. Firstly, we had to ensure that the game is easy to pick up. To achieve this, a virtual twin stick controller was implemented. The twin stick is the most widely used control method in modern games, so this means that people could intuitively understand how to control their avatar without a lengthy explanation. The left stick is responsible for the player movement and the right for the turret rotation and shooting mechanic.

Each stick of the user interface is responsible for controlling an axis. These axes are used to provide continuous input, resulting in smooth player navigation in the game space without any need to remove their hands of their device. The axes themselves are simply a normalized 2-dimensional vector representing direction in which the joystick is being dragged.

One very important aspect of the user controls is ensuring that they are always consistent regardless of rotation in the observer view, i.e that up always moves the player up and similarly for the other directions. We achieve this by sending occasional updates of the observer rotation to the clients so that they can move respective to the **up-vector** of the observer object. The positions of the players are then sent to the server after threshold distance has been exceeded, and following this the server relays the information to the observers.

Therefore up-vector of the observer camera is an important component of gameplay and is calculated as follows. Each player contains a "fake camera" which is carried round inside their objects. This fake camera also contains an up-vector, which was mainly pertinent in times when players were visualised on their own devices. The average up-vector of the players is then taken, and the observer camera rotation is interpolated with this new value.

5.5 Enjoyability

A major focus of ours when it came to this game was to enable people with varying experiences of gaming to easily engage with our game, be it they've barely played any or are regular gamers. In order to do this, we ensured from the offset that while the controls and basic functionality of the game was simple to pick up on, there would be greater depth for people who are more avid gamers. in terms of simplicity, we ensured that the controls were simple to use, going with a classic joystick setup as mentioned above. The basic aims of the game, kill the enemy team and collect resources to win, ensure a fast-pace with simple objectives to ensure everyone can have fun from the get go.
For the more avid gamers, or simply those with an eye for tactics, there are a number of strategic methods in order to gain an advantage over the opposition. Due to the way the camera works, players are locked within a region of the world, which allows for a team to try and stop the opposition from moving to a potentially more lucrative area by staying on the far edge of the visible map, holding the camera in place. Other methods such as having one of your team be a collection ship, with other members of the team acting as meatshields for it is another strategy we hope more avid gamers will quickly realise and apply in the game.
In general, the worlds each have their own special enviromental hazard which makes each map different in how one needs to approach it. For example, the City Planet has portals which teleport you around, allowing for a lot of potential in how one approaches resources and the enemy. Another planet has meteorites which strike the planet occasionally, damaging anyone who is nearby. All of these events help to bring a uniqueness to each round, ensuring the game does not become repetitive.

5.6 Screen views

5.6.1 Local view and Observer view

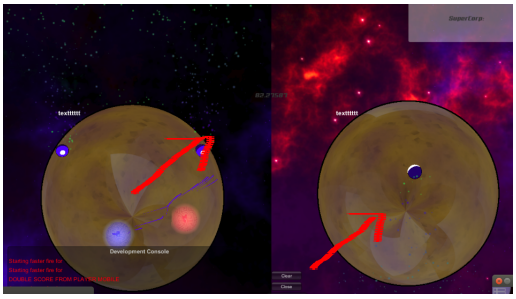
Our first implementation of the game involved having two separate views, a local one for each player which would be focused on the local avatar and a separate view for an observer, which would be ported on a large screen or the @Bristol Planetarium. However, during testing we encountered several issues with this implementation.

Firstly, we noticed that the player focused more on their own screen rather than on the larger battlefield view that we provided on the observer screen. In order to tackle this, we decided to remove the field of view of the player from the mobile phones and keep the data we display to a minimum. The only data visualised on the players device over the duration of a game is their unique ID with their selected team colour and twin stick controller. Additionally, in between round players are able to see their personal scores on the device and can observe their total team stats on the Observer monitor.

This change was very important to maximize the use of the Planetarium screen in particular, as there is a massive range of space to utilize, and to justify its use in general. A follow on point from this is to consider how we visualised the state of the game onto the large screen. Secondly, as the game took place in space on a planet, the dome view would only be able to see one side of the planet. Hence, if the players would move to the other side, they could not be visualised. It is of course important to ensure that all players are visualised on the planet, but also to allow players to explore parts of the world by themselves and not feel like the game is a closed grid.

Two main ideas were experimented with; a split screen version of the game which displayed both sides of the planet, and one single screen version. In the final product, we decided on the single screen view for ease of play.

