Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name:
Student ID:
Supervisor:

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:_____
Date:_____

National College of Ireland
BSc in Computing
2016/2017

Sean Trant
x13332576
seantrant@gmail.com

Technical Report

National College of Ireland

Table of Contents

Executive Summary

Maximum 300 words. The abstract should mention the problem being addressed, describe the technical solution and briefly report the findings of the evaluation.

This document will serve as a technical report for the project “Cargo Cruiser”. The main objective of this project is to re-introduce racing through new ideas. Players are tasked with finding cargo and bringing it to their getaway vehicle after the caravan convoy has crashed. Next, they must escape by transporting cargo to the finish line as fast as possible while being pursued by the hungry gorilla.

It is currently being developed using the engine Unity, the modelling program Maya and the coding platform C#. Additionally, the project is backed up by a live website made with HTML, CSS and PHP.

From analysis, I gather that the project has very achievable goals based on what is stated in the requirements (this is also based on what has been achieved thus far) and has room to grow. The system has been evaluated through integration testing, test cases and end-user (customer) testing. The report does evaluate and conclude that this project is an ideal candidate that meets its presented challenge. Discussed below is the concept in great detail, the overall objectives of the project, the user requirements, among others.

1 Introduction

The scope of this project is to develop a 3D racing game made using unity and C# that is aimed at pushing the boundaries of the current racing genre by adding new gameplay elements and twists.

The primary objective in this game will be to race against the clock, the gorilla, and other cars while keeping as much cargo inside the vehicle as possible, this will contribute to a player's score, which is not just reliant on who finishes first, but also who has the most cargo remaining. There will be an AI controlled gorilla that will pursue players through the race and attempt to eliminate them from it.

Game Description	
Genre:	Racing
Game Elements:	• Collecting cargo • Driving • Throwing cargo • Escaping • Competing

Game Technical	
Technical From:	3D Graphics
View:	Third person, first person, rear view
Platform:	Unity, C#
Device:	PC

1.1 Background

To surmise why I decided to undertake the specified project, I have a personal interest in seeing whether or not innovative twists moulded into an otherwise ordinary racing game will make for a more entertaining title. I find myself bored by many of the racing games I play myself, and while I understand this is down to personal preference, the curiosity remains.

The idea stemmed from a simple project from 3rd year, a racing game in which a player would lose points when cargo fell out of the vehicle, this was physics based, which proved to be very unreliable. Upon completion of the project, I felt unsatisfied, like my team and I had merely scratched the surface of a good idea. So, from here I expanded and developed it to the point where it was entirely upgraded.

The highly popular mobile game Temple Run showed that the idea of being chased could add a fresh improvement to the standard racing game. I then found this thought reinforced by a multiplayer game called Speed Runners, in which 4 players race each other in a 2D setting continuously until the camera closes off behind them and begins eliminating players from the race.

In the initial idea proposal, I was advised to add more original elements to the game. This made me think of a setup phase, which could allow players to alter the race track and select different pieces of cargo to load onto their vehicle, which would affect its weight and speed, but also the value at the end.

1.2 Aims

The following are critical goals of the project:

- Race against other AI controlled racing cars
- Vehicles loaded with cargo which players must keep intact to achieve score.
- A pre-race level in which players have to find cargo to transport
- The ability to remove cargo to gain speed and safety.
- An enemy player/AI which will chase the racers and attempt to eliminate them.
- Online leader boards for players to share their scores and compete

These are discussed in greater detail in section 2.1.4 User Requirements.

1.3 Technologies

Unity – This is a game engine. It will be used to design many aspects of the game and ultimately build and run it.

Maya – This is a modelling program. It will be used to design any complex objects and characters. Maya models can be easily exported for use in unity.

Visual Studio and C# - This is a coding platform that is an accessory to unity. It will be used to program all functionality of the game through Unity's C# library.

PHP – This is a language to interact with the backend MySQL database through HTML and C#.

MySQL – This is a Language for writing database code.

HTML- This is a language to writing code to display data on a webpage.

1.4 Structure

Brief overview of each chapter

2 System

2.1 Requirements

2.1.1 Functional requirements

- The game must run when launched.
- A player must be able to select a level.
- A player must be able to load their vehicle with cargo.
- A player's must gain speed when they throw cargo from their vehicle.
- A player must be able to control a vehicle movement with the "wsad" or arrow keys.
- A player must be able to remove cargo from their vehicle with the press of a key.
- The system must detect and record the distance between a chaser and a player.
- The system must be present the speed of the player's vehicle.
- A computer controlled racer should be able to follow a given path of nodes (game objects).
- A player's score should be negatively impacted every time cargo is removed from the vehicle.
- Computer controlled chasers should be able to eliminate(attack) players from the race when they are within a certain distance.
- There should be sound to indicate how close a chaser is to a player.
- The player should be able to upload their score to the leader boards at the end of the level.
- The vehicle and gorilla must be disabled when the player crosses the finish line
- The time taken, cargo left and score must be displayed on the screen when the player crosses the finish line.
- The leader boards should be visible on the main menu with the click of a button.
- The leader boards should be visible on the website.
- The player health and cargo status should both be displayed as health bars on the screen.

2.1.2 Non-Functional requirements

- The game should be implemented and deployed using unity.
- There should be a screen where the basic concept of the game is explained.
- Functionality should be coded using C#.
- The minimum frame rate must be greater than 20.
- The average frame rate must be around 30, but ideally should be 60+.
- The game must run on windows 7 and windows 10.
- Response time for user input must be less than 0.2 seconds.
- The game's menu should be easy to navigate.
- All GUIs should be aesthetically pleasing.
- The website and game should load the leader boards quickly, in no more than 1 second.

2.1.3 Data requirements

It is unknown exactly what kind of hardware the game will need to run effectively. However, it should not be very taxing as generally Unity games run very well even on less impressive systems. The standard for running Unity games on desktop (the platform for this project) is:

- OS: Windows XP SP2+, Mac OS X 10.8+, Ubuntu 12.04+, SteamOS+.
- Graphics card: DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities.
- CPU: SSE2 instruction set support.

"Unity - Unity - System Requirements". Unity. N.p., 2016. Web. 11 Dec. 2016.

