**Group Submission – Front Sheet**

**Level 2 Group Project**

Component Number: 3

Programme Title: Games Computing

Group: GC3

Supervisor: Andy Cowe

The following table shows the agreed allocation of workload against this assignment submission. The table has been checked and all members in our group as identified in the Group List presented on Blackboard, are shown correctly in this table. The percentage values provided have been agreed and total 100%.

|  |  |  |  |
| --- | --- | --- | --- |
| Enrolment  Number | Surname | Forename | Agreed Percentage Workload |
| 10237560 | Bowes | Adam |  |
| 12266105 | Fisher | Thomas |  |
| 11207513 | Pearson | Darryl |  |
| 12281943 | Graves | Joshua |  |

Contents

[Introduction 3](#_Toc386640126)

[Design & Requirements 3](#_Toc386640127)

[Initial Design 3](#_Toc386640128)

[World 3](#_Toc386640129)

[Objects 3](#_Toc386640130)

[Goals 4](#_Toc386640131)

[Tools and research 4](#_Toc386640132)

[Unity 4](#_Toc386640133)

[Havok 4](#_Toc386640134)

[Microsoft Project 4](#_Toc386640135)

[GitHub 4](#_Toc386640136)

[Facebook 4](#_Toc386640137)

[Skype 5](#_Toc386640138)

[Qubicle 5](#_Toc386640139)

[Maya 5](#_Toc386640140)

[Magix 5](#_Toc386640141)

[Project planning and management 5](#_Toc386640142)

[Artefact Implementation 5](#_Toc386640143)

[Testing Strategy 6](#_Toc386640144)

[GUI - Graphical User Interface 7](#_Toc386640145)

[Building Placement 7](#_Toc386640146)

[Building Deletion 7](#_Toc386640147)

[NPC’s - Non Playable Characters 7](#_Toc386640148)

[Water affects 7](#_Toc386640149)

[Changing colour based on pollution 8](#_Toc386640150)

[Ripple affects 8](#_Toc386640151)

# Introduction

For this assignment the team needed to produce an artefact which would be a serious game with the intent of informing the player of the challenges facing the conservation of marine environments. The game needed to have a focus on action whilst remaining casual and appealing to a wide audience. Also the game could be built on PC but needed to be suitable for a direct port on to a mobile platform. The team chose to make a game that put the player in charge of managing a beach with the intention of showing them how litter left on beaches can affect marine life. This was done with the intention of showing the player the way they impact marine life from a first-hand perspective as most people at some point have been to a beach. With this game the team intended to put the player in a management role to see how individual buildings on a beach can contribute to pollution. As well as this, the team wanted to show how people visiting the beach caused pollution and what measures could be taken to reduce the effect.

# Design & Requirements

## Initial Design

The initial design involved putting across just how large an effect beach based businesses have on marine life and the ocean. The player would be tasked with looking after a beach, keeping it both profitable and healthy, with health being the most important factor. The more businesses the player owns, the more trash will be produced, making it more and more difficult to look after the beach. The player is then faced with the choice, will they build large businesses but struggle with the environment, or will they stay small but win in the long run?

## World

The game world needs to be a bounded beach area on which the player can begin to develop in to a popular destination. The beach will be affected by litter which will be created by visitors to the beach and the beach will graphically deteriorate to reflect this, Also the NPC’s will react to the litter by leaving the beach.

One of the changes made from the original concept was the piers. The team chose to remove the piers on the grounds that they added little to gameplay and did nothing to enhance the message.

## Objects

The game has a wide range of objects that allow the beach to take shape. The buildings that are available are as follows: lifeguard stations which increase the number of people allowed on the beach, fisheries which create income based on how clean the beach is but generate waste in the process, hotels which increase the number of people who come to the beach and ice-cream shops, gift shops and clubs which NPC’s visit and gave money but also cause the NPC’s to have litter which they may drop.

Some of the changes made involve the removal of some aesthetic items. These were removed due to the way the map is created. As the beach can be made any size the team would need to develop way to seed the map with these objects which would be time consuming

## Goals

The goals of the game are to make the beach a popular place to visit by building different structure to attract and then meet the needs of the NPC’s, at the same time the player will need to manage the pollution generated by the visitors which decreases the popularity of the beach. The player needs to manage the beach in a sustainable way to ensure the both the environment and the visitors are happy despite them being in conflict.

# Tools and research

## Unity

For this project the team chose to use unity to develop the game. Unity has gained a large amount of popularity and has also has a large supporting community which made problem solving and dealing with bugs very easy in most case. Unity has proven to be a very versatile tool for making games, the built in PhysX allowed for quick and optimised responses to collisions and the visual scene editor makes debugging and scene building very simple. The ability to edit code as the game is running is very useful also as it means new features can be tested quickly without waiting.