5.6.2 Split screen



The split screen view consisted of two cameras that display one half of the planet each. The interesting aspect of this view was that the controls must be consistent when transitioning from one side of the planet to the other. So that for example, when a player moves off from the top right of the left view, they appear moving in the respective direction, from the bottom left of the right view. This meant that the right view image must be inverted in the horizontal axis. A result of this inversion means that while the players will move in a consistent direction across the views, their controls will now be reflected in the axis upon which they crossed into the other view.

Therefore, we introduce the notion of the joystick adjustment matrix J_{adj}, which is multiplied with the normalized joystick vector J to correct the direction in which a player should move. When a player crossed the boundary from one view to another, another reflection R is applied to the joystick adjustment matrix.

$$J = \begin{bmatrix} X & Y \end{bmatrix}^T \qquad R(\theta) = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} \qquad J_{adj} = \prod_{\theta \in crossings} R(\theta) \tag{1}$$

$$J_{new} = J_{adj} J$$

This method worked well for seeing the entire planet on one screen however through testing we found that the concept was difficult to grasp given that the transition occurs on a spherical shape. Therefore we required a more simple option.

5.6.3 Single Screen

In the single screen view all players are enforced to stay within the the view of one camera, which displays one half of the planet. This is achieved by constraining the player position to be within the distance between the camera and the centre of the planet. The camera is positioned at twice the radius of the planet, while always facing the centre of the planet. To determine where the camera should move, the average position of the players is calculated, and the camera object is rotated to face this point. This rotation is then synchronized occasionally with all players to ensure that they cannot move out from the field of view.

The single screen allows for a gameplay-flow effect where players of the opposing teams can push to explore different areas of the planet in search for resources and power ups. The only time where the view would be locked is when equal amounts of players are pushing against opposite ends of the camera view on the planet, which is unlikely given that the objective is to maintain and steal resources from the opposing teams.

5.7 Graphics and interface

An important design decision that had to be taken early in the project's development was the direction that graphics should take. There were two directions that the project could take in regards to that. Either take a more photo-realistic approach or a more cartoony one. In order to reach a decision all the affected factors were taken into consideration.

A primary concern was the fact that the game had to perform well on the Planetarium Dome, and for the result to be as smooth as possible the polygon count had to be kept at an optimal level. We were limited to around 4,000,000 polygons, which would be infeasible with a photo-realistic approach, considering the amount of players the we were aiming this for could cause serious rendering and synchronization issues.

The Planetarium also made clear that if the game was a success, they would be interested in taking it on as a a new showcase. Thus, the Planetariums standard demographic had to be taken into account, which includes young children. This meant cartoon-like graphics stood out as the preferred choice.

The design of our models were heavily influenced by the two factions in the game. The first two ship models were created before the two teams were decided. However, after we decided that pirates and Super Corp were to be our factions, we re-designed the two ship models. For the Super Corp we were aiming for a sleek, almost simplistic design, whereas the Pirates needed to inspire a more rough, gritty feeling. The two colour schemes utilised to distinguish between the teams, blue and red, were chosen for the same reason.

Deciding upon the two factions in our game also led to changes in planets. For example, as the city planet was aimed to be the home of the Super Corp, the planet was designed to inspire just that, by creating huge buildings. For the same reason we designed a sand planet, the home of the pirates, where we included elements such as meteor strikes and mountainous terrain.

The colour scheme for the U.I. was chosen whilst taking into account the two teams. We chose a neutral colour, turquoise, seeing as it was a good fit due to being suitably different from the team's colours as well as stand out and give a "spacey" feel to the interface.

5.8 Music

Additionally, the music was also tailored made to the projects needs. Inspired by our decision of the two factions, we wanted the music to relate with each factions ideals and purpose in the game. Working closely with the composers, we decided that the themes should be very strong and powerful, but still retaining pace, so as to immerse the player in a feeling of a great battle. All scores respect a certain time signature so that themes can be combined.

Similarly, each planet needed to feel distinct from the rest, that's why we required the theme for each planet to represent both its environment and purpose to the games story. The faction themes are overlaid onto the planet theme, so that every planet has a different feel, to enhance the story throughout the gameplay.

6 Project Planning

6.1 Initial Goal Planned

For organisational purposes, we set some milestones that needed to be accomplished by various points during the year so that work could proceed smoothly:

Getting to know Unity: No one in our team was particularly experienced with game development, and so we scheduled a twelve hour coding session in order to familiarize ourselves with Unity.

Git learning sessions: Members of our team were not that familiar with git. We set up a session that would enhance everyones ability to use this tool.

Basic single player implementation: After going through some tutorials related to our game concept we made our first step of our game creation. We implemented a simple single player demo where you can control an avatar which can move around on a plane, as well as shoot projectiles.