The website

2.1.4 User requirements

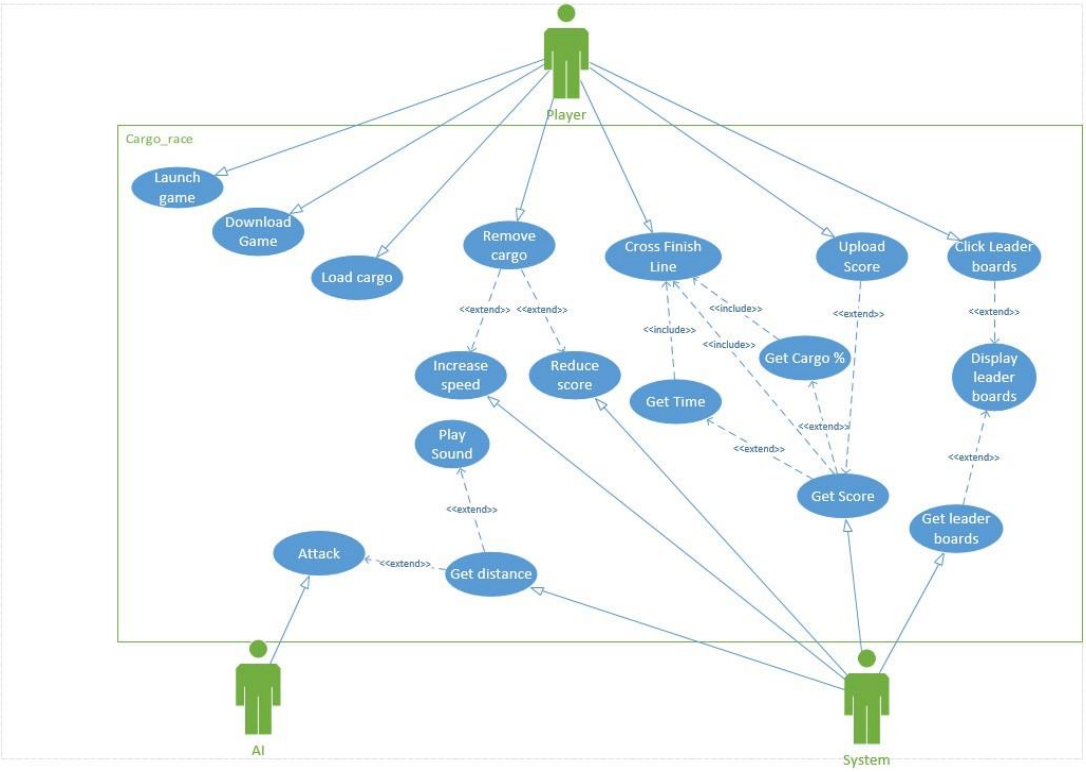
In this instance, a 3D racing game is being designed in which users are tasked to find cargo to load into their vehicle and take to the next level during a 45 second time window called the “setup phase”. After which, they must attempt to transport these objects from the start to finish of the track as fast as possible while avoiding crashing and being caught by the chaser(gorilla).

The player has two health bars, one for the state of their vehicle, and one for the amount of cargo they have left. Their primary objective will be to maintain these as they progress from point A to point B. If the gorilla gets dangerously close, the player can throw cargo from their vehicle to gain speed and escape. This negatively impacts the cargo health bar and the score as a result. As mentioned previously, an objective of this project is to tempt players’ greed, thus cargo being removed from the vehicle will be entirely by player choice, not via physics.

Users are warned when the gorilla is in close proximity to them through audio signalling. They can also make use of the multiple camera angles to figure out the distance between them and the gorilla. These features are key in the decision making of throwing cargo from the vehicle.

The gorilla exists to pressure the player and attempt to eliminate him. It will follow the player and attack when it is within range. If the player manages to greatly outpace the gorilla, the gorilla will then be “rubber banded”, meaning the further the player is away from the gorilla, the faster it will become and inversely the speed will gradually reduce to normal as the distance tightens. This means that a skilled player who activates this mechanic will need to maintain their perfect driving.The game's level is a highly detailed jungle map, it takes players on average around 1 minute and 45 seconds to complete (see customer testing).

2.1.5 Use case model



2.1.5.1 Use Case 001 – Launch game

Scope - This case entails that the game should run after the executable is launched by the user.

Precondition – The user has the game installed on their machine.

Activation – This use case begins when the user launches the game.

Normal Path

1. <Player> opens the executable.
2. The game runs as expected.

Alternate Path

1. <Player> opens the executable.
2. An unexpected error occurs.
3. The game does not run as expected.

Termination – The user is directed to the menu.

Post Condition – The system waits for new entry.

2.1.5.2 Use Case 002 – Select Level

Scope – This use case entails that a user should be able to choose from the available levels.

Precondition – The game is running and the user is on the menu.

Activation – This use case begins when the user clicks on the level select button of the main menu.

Normal Path

1. <Player> enters the level select screen.
2. <Player> selects the first level.
3. The first level is unlocked.
4. The system continues to the next screen.

Alternate Path

1. <Player> enters the level select screen.
2. <Player> selects the second level.
3. The second level is not unlocked.
4. The system prompts the user that the level is not unlocked.

Alternate Path 2

1. <Player> enters the level select screen.
2. <Player> selects the second level.
3. The second level is unlocked.
4. The system continues to the next screen.

Termination – The system runs the selected level.

Post Condition – The system waits for new entry.

2.1.5.3 Use Case 003 – Display Scores

Scope – This use case entails that a user should be able to see the leader boards with the click of a button on the main menu.

Precondition – The game is running and on the main menu screen.

Activation – This use case begins when the user clicks the leader boards button.

Normal Path

1. <Player> clicks the leader boards button.
2. The leader boards appear on the right hand side of the screen.

Alternate Path

1. <Player> clicks the leader boards button.
2. <Player> is not connected to the internet.
3. The panel will show a html error message.

Alternate Path 2

1. <Player> clicks the leader boards button.
2. The leader boards appear on the right hand side of the screen.
3. <Player> clicks the leader boards button again.
4. The leader boards disappear.

Termination – The get request is complete and the leader boards are visible on the screen.

Post Condition – The system waits for new entry.

2.1.5.4 Use Case 004 – Load cargo

Scope – This use case entails that a racer should be able to choose which cargo to carry in their vehicle during the setup phase.