## Havok

We also chose to consider using Havok as a tool to develop the game, after implementing a few very basic features (such as terrain and lighting) we realised that in order to do our project in Havok it would be near impossible. Havok is a good tool to use if you already have knowledge of how to use it, once we found out how to do simple things like the terrain they became incredibly easy but before then all we had was a flat surface.

The Havok community is still very small when compared to the Unity community so this would mean that online tutorials and examples are very limited and sometimes not informative. Coding in Havok is allot more complicated with very little support in unity you know how to find the coding options but in havoc they seem to be hidden away from the user.

## Microsoft Project

When we started this project we used Microsoft Project as a way to manage our schedule and our workload, for each task we had set deadlines and milestones. Because every member of this group has other projects to complete alongside making this game we started by only working on this project one day away so we could complete all projects in the set time.

Microsoft Project also allowed us to allocate tasks to individuals for example we allocated josh to design the NPCs

## GitHub

The team chose GitHub to compile the groups work. This was done so that changes could be kept track of and so that everyone would be up to date when working on the project. GitHub has been very useful allowing changes to be viewed quickly by other team members. It also prevented anyone working on an old version of the game which would have meant that their code might not work with the latest version.

GitHub has a few problems which come from working with Unity. Due to the fact that a team member would need to run Unity in order to see any changes made to code, Unity would often change its metadata and mono would update the project files which means when merging an old version of the project with a new one some conflicts emerge which can sometimes cause problems

## Facebook

We decided to choose Facebook as our main communication tool, this is because we all had a Facebook account and used it regularly. With Facebook we created our own private group so that only the group members would be able to access it, this allowed us to easily communicate with each other on a daily basis. We also used Facebook to post updates on the project so that members of the team knew what was completed and what still needed to be done.

The main drawback of us using Facebook as a communication tool was that it was also distracting at times due to it being a social media site, this meant that sometimes instead of doing project work members would talk to friends and family.

## Skype

We used Skype to allow us to hold online meetings when we were not able to physically meet up at a team. This allowed us to discuss any issues which had arisen as well as setting tasks and deadlines.

The only real drawbacks of us using Skype were that it requires the user to have an internet connection, microphone and speakers. Also at times people could speak over one another accidentally as we were not able to see each other.

## Qubicle

Every model present in the game was made in Qubicle, a 3D modelling program that is used solely to make voxel based models. This program was ideal since the models did not need to be complex and it gave the models a theme / look. The only issue we had from Qubicle is its inability to export models in a usable format, meaning we had to use external software to make the models usable. The first solution was QubicleUnite, which fused all the voxels in a model, lowering the number of vertices and creating a texture overlay. This would then need to be converted to a usable file.

## Maya

We did not use Maya for its usual modelling capabilities. We instead used it solely as a converter that made our Qubicle files usable. It could import a Qubicle file and output it again as an .obj file.

## Magix

Magix was chosen as our music making program since it was quick and simple to use. Since the group lacked musicians or people familiar with music production, it was important for us to find an easy to use program. The primary drawback to the program being so simple, however, was that it also lacked complexity. Complex and interesting songs would be almost impossible to make, since the program wouldn’t allow it.

# Project planning and management

Due to a large workload from other projects, it was necessary to have a dynamic and stable way to communicate with team. To this end the team used a Facebook group. The group allowed us to have a space where members could report on their progress and tasks could be given. As a result there was a large amount of communication as the team members asked each other for assets and code to be finished. Facebook also has a time stamp on its posts meaning when things were said and assigned can be reviewed. Using Facebook however had a few disadvantages, due to the nature that the tasks were assigned as they became apparent the critical path was not well tracked and instances occurred when team members had to wait for things to be finished by others. Another way the project was managed was with GitHub. GitHub stores the time and amount contributed of each commit to the project. This meant that progress of team members could be tracked and the tasks completed could be monitored.

# Artefact Implementation

//what we did plus challenges

## Unity

One of the first challenges faced was creating the map. The map needed to have changeable rows and columns while also allowing buildings to be built on individual tiles. To do this the tiles were created in a nested loop which created and moved the tiles depending on the rows, columns and the tile size. By having this array, the tile objects can be accessed and by having the rows, columns and tile size as public they can be accessed from other scripts.

Another challenge was creating a set of NPC’s that could navigate a dynamic and changing world while still being simple enough to run on a phone. To tackle this the NPC’s were given enlarged capsule colliders and the buildings were also given capsule colliders so that when an NPC walked in to a building the colliders would prevent them from walking through a building and would instead get pushed around meaning that the NPC’s give the impression that they walk around each other and obstacles without putting a large load on the CPU.