Research networking: The next step was to make our first attempt at turning the existing single player demo into a multiplayer one. We researched ports, along with IP address assignment.

Debugging: Debugging was mostly something that occurred consistently throughout the project, ensuring that the game was working properly at any stage. However, we also organized some dedicated debugging sessions as well, ensuring that the game was as functional as possible at any key stages. Often we would involve inexperienced users to try it out to help find any issues.

6.2 Gameplay Revisions

We continually added and adjusted the planets, as we felt that they needed to be as diverse and memorable as possible. We crafted backstory for the game's universe, as well as why each planet is important to the game's lore and universe.

Following this, we needed to consider the way players interact with each other and with the game environment. Our initial game concept was mostly focused on the technical and gameplay aspects. Our vision for the game has come a long way since then. In order to expand our game's scope we introduced resources, an in game item so important to the survival of each of the teams and their plans, that they are worth fighting and dying over.

Going beyond story and texturing, we decided to introduce the concept of Environmental Hazards. These are features that make each planet even more unique and add an extra layer of gameplay complexity. For example, one of our planets is surrounded by an asteroid field, however the catch is that some asteroids every now and then are detached from their belt and proceed to collide with the planet, damaging anything near the impact zone.

7 Team Process

7.1 Team Members and Roles

After finding out each member's strengths and weaknesses we divided the members into teams responsible for different aspects of the project. Graphics, Networking and Gameplay were the three main project pillars. Georgiana and Jamie were largely responsible for the design of textures and models, providing these to be integrated into the respective parts by the other teams. Likewise, if a task was related to the networking part of the project, then Callum and Thomas Camp would take focus on that and the rest of the team would focus on making sure the existing system would be able to adapt to the new network configurations, else make changes accordingly. The remaining members would focus on gameplay improvements and integrating new features to the game, along with making sure that no new changes and additions were causing problems to the existing implementation.

7.2 The Good Moments

The good moments include the first time that the networking component ran smoothly without any game-breaking bugs, and the first time we ported the game to the Planetarium correctly, being able to play a full game in that environment, as well as the first time we managed to port the game in 3D.

7.3 The Improvable Moments

As deadlines for different modules started piling up, various members of our team couldn't work on the project as much as they wanted. We experienced some periods when the team just couldn't focus on the game's development, either due to exams or to coursework deadlines coming up. At the first development stages, with the project deadline being some time away, there was no particular pressure on the team, and everything was proceeding at a more lax state. This might not have worked to our advantage as we could have made use of that time, to make even further progress, especially in regard to 3D. In hindsight, we would have looked deeper into the work pipeline, rather than trying to deal with what was in front of us at that moment.

7.4 Knowledge Gained Through Collaborative Teamwork

Communication in large scale projects is one of the most important columns that keep a team standing. We quickly realised that we needed to be constantly updating one another on project's development as most of our system's functionality is interconnected, hence a small change made in one aspect of the system could potentially break some functionality relying on it.

We made use of Trello, a site where we posted the next implementations scheduled for the project as well as who was responsible for each of them. That way we all made sure that knew what needed to be done, as well as whom we were to collaborate in case our jobs were either similar or related.

Another very important aspect of working in a team is that you should always keep an eye on your team members. Make sure that you pay attention on how they are doing. If they cannot accomplish their given task help them, pair up with them so that they can better work and not struggle needlessly. Also, in case they do not seem okay, show your team that you care more than just the project.

7.5 Given A Second Chance We Would Change ...

Start exploring each others strengths and weaknesses, so that we know how to better distribute the work around the team. Try to put more strict deadlines in regards to content so that we could have better understanding of how long it would take to develop some functionalities. Also try to communicate more often with other team

members, especially the ones assigned to high priority tasks, in order to make sure that everything is up to schedule. It also ensures that if difficulty arises it can be remedied quickly.

7.6 Methods Employed To Assist Collaborative Development

Slack was used, an application specifically designed for team communication, which allowed as to directly get a hold of anyone in the team, post something important as well as add alerts for whenever a new Trello card was added, or modified, and whenever a new meeting was scheduled.

We organised weekly meetings to discuss and work together, as well as monthly or bi-monthly 12 hour working session depending on the workload, where we would not only work together but also discuss what has worked and what hasnt in terms of organisation. All meetings were scheduled long in advance so everyone could organise their work on other units and were added onto the Google calendar so everyone would receive reminders the day before a meeting as well as on the day.

At the beginning of the meetings we would plan out the day as well as the workload until the next meeting and all tasks would be added onto Trello, a task management tool that allowed us to keep up to date with that needs to be done, what is in progress and what has already been accomplished, as well as assign team members for the tasks and set deadlines.