Precondition – The game is running and a level has been loaded.

Activation – This use case begins when the setup phase timer begins to countdown.

Normal Path

1. <Player> picks up a block of cargo.
2. <Player> puts the cargo in the designated position beside the vehicle.
3. The system increases the cargo count.
4. The system removes the cargo from the scene.

Alternate Path

1. <Player> picks up a block of cargo.
2. <Player> attempts to put the cargo in the designated position.
3. The vehicle has no more space.
4. Nothing happens.

Termination – The timer ends and the system continues to the next screen.

Post Condition – The system waits for new entry.

2.1.5.5 Use Case 005 – Remove cargo

Scope – This use case entails that a racer should be able to remove cargo from their vehicle during the racing phase.

Precondition – The game is running; a level has been loaded and has left the setup phase.

Activation – This use case begins when the racing phase does.

Normal Path

1. <Player> presses the remove cargo key.
2. The system removes a piece of cargo.
3. The system increases the player's speed.

Alternate Path

1. <Player> presses the remove cargo key.
2. Less than 3 seconds have occurred since the last removal.
3. Nothing happens.

Alternate Path 2

1. <Player> presses the remove cargo key.
2. There is no cargo left in the vehicle.
3. Nothing happens.

Termination – The action completes and the system reduces the player score and increases speed accordingly.

Post condition – The system waits for new entry.

2.1.5.6 Use Case 006 – Upload score

Scope – This use case entails that a player should be able to upload their score to the leader boards.

Precondition – The game is running; and the player has crossed the finish line.

Activation – This use case begins when the player clicks the upload score button.

Normal Path

1. <Player> clicks the upload score button.
2. The score uploads to the database.

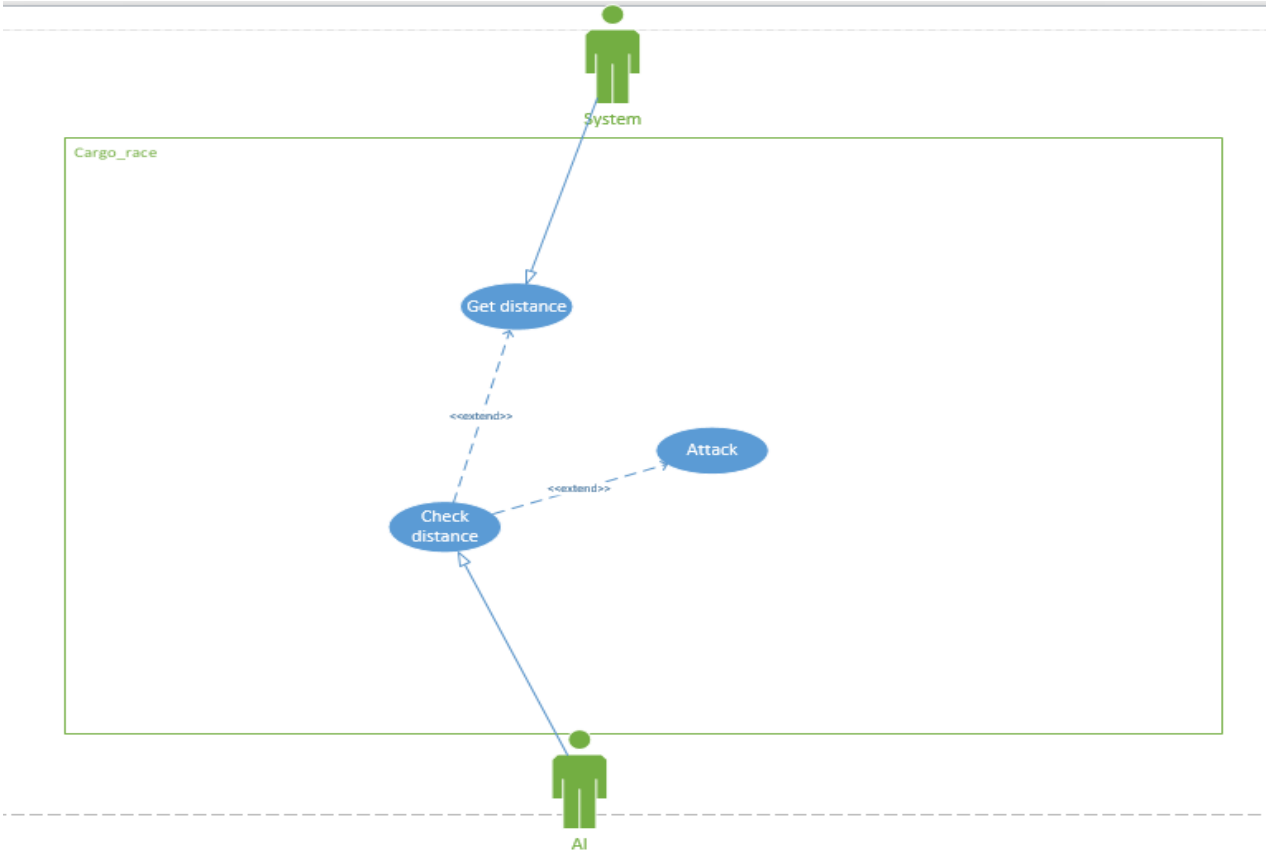
Alternate Path

1. <Player> clicks the upload score button.
2. <Player> is not connected to the internet.
3. The score will not upload, but the player wont be informed of this.

Termination – The post request is complete and a new score is added to the database.

Post Condition – The system waits for new entry.

2.1.5.7 Use Case 007 – Attack



Scope – This use case entails that a computer controlled chaser should be able to attack a racer when within a certain distance.

Pre-condition – The game is running; a level has been loaded and has left the setup phase.

Activation – This use case begins when the racing phase does.

Normal Path

1. <AI> checks if the racer is >3(distance) away.
2. The racer is within 3, <AI> performs the attack action.
3. There was a hit, the system reduces the player health.

Alternate Path

1. <AI> checks if the racer is >3(distance) away.
2. The racer is not within 3, <AI> continues to check once per frame.

Alternate Path 2

1. <AI> checks if the racer is >3(distance) away.
2. The racer is within 3, but less than 5 seconds have passed since the attack action.
3. <AI> continues to attempt the attack action once per frame, if the >3 distance remains valid.