The NPCs were a main part of the gameplay and needed to go to shops to generate money as well as react to the environment so players could see the effect it had. To do this the NPCs were given a set of needs, these needs had a random number assigned from a range representative to the need in question (ice-cream took between 10 and 100 seconds while buying a gift took between 60 and 100 seconds), this meant that the price shops sold at might not represent how much money is made adding an extra layer of complexity. Amongst the NPC needs there was also a need to get rid of litter acquired from shops. If this need was not filled by a bin in time then they would drop the litter. To enhance the focus on the effects of pollution, the NPCs also have a tolerance for litter. This means that if a beach is too littered then they will make a verbal comment on the state of the beach and then leave, this brings the issue to the players’ attention whilst having a negative impact on the income from the beach.

Building placement had to be implemented which meant that a message had to be sent to a building control object from the GUI control object telling it what building to make. As the buildings had a type and a size the data had to be passed by an array as unity does not allow multiple parameters to be sent to a function using the “sendmessage” function. As well as this, a ray had to be passed from the camera and through anything that wasn’t a tile. This was achieved by using tags which allowed the ray to identify different objects and so only react to tiles. This also allowed the buildings to be shown on the tiles before they were actually placed.

## Havok

The project we created in Havok is very basic the biggest challenges faced was to find the options in order to perform options, for example the properties and the scripting editor occupy the same window and you have to go and enable the scripting window from a drop down menu at the top of the screen.

One success of the Havok project is the terrain, using Havoks terrain editor we was successful able to put the beach on a slow slope into the sea, so the waves will role all the way up to the sand. Not one bit of the beach is the same size as the bit next to it which is very realistic and it also allowed us to create sand dunes this also adds to the realism of the game.

One of the hardest things to do in Havok is a GUI not only is it hard to find the scripting window, there is very limited support from the program to help you code a GUI, the help window in Havok provided a very good example on how to program a GUI but after programming it the compiler wouldn’t go past the first line.

The camera feature is not perfected in the game, it was made by using a simple camera faced at the middle of the beach, a path for the camera was then made so the camera can only go horizontally to prevent part of the game disappearing because Havok only displays a certain distance from the camera and this is not what we want for our game. There is no property on the camera option to make the camera stick to this path and there are no instructions on how to make these work together.

# Testing Strategy

We used a number of different testing strategy’s while we was creating the project, our main testing strategy was unit testing this is because when we a member of the team had finished their task they would then commit there changes to GitHub and we needed to ensure that, that version of the project would always run without any errors. Due to the nature of how GitHub works, it is best to only have one user making changes to a single file at a time, this tied in well with our testing strategy, after each task had been finish we tested the new code to ensure that the game would run correctly without any errors before uploading it to GitHub.

We also used Unity’s debugging tool often, this allowed us to display a message to the Unity console when a certain task was performed, for example when testing the building placement we used the debugging tool to tell us what button we pressed, what building was selected and where on the grid it was placed.

When creating different parts of the game we had to test them slightly differently. These are the ways that we tested them:

## GUI - Graphical User Interface

When creating the GUI elements we ran the game after each new element was added to ensure that the element was the correct size and placed correctly.

After the graphical user interface had been fully created we then tested it on different size screens and also on mobile devices, we then made changes to the interface to ensure that it would be a useable size on the different screen sizes.

## Building Placement

When testing the building placement code we had to test for a number of different things like if every building could be placed. During this process we noticed a number of bugs, the user was able to build multiple buildings on the same spot and some of the buildings overlapped. We fixed these bugs by changing the size of the building models and then adding a function which checks if there was already a building placed in that position.

## Building Deletion

We tested the building deletion by placing one of every building on the map and then deleting them.

The first time that we did this we were only able to delete the last building we had created, this was due to a wrong variable being used which held the last building placement instead of where the user had clicked.

The second time that we carried out this test most of the building could be deleted apart from two categories, the fisheries and the clubs, we realised that this was due to the way the prefabs for these models had been made was slightly different from the others, the collider was a child of the model for the fisheries and the clubs where as for the other buildings it was a child of the prefab, after moving the collider we then retested on the affected building. After doing so all the buildings would delete correctly.

## NPC’s - Non Playable Characters

We tested the NPC’s by running the game and adding the different building they interact with. We then looked at how often they spawned to check that it was between 10 and 100 seconds. We also looked at how often they dropped litter to ensure that they were not dropping it too often and that they were dropping it. We looked at how they interacted with the buildings and each other, to ensure that the colliders worked and they couldn’t walk through each other or the buildings. When the NPC’s walked up to a building we checked that the user gained the correct amount of coins.

## Water affects

There was two water affects which we needed to test, these were:

### Changing colour based on pollution

The colour of the water was meant to change from blue to green based on the pollution level, we tested this by adding waste to the game to increase the pollution level to see if the water changed to its correct colour.

### Ripple affects

The water ripple effects were tested by observing the water to ensure that the water rippled in the correct direction to mimic an ocean sea where the sea level rises up and down the beach front.

# Testing Requirements

//We got people to play the game, now they magically hate trash on beaches.