Additionally, as git was utilised, all members were notified whenever a new contribution was added to the project so they could code review. This way, everyone was up to date with the latest project updates even if working remotely.

During the holidays, as most members were away, we had meetings utilising Google Hangouts to discuss progress and plan out the next development period.

7.7 Contributions

7.7.1 Georgiana Ginghina - Project Manager

- As a team manager I dealt with group communication and organisation:
  - Made sure there were regular meeting and work sessions where everyone would catch up and work together.
  - Communication with any external factions and make sure everyone is informed.
- I worked on graphics and UI :
  - Made the two initial ship models, as well as the final pirate ship model once we set a theme for the game
  - Made models for resources, as well as various planet enhancements, such as palm trees for the sand planet, a basic low-poly tree for the maze planet, buildings for the city planet.
  - Made the aesthetics for the final version of the UI, involving menu, animations and button responsiveness and timer visualisation.
- I worked on game play and networking communication:
  - I created a team manager that would keep stats for each team such as scores, players etc.
  - I created a kill feed.
  - Wrote the round manager with Elle which deals with changing the rounds when a timer would run out as well as keeping and resetting round data.
  - Made the communication between the team manager and round manager employed on the server and the clients.
- I worked on the observer view onto the planetarium:
  - After experimenting with different views onto the planetarium, I built the final view which would work as an arena, players only being able to access one half of the planet and also being able to move the view/ the arena towards them.
- I worked on identification for players:
  - Made players be assigned a unique id, visualised in a tear-drop pin style onto the observer screen.
  - Made players be able to tap on their id-pin on their mobile devices and that would trigger an animation of their pin enlarging and shrinking on the observer view.
- Added the original basic music system.
- Added the portals system onto the city planet.

7.7.2 Callum Mann - Lead Programmer

- As Lead Programmer I put effort into making sure all members of the team had tasks and that everyone was familiar with how their parts will integrate into the projects:
  - Providing project structure and guidance, the importance of good APIs etc.
  - Supporting other team members with their code and source control processes
  - Reviewing code on GitHub thoroughly
- I made large contributions to the networking component including:
  - Simulating the planetarium network and bridging wired to wireless
  - Setting up and maintaining the access point
  - Setting up the dedicated server
  - Fixing coma-inducing bugs relating to the synchronization of player objects
- I worked on the player experience:
  - Ensure that controls correctly map from user device to observer screen, i.e so that up is always up regardless of screen rotations
  - Tweaked the joysticks so that they are more usable on different devices
  - Experimented with split screen view and single screen view
- I integrated the 5 Camera system into the project so that objects in the game world are correctly drawn inside the Planetarium
- I worked on some UI screens related to the Observer
- I modelled some parts of a Planet in Unity.

7.7.3 Tom Michiels

- Was responsible for implementing a user friendly way for the user to interact with the game. Came up with a twin stick model, one stick for controlling your spaceship and the other for controlling the spaceships turret and fire.
- Worked on Player and Lobby U.I.
- Worked on the player movement and old camera system

7.7.4 Thomas Camp

- My first task was to establish a basis for the networked game which could be easily used by the rest of the team. The initial creation made use of many of unitys pre-existing network structures, which ensured for an easy to use basis for testing the game at an early stage. This largely consisted of making an operating server-client layout, allowing for the synchronisation of movement, rotation and scores.
- The game required more complex networking changes as we progressed however, such as having to have all interaction checks done on observers rather than on the respective local clients for each player. This required a lot of the code to be done using only single message passes due to Unity lacking any inherent capabilities to deal with such a situation.
- In addition we wished to make the game able to run with as many players as possible, which required us to try and reduce the network traffic as much as possible. This resulted in a re-work of the network system we had in place to only include the specific messages we required for our game, taking a step away from the pre-existing Unity networking solution. To ensure the changes were easily used we also sorted everything into easily-used managers for the various networked portions of the game, allowing for the members of our team less focused on the networking aspects to still be able to use it with ease.
- Beyond the networking aspects I also worked on the backend structures for scores, players and rounds, also working on ensuring these structures are correctly populated and used throughout the game.
- I also looked into 3d solutions along with Callum, testing Cluster Rendering as a solution for a period until unfortunately coming to the conclusion it was not fit for our needs of a networked synchronised game. We looked into other solutions to the issue of synchronisation for 3d, and eventually settled on an algorithm to be implemented. Unfortunately this did not get implemented in the final release.