Termination – The action completes and the system increases the AI agent's score and decreases speed accordingly.

Post-condition – The system waits for new entry.

2.1.5.8 Use Case 008 – Cross the finish line

Scope – This use case entails that the score, time taken, and cargo left must be displayed when a player crosses the finish line.

Precondition – The game is running; and the player is in the racing level.

Activation – This use case begins when the player crosses the finish line.

Normal Path

1. <Player> crosses the finish line with 1 or more boxes of cargo.
2. The system returns the score, time taken and cargo left.

Alternate Path

1. <Player> crosses the finish line with no cargo left.
2. The system will calculate the score and will return a result of 0 for score and cargo left.

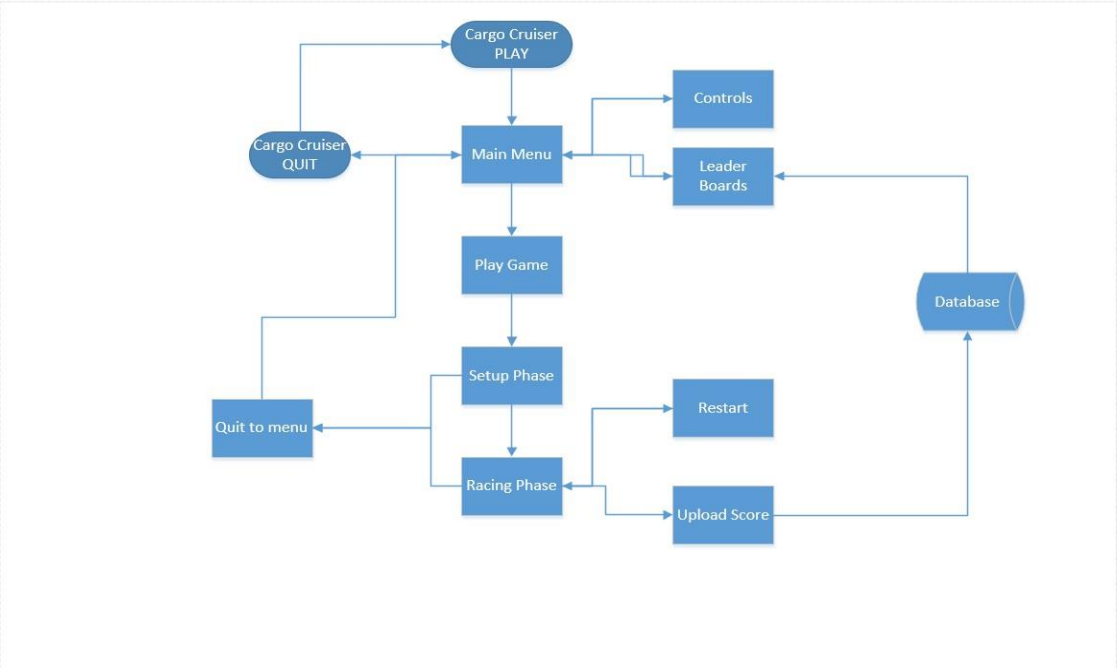
Termination – The score, time taken and cargo left are displayed on screen.

Post Condition – The system waits for new entry.

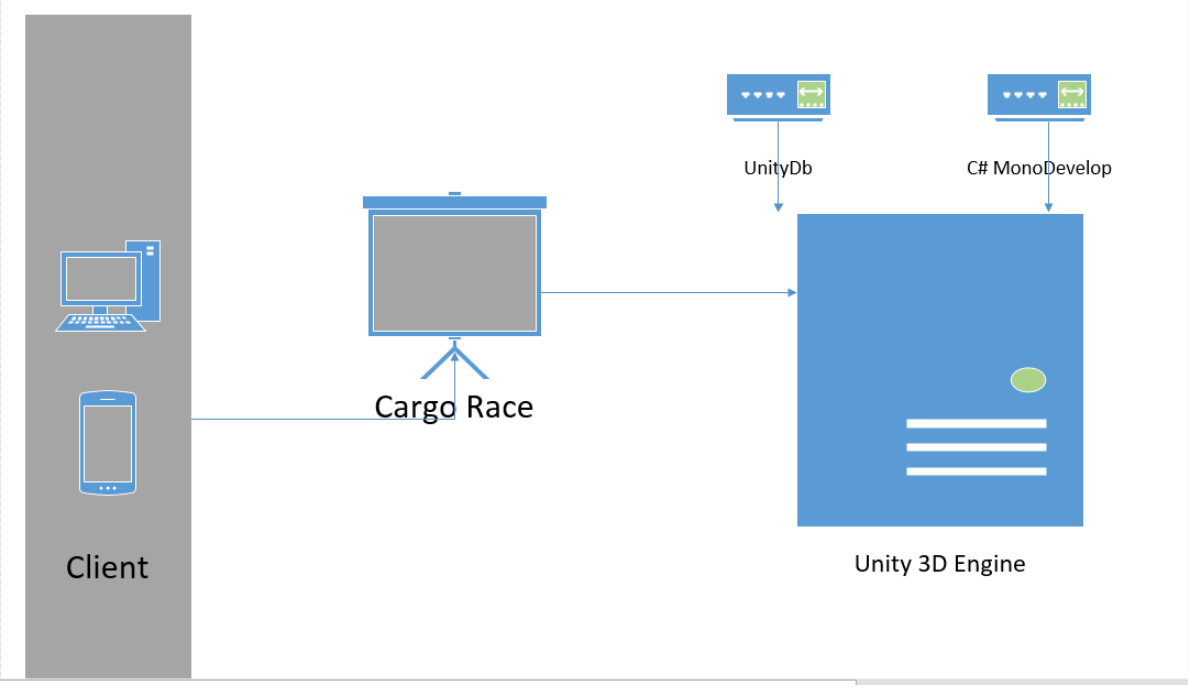
2.2 Design and Architecture

2.2.1 Architecture diagrams

This is a mid-level flow chart detailing how a user interacts with the game.



All algorithms for the game were written in C#. The website was written in HTML and backed up with php scripts.



This is the system architecture diagram for Cargo Cruiser showing how it interacts with the client on a physical level. It was chosen because it presents how the game will be run from the Unity 3D engine, which is supported by its own database and the C# MonoDevelop program for scripting purposes.

2.2.2 How to play copy

All menu screens and UI buttons are controlled with clicking the mouse.

The objective is to first find cargo and put it into your vehicle (setup phase). After which, you must drive the car to the end of the track.(racing phase)

Setup phase (Controls) - Movement with wsad or the arrow keys, cargo can be grabbed by holding down the left mouse button while looking at it, you then must move with it to the rotating platform and release the mouse.

Racing phase (Controls) - Driving with wsad (w speeds up, s slows down, a turns left, d turns right) or the arrow keys, cargo can be thrown by pressing space. The game can be paused at any time by pressing the esc key.

2.3 Implementation

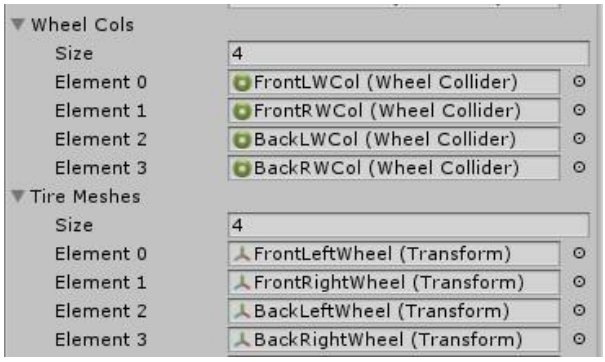
This section will serve to document the game's core functionalities.

2.3.1 Controlling the vehicle

The first thing I needed to do was create two List type variables to hold my wheel colliders and my wheel meshes. In Unity, you can hold game objects (GameObject, Transform, WheelCollider, BoxCollider, etc.) in List types to avoid declaring multiple variables.

```
[SerializeField]
private WheelCollider[] wheelCols = new WheelCollider[4];
[SerializeField]
private Transform[] tireMeshes = new Transform[4];
```

I then assigned the objects to this list from the in-editor inspector.



The car drives on a horizontal and vertical axis, think 1 for driving forward and –1 for reversing/slowing down.

```
void Driving()
{
    /*these input axes are by default assigned to  w,s (vertical) and a,d(horizontal)
    also the arrow keys
    */
    if (stopCar == false)
    {
        steer = Input.GetAxis("Horizontal");
        accelerate = Input.GetAxis("Vertical");
    }
    /*turning on the front 2 wheels
     * finalAngle is how sharp of a turn we can make*/
    float finalAngle = steer * 30f;
    wheelCols[0].steerAngle = finalAngle;
    wheelCols[1].steerAngle = finalAngle;
    /*the wheel colliders are what we use to drive the vehicle
     *they provide rotation speed and angles to our wheel meshes
     *the meshes will try to fill the space of the wheel collider
     */
    if (!isBraking)
    {
        for (int i = 0; i < 4; i++)
        {
            wheelCols[i].motorTorque = accelerate * maxTorque;

        }
    }
}
```

This is the method I made to make the wheel colliders move, as we can see the variable float accelerate controls any input on the vertical axis (w and s, up and down arrow keys). This also controls the vehicle direction and speed by using the motorTorque method of the type wheel collider and setting it to equal our accelerate multiplied by our desired speed. This function then runs in the FixedUpdate() method, as this caters for physics based scripts.

The meshes are updated in a separate method to make them keep up with the 2D wheel colliders and give the vehicle a realistic look. To update the meshes first requires a for loop with a value of 4 (size of both collider and mesh lists). I made a quaternion variable (to track rotation) and a vector3 variable (track 3d position x,y,z), I then get the position and rotation of the wheel colliders, and output them to the i variable of the for loop, the meshes are then set to equal the value of the i variable through both position and rotation.

```
for (int i = 0; i < 4; i++)
{
    Quaternion quat;
    Vector3 pos;
    wheelCols[i].GetWorldPose(out pos, out quat);
    /*get the position of the wheel colliders
     *loop of 4, equal to size of cols and mesh lists
     *quaternions are for rotation, vectors are for positions
     *set the meshes rotation and positions to equal that of the wheel colliders
     */
    tireMeshes[i].position = pos;
    tireMeshes[i].rotation = quat;

}
```

2.3.2 Tossing cargo

This is done very simply by instantiating a prefab every time a player presses the space key.

```
GameObject go = (GameObject)Instantiate(cargothrow1, transform.position, Quaternion.identity);
```

This function can only occur once every 3 seconds, as the script stores the current elapsed time and adds 3 seconds to it every time the space bar is pressed.

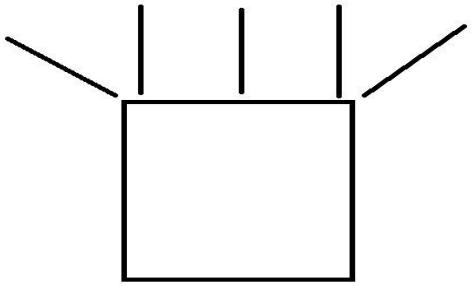
```
 */
if ((Input.GetKeyDown(KeyCode.Space))&&(Time.time>nextUsage))
    {
    /*     when we press space, we set nextUsage to equal the current elapsed time and add delay to it(3 seconds)
      this means it will take time.time approx 3 seconds before it will be greater than nextUsage again
*/
    nextUsage = Time.time + delay;
```

The cargo health bar is updated in the same way, a value of 20 is subtracted from the health every time this function occurs.

2.3.3 Vehicle AI

The vehicle AI follows a set path of nodes (game objects I position in the editor). It will try to reach the first node, and then move to the next until it reaches the last node, when it comes close enough to the current node, it considers the next node as the current node. I re-use the functionality to control the player vehicle in this script to update the vehicle's meshes with its wheel colliders.

To avoid collisions, I added sensors to the vehicle. Sensors tell the vehicle when there is an object in front of it, they must have a long enough range to give the vehicle time to avoid it. The vehicle has 5 sensors.



This is conceptually how the sensors work, it can detect objects directly in front, to the front left, front right, and off at left and right angles of 30 degrees.

```
//front right sensor
sensorStartPos += transform.right * frontSideSensorPosition;
if (Physics.Raycast(sensorStartPos, transform.forward, out hit, sensorLength))
{
    if (!hit.collider.CompareTag("Terrain"))
    {
        Debug.DrawLine(sensorStartPos, hit.point);
        avoiding = true;
        avoidMultiplier -= 1f;
    }
}
```

Snippet of the front right sensor. It is detecting whether or not the raycast object has hit anything not tagged as drivable ("Terrain"). The "avoidMultiplier" variable can be considered as an angle on an axis.

2.3.4 Finish Lines, Death sequences, timers

All UI text, scripts and game objects can be controlled by other scripts using methods like ".enabled" and ".setActive()". Using these you can activate and de-activate whatever you want at specified times.

```
if(start == true)
{
    vehicle.enabled = true;
    gorilla.SetActive(true);
    countDown.enabled = false;
    aicar.enabled = true;
}
```

In this snippet, when the boolean start is true, then the countdown is over and the race has begun, so we enable all the scripts needed to play the game and disable the countdown text.

To handle the player "dying" I used multiple particle effects on the vehicle. Every time the gorilla's hand collides with the player object, the player health is reduced. As the player health is reduced, I made these particle effects gradually visible.

```
void death()
{
    if (v_Health.CurrentValue < 75)
    {
        sevenfhp.SetActive(true);
        if (v_Health.CurrentValue < 50)
        {
            fiftyhp.SetActive(true);
            if (v_Health.CurrentValue < 25)
            {
                twentyfhp.SetActive(true);
```

Sevenfhp, fifthp, twentyfhp are particle effect game objects which show that the car is on fire.

2.3.5 The Gorilla

I made the gorilla "chase" the player by making the gorilla object constantly look at the player object and then translate its position towards the player object. The following code is played in the Update() function so it runs every frame.

transform.LookAt(target);

*transform.Translate(Vector3.forward * speed * Time.deltaTime);*

Previously I mentioned how if a player gets very far ahead of the gorilla, then the script introduces a "rubberband" mechanic.

```
void setSpeed()
{
    if(onetime == true)
    {
        if (distance > 50)
        {
            speed = 30f;
        }
        else if (distance < 50 && distance > 25)
        {
            speed = 26f;
        }
        else if (distance < 10)
        {
            speed = 12f;
        }
    }
```

Using a boolean to detect whether the player has escaped the gorilla, I give the gorilla greater speeds depending on how far away it is, as it gets closer, the speed gradually returns to normal.

The gorilla will chase the player until the player finishes the race or until it eliminates the player. If it does eliminate the player, it will immediately start chasing the AI racing car.

2.3.6 Loading cargo

The way the cargo is "loaded" into the vehicle in the setup phase is by the player grabbing a box and bringing it to the rotating platform. I use tagged "Triggers"(these are box colliders tagged as triggers, for use in scripts) to detect when a cargo object has entered the space, this cargo object is then destroyed and the cargo collected variable is increased.

```
if (other.gameObject.tag == "cargo")
{
    cargoCount++;
    Destroy(other.gameObject);
}
```

2.3.7 The website and leader boards

To make the website I used a simple bootstrap template. The only purpose of it is to have a space to download the game, and display the highscores attained by players, so it was kept simple.

I hosted the website on the "x10hosting" service and hosted the game exe on the Amazon AWS cloud platform.

http://cargorace.x10host.com/

*Note: You will not be able to download the game, I will make it publicly downloadable for presentation purposes and afterwards take the mirror down to avoid potential hacking of my code.

To implement the leader boards, I made a small database in MySQL and used php files to access it, one for displaying them and the other for adding new scores.

```
$sth = $dbh->query('SELECT * FROM scores ORDER BY score DESC');
$sth->setFetchMode(PDO::FETCH_ASSOC);

$result = $sth->fetchAll();

if(count($result) > 0) {
    foreach($result as $r) {
        echo $r['name'], ":\t", $r['score'], "<br/><br/><br/>";
    }
}
```

Here I ran a query to select records by score, using a "foreach" loop to sort the rows in a neat order.

To call these php scripts in-game I simply call the url and use a get request in a C# script. I then assign the text of the result of this request to a text field that is displayed on the canvas in-game.

```
IEnumerator HandleWWWRequest()
{
    string content = "cargorace.x10host.com/displayUnity.php";
    WWW www = new WWW(content);
    yield return www;
    string scoresDataString = www.text;
    scores.text = scoresDataString;
}
```

Adding highscores is done in-game upon completion of the race. I use a C# script to send a post request in the form of a WWW form to the php script.

```
public void insertScore()
{
    WWWForm form = new WWWForm();
    form.AddField("namePost",nameTF.text);
    form.AddField("scorePost", scoreField.text);
    WWW www = new WWW(scoreURL, form);
```

The php script "addscore.php" sends a query to the database which adds the player name and high score based on the values sent to it.

2.3.8 Audio

The audio in the game is controlled by scripts and box colliders. Soundtracks are played when the scenes start, there is no need to control them.

```
void OnTriggerEnter(Collider other)
{
    if(other.gameObject.tag == "inrange")
    {
        if (!aud.isPlaying)
        {
            aud.Play();
        }

    }
}
```

This is how the gorilla audio is controlled. When he enters a Trigger Collider that is attached to the player object, the sound will play. It first checks if the sound is not playing already so it does not repeat every frame.

2.3.9 Hide/unhide the cursor and pause/un-pause the game

By default, the cursor is visible on the screen, this is not suitable for a third person racing game. To hide it, I needed to set the cursor state to locked, and unlock it when I press the escape key to pause the game.

Cursor.lockState = hide = CursorLockMode.Locked;

Cursor.visible = false;

This will confine the cursor to the center of the screen and hide it from view.

Cursor.lockState = show = CursorLockMode.None;

Cursor.visible = true;

This will unlock the cursor and make it visible.

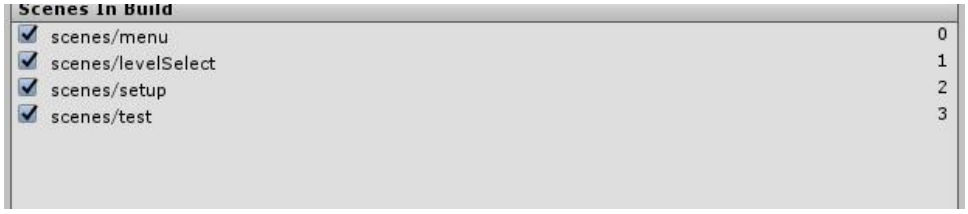
To pause and un-pause the game I use the Time.timescale method in conjunction with the cursor being visible. Provided the player is not past the finish line, when you press

the escape key we set the cursor to visible and the time scale to 0 (pause, 1 is play) or vice versa.

```
if (Cursor.visible == false && Time.timeScale == 1)
{
    Cursor.visible = true;
    Cursor.lockState = show = CursorLockMode.Confined;
    Time.timeScale = 0;
```

2.3.10 Loading new scenes

A requirement for the game was to switch between different levels. The game in Unity has a build structure, which is the load order of scenes.



0 is the first scene that is displayed. The first way I switch scene is with button click methods. Using the scene manager, I get the active scene's index and add 1 to it, which moves to the next scene.

```
}
public void ChangeSceneTest()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}
```

I attach this method to a UI button, so when it is clicked, this method runs. I used the same method for the setup phase except I activated the method when the timer reached 0.

2.4 Graphical User Interface (GUI) Layout

In this section I will describe the UI, graphics, camera angles and perspectives in the game.

This is the main menu, it displays an animated scene in Unity of a burning vehicle and caravan, and two gorillas run across the screen. This is set up as such to get the concept of the game across at an early stage. The Play button will go to the next screen.

The controls button will display the game's controls on the screen. The leader boards button will show the top 5 high scores made by players.

This is the screen that follows when a user clicks the Play button. The purpose of this scene is to explain the concept of the game and instruct the user on what they have to do.

This is the setup phase. The platform at the back of the vehicle on the left side is where the player must drag and drop boxes of cargo, this is made abundantly clear by the fact that in-game it rotates. We can also see the timer and the amount of cargo we have currently saved at the bottom of the screen.

This is the racing level. The two bars on the bottom left of the screen are the vehicle health(top) and the cargo health(bottom). I have made the distinction between this clear by using different colours, as well as the box and car icons beside their corresponding bar. The speed is in mph on the bottom right of the screen.

2.4.1 Player View

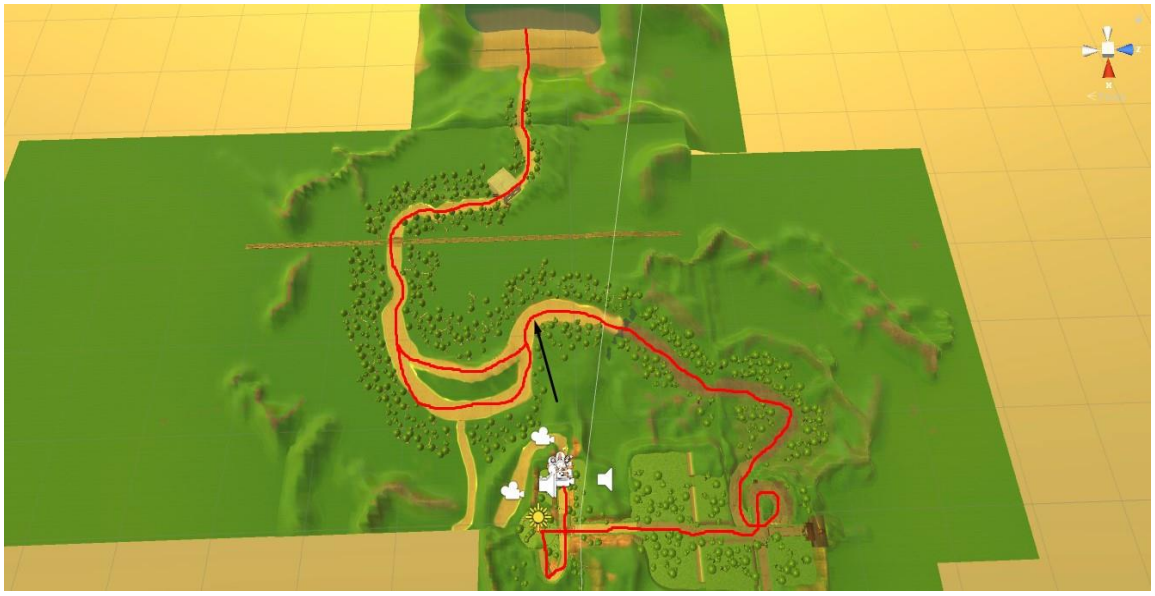
The camera is static (locked in place) and follows the player vehicle. As stated previously, the player has access to third person, first person and behind the car perspectives.

This is the third person perspective. This is the first viewpoint the player will be presented with. The wheels are animated with physics and the exhaust pipes blow smoke.

This is the first person perspective. It gives the player a view of the steering wheel from inside the car as is common in many racing games. The wheel and the arms are animated and will turn with player input.

This is the behind the car view. The purpose of it is so the player can see how close the gorilla is. The car will blur out of view to focus on other aspects of the environment.

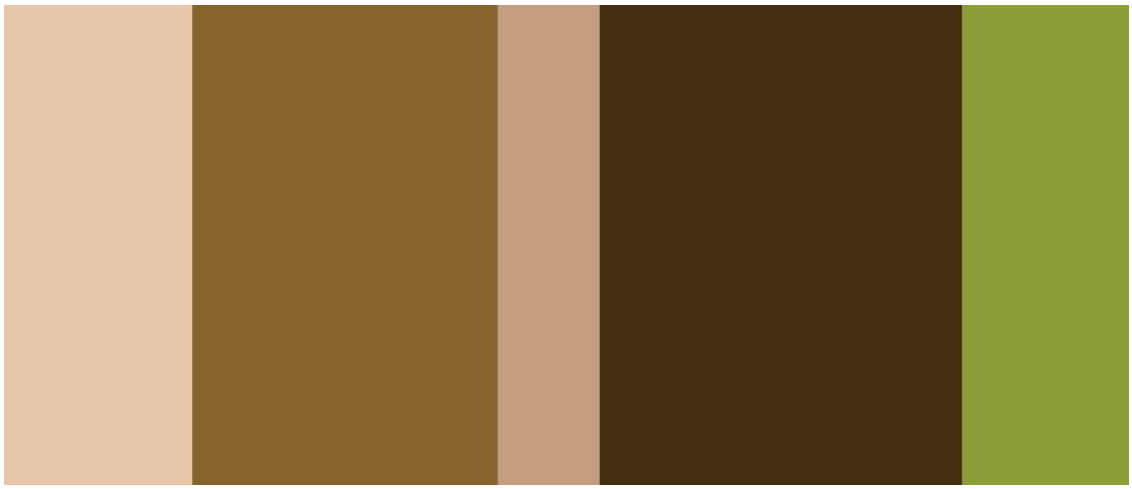
This is the player perspective when they have been eliminated by the gorilla. The health bar is visibly empty and the car is on fire.



This is a perspective of how big the level is vs what the player sees. I have drawn the route the player takes through the level. The black arrow points to where the behind the car view(previous picture) was taken. It is stated previously in this document that the level is approximately 1 minute and 45 seconds long travelling at high speeds in the car.

2.4.2 Visual Design

The level I set out to create was a race track surrounded by a jungle. With this in mind, I decided to keep the following colour scheme in mind as a template and proceed from it.



*Ref: **2** see References(5)*

Exceptions were made to this but it was useful in figuring out I should not use too many different colours in the level.

The graphical style in the game was enhanced with image and lighting effects. To demonstrate the changes I made, I will compare a screenshot of the default unity settings with a screenshot containing the effects I implemented. Changes were made by adding image effect scripts to the camera and adjusting the lighting settings.

Default graphics, no lighting effects, no image effects.

Enhancements made: Fog, Colour Correction Curves, Screen Space Ambient Occlusion, Depth of Field.

2.5 Testing

Testing was performed in the following ways:

- Test cases for anything that occurs by player input
- Integration testing in Unity for static occurrences (i.e. occurring without input from the player)
- End-user/Customer testing(see heading 2.6)

Test cases were done both before and after customer testing. I tested all core inputs of the game that I could identify. Test cases include, vehicle controls, tossing cargo, main menu buttons, walking controls, grab function, pause function.

Test Case ID	Test_008	Test Case Description	Investigate reported bug		
Created By	Sean Trant	Reviewed By	Sean Trant	Version	0.19

QA Tester's Log

Tester's Name	Sean Trant	Date Tested	April 22, 2017	Test Case (Pass/Fail/Not	Pass

S #	Prerequisites:
1	Access to unity
2	Access to the current project version
3	The race scene
4	

S #	Test Data
1	the esc key
2	the mouse
3	retry button
4	pressesc()

Test Scenario: User reports that when you pause the game and click retry, the game does not unpause

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run the project	The game scene should play	As Expected	Pass
2	press esc	The game should freeze, two buttons should appear	As Expected	Pass
3	click retry	The scene should start again and play	The scene starts again but does not restart, as user reported	Fail

This was a significant test case as it originated from a bug reported by a participant. As can be seen here there was a problem with the pause functionality which I confirmed through a test case and was then able to fix.

To implement integration tests in Unity I made a separate scene to handle tests. Tests are not included in the final build or in the other scenes as they hinder performance. It is standard to include all tests in a separate scene. I tested the following elements:

Can "health" be reduced when the gorilla intersects with the player object: To do this test I made a small script with a health variable of type float, that is reduced by 50 every time it collides with an object tagged as "gorilla". When health reaches 0, the box collider is disabled. I then use an assertion component to check is the box collider enabled vs a constant boolean value of true. This means that when the health reaches 0, this assertion component will fail, so I set the integration test to succeed on AssertionException (when the assertion component fails) as I know that when the component fails the test has done exactly what I want it to.

Can the gorilla follow an object that does not move in a straight line: This test was done to prove that the gorilla can follow an object that turns and changes direction. I put a trigger collider where the turning object stops and put a call test on this, so when the gorilla reaches that point, the test passes. Since the gorilla is scripted to follow an object, we know the only way it could reach this point is if it followed the object there.

Test AI speed: This test proves that the AI vehicle can start from 0 and reach a point before a timer runs out. If the vehicle does not reach the trigger collider in less than 10 seconds, the test fails, if it does, it passes.

Test the car's rigidbody: This test proves that the player car obeys gravity, specifically that it falls from a height to the ground within a given amount of time. Using a call test, if it reaches the ground before 5 seconds, the test passes, if not, it fails.

Test making text visible: This test proves that text can be made visible when the player or any other specified object enters a marked trigger collider. It succeeds in the same

way as the health test, through the failure of the assertion component (AssertionException).

These integration tests are important because it lets me as a developer to never need to worry about any of these functionalities again. I have solid proof that they work and thus can focus on other tasks.

2.6 Customer testing

I elicited feedback from multiple participants in various ways and in different stages of development to improve the project and cater to the customer market.

2.6.1 Trunk Test

For this test I gave participants a build of the game with the race scene and asked them to make it to the end of the track. I was trying to find out how difficult the game was and any other problems players who had never seen the game before would have. I told them the controls as this test was done in an earlier stage of development where controls were not explained in the game. The build exe they used was created on March 20th.

I used another build later on (April 5th) to see if the changes I made improved the problems identified by participants previously. This build had a main menu with the controls visible, leader boards can be displayed here, you can upload your score to the leader boards at the end of the race. The setup phase and rubber-banding mechanic for the gorilla had also been introduced at this time.

Colm, 19

*Note: This participant received 2 builds of the game on separate dates, with different objectives in mind

March 20th

Did you make it to the end? Yes.

How many attempts did it take you? 3.

What were the biggest difficulties? I crashed a lot, it is hard to slow down and turn at high speed.

What time did you complete the level in? 1 minute 41 seconds.

What would incentivize you to invite your friends to play this game? Maybe if I could see my friends scores I would want to try and beat them.

April 5th

Did you find all the cargo in the setup phase? Yeah and I had about 20 seconds to spare after I found it all.

Was it easier to drive than last time? Yes it was definitely easier to handle, it is still hard to slow down at high speeds, but clearly better than before.

Did you make it to the end of the race? Yes.

How many attempts did it take you? 1.

What time did you complete the level in? 1 minute 34 seconds.

Did you see any problems with the game? I do not understand what this game is about or why I am being chased by a