Graphics Project 2 Report

Brief explanation of the game

Island Survivor is a game where the player spawns on an island dotted with trees, coconuts and rocks with nothing but their wits to save them. During the day, the player can eat coconuts to sustain their energy levels and collect rocks to help fend off the gorillas which attack at night. The aim of the game is to survive as many nights as possible.

How to use it (especially the user interface aspects)

The game is controlled using user input from the keyboard and mouse. The arrow keys and WASD keys control the player's movement and the mouse controls the player's rotation and also shooting. As the mouse moves, movement in the X axis of the mouse causes the player to rotate around it's y axis to the corresponding direction of the mouse movement. When the user clicks, if the player has rocks, a rock is thrown in the direction that the player is facing.

How you modelled objects and entities

There were a few distinct objects/entities we modeled.
Lighting :

- All of our light sources were orbiting bodies. We attached a script to each light source, which used basic trigonometry to have the object move in a circular fashion, based on a given speed, radius and starting time.

Characters:

- The player is seen as a capsule, with an attached Unity store model. It also has a script attached which allows it to shoot projectiles. The inspiration for this was sourced from the Tutorial 7 solutions
- The gorillas are seen in a similar way, being capsules and having Unity store models, except they cant shoot but instead have a script which moves them towards the player smoothly.

Island:

- To create the island, we roughly modeled a circle, with a variable precision (how many points will lie on the edge of the circle). We then added a random number to each point to make it appear somewhat rough.

How you handled graphics and camera motion

Graphics was handled by the standard unity graphics pipeline. Camera motion was executed by making the camera a child of the player, hence, when the player moved, the camera followed in a fixed third person view.

Descriptions of how the shaders work

Two separate shaders we're used in this game.

1. One of the shaders is a basic Gouraud shader from the tutorials and Assignment 1. It uses Gouraud shading techniques on the island itself to light it. The shader works by averaging the normals of the vertices of each triangle and subsequently linearly interpolating the colour for each pixel across the face of the triangle, based on the colour of the vertices comprising the triangle. as there are multiple light sources, the colour and intensity of all the sources is averaged and the direction of the primary light source is chosen as the overall light direction, whether that is the sun or the moon.
2. The second of the shaders is used for the water effects. In this shader, a main texture and a distortion texture are used. The main texture is a simple image of water from the unity asset store. The distortion texture is an image of a greyscale "cloud". Details for this texture are provided in the commenting of the "WaterShader.shader" file. In summary however, the main texture UVs are distorted using the cloudy distortion texture.

Description of the querying and observational methods used

Querying Technique

<u>Description of the participants (how many, demographics)</u>
For the querying technique, we had 5 participants. A table of the demographics of the participants is shown below.

Participant Number	Gender	Age	Preferred Language	Country of Residence	Occupation
1	Female	20	English	Australia	Student
2	Male	23	English	Australia	Student
3	Female	19	English	Australia	Student
4	Female	20	English	Australia	Student
5	Male	19	English	Australia	Student

<u>Description of the methodology (which techniques did you use, what did you have participants do, how did you record the data)</u>
For the querying technique, we used a questionnaire format. As we wanted to ensure our questionnaire was reliable, we decided to use the the System Usability Scale as it is widely used for measuring usability. We asked participants to score each of the ten questions listed below using the following five answers: (1) Strongly disagree, (2) Disagree, (3) Neutral, (4) Agree, (5) Strongly Agree.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

We recorded data in the form of a printed questionnaire where participants could circle their response to each question.

Feedback gathered

The results of the questionnaire are shown below. Note that answers are shown with their corresponding number rather than the actual description listed in the questionnaire.

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	4	2	4	1	4	1	5	1	5	2
2	5	1	5	1	5	1	4	1	4	1
3	4	2	4	1	5	2	3	2	5	2
4	4	1	5	1	4	2	4	1	4	2
5	4	1	4	1	4	1	3	1	4	1

Observational technique

Participant Number	Gender	Age	Preferred Language	Country of Residence	Occupation
1	Male	20	English	Australia	Student
2	Male	21	English	Australia	Student
3	Female	20	English	Australia	Student
4	Male	19	English	Australia	Student
5	Female	22	English	Australia	Student

We chose the Cooperative evaluation method as our Observational Evaluation method. The reasons for this decision were as follows. We wanted to create an organic environment in which we could have a back and forth with the user. For us, discovering the user's exact response in the moment was important in the early stages of the game. We didn't want to impose our ideals onto the users too much in these stages, as we quickly realized that being the ones to create the game drastically shifts your perspective when playing it.

What we learnt and changed:

Originally, we tied the rotation of the character to the left and right arrow keys. This created an issue we quickly discovered within the participants. At the start, we had the rotation as fairly slow, but upon talking to the players, they found that it took too long to turn around to attack gorillas behind the player. After that, we increased the speed of the rotation, but it became evident that the players felt it was too hard to aim when the controls were so sensitive. As such, we connected the rotation of the player to the mouse. This makes the rotation more dynamic and makes aiming easier.

Creating somewhat realistic lighting was an aspect we took a lot of direction on too. Unity natively has an ambient light source, but it was too bright to create a realistic night, as evidenced by the players' remarks. Consequently, we toned down the ambient light. This created a new issue, where at dawn and dusk, then both the sun and moon were on opposite ends of the horizon, the plane was essentially black. To combat this we created a couple of smaller light sources to orbit slightly before and after the sun to make sure that at dawn and dusk there was still some amount of light shining. After a suggestion from one of the players, we also tinted these light sources orange to simulate a realistic dawn and dusk.

Another facet of the game that the players commented on was the size and shape of the island. The first version of our game had a square island which was fairly small. The players, upon asking what they thought of our decision to have it as a square, responded that it didn't feel organic and was too small. As such, we decided to change it to a circle. The players felt that this was a bit too regular though, as a couple of our test subjects mentioned it felt a bit artificial. As such, we wrote a basic algorithm to create a circle and give it some roughness.

Document the changes made to your game based on the information collected during the evaluation

Player movement

- We initially had the rotate speed too slow, users we tested the game said they would like it to be faster
- Also, the space bar was originally used for shooting however we changed this based on feedback so that rotation is based on mouse movements and shooting is done by clicking the mouse button
- Changed the size and shape of the island from a small square to a larger rough circle

Pickup objects

- Made pickup objects float and rotate to make it evident they can be picked up

Lighting

- Added day and night
- Also added small light sources to simulate dusk and dawn (explained above)

Water

- Dynamic movement of water

Enemies/Player Death

- Blood splatter/explosion upon death of enemies or health loss of player

A statement about any code/APIs you have sourced/used from the internet that is not your own.

The player, gorilla, rock, bullet, coconut and tree assets were sourced from the Unity asset store. These only visual aspects of these prefabs were used other than the slight animation of the player. Also, the texture for the water was found on the Unity asset store and the cloud texture for the warping in the water shader was created using Photoshop. Tutorial 7 solutions also provided some good skeleton code for handling health, shooting, and particle effects.