7.7.5 James O Connor

- Designed the Super Corp ship model on Autodesk Maya.
- Made the sand planet texture, as well as the power-up pickup models.
- Made the skybox.
- Wrote some of the code for projectile firing, resource networking, respawning after death, visualizing the resources and scores, as well as the players gaining points for picking up resources.

7.7.6 Eleftherios Christodoulou

- Feedback on the development of the game and helped evolve its gameplay.
- Helped create the initial Player Controller, which included how the player behaves inside the game environment as well as how it shoots.
- Created a script, responsible for managing the different resources around the planet. Namely it controlled the behaviour of power ups, resource pick ups as well as planet hazards, like meteors. The script is responsible for keeping the planets populated by resources and power ups at any given time during the round. It keeps track of their current numbers and keep them in between the desired levels. The script is also responsible for spawning the players as soon as the game starts.
- Play tested the game in order to ensure the best experience for the player. This included, regulating the number of power ups and resources around the planets as well as the power up lifespan. Meaning the amount of time before a power up stops functioning as well as the rate that a resource increases in value. Made sure that no round would ever lead to a draw as the system takes into consideration variables like team score, kills and deaths.
- Created a manager responsible for managing the music scores for each part of the game. Assigned the corresponding audio clip to each round, menu and team of our game. The script provides the user with basic controls over the clips. The user can freely play and stop the different audio clips of an instance, as well as, regulate their individual volume.
- Wrote different scripts and functions for providing the User Interface with useful information regarding the game. Created a script for visualising the results of every round as well as the final results for the game.
- Worked on resolving a series of issues that our game had. Fixed a bug where in case the mobile device was not held horizontally when initializing the game it would immediately minimize. Resolved an issue, where when a player got destroyed, the observer was also destroyed. Resolved by adding a check to see whether the collision was local or not.

8 Appendix A

BackStory - (Clean up this)

Our game is set in a universe where Earth is no more. Years of fighting have finally rendered the planet inhospitable so the humankind had to abandon its birthplace and look for other planets to call home. And so, the last survivors of the human race, boarder large spaceships and began an age long trip into the nothingness of space in search for new planets.

While cruising the vastness of the Milky Way a wormhole sucked the remnants of humanity into a new galaxy, which the named Elpis, in hope that there lied their hope for the future. But as always, humanity could not stay united for much longer, so the populace was split into factions. We have Super Corp, the large organization responsible for the construction of the spaceships that transported the rest of human race off Earth. They declared that since they provided the means for the humans to escape, the human kind should forever praise them as Gods and do as they command. This unexpected and forced employment contract caused an uproar amongst the people and soon after this declaration some brave space pilots took it upon themselves to oppose the might of the Corporation.

Being originally pilots under contract of the Corporation they announced their resignment and took off with their spaceships to form a colony in the far reaches of the new galaxy, away from the clutches of the Corporation. Their goal was to become a thorn to the Corporation, which soon branded them as terrorists and Pirates, and they were going to accomplish that by stealing their valuable resources which they put people to work like slaves to extract them and make people see the true face of Super Corp. Their idea is that more and more people would join their cause and one day Super Corp might find its end.

That was 20 years ago, the Pirates have more than enough power to challenge Super Corp, but have gotten drunk with power and limitless freedom. They became corrupted by their unlimited freedom in the galaxy. The lines between good and bad, light and darkness, are fading away and all that remains its just War.

9 Top 10 Contributions

i We have a full stack solution for a networked game including access point, dedicated server and distributable package. We have managed to host around 10 players on mobile devices.

ii The game design process went through several experimental phases which focused around how to use the large screen effectively. In the end we have settled on a gameplay where all players look at one screen.

iii We integrated the game into the Planetarium which may open possibilities for further collaboration and enhance interactivity inside the dome environment in the future.

iv Further to the previous point, we have detailed in our report the modifications required to visualise the game in 3D, which we were not able to integrate due to lack of time.

v Created our own planet textures as well as space ship models using Autodesk Maya. We created 5 sample planets but for the final game had to cut down the number to 3. The spaceship models were quite different at first but they were refined later.

vi The entire game is written in C# using Unity engine, in total around 8000 lines of code written refined down to 5000.

vii Collaborated closely with composers to develop music that enhanced the atmosphere of the Planetarius universe.

viii In the last month we operated in a smaller team force due to two members having health issues.

Table 1: Assessment Page

Name	Weighting	Signature
Georgiana Ginghina	1.08	
Callum Mann	1.16	
Thomas Camp	1.08	
James O' Connor	0.8	
Tom Michiels	0.8	
Eleftherios Christodoulou	1.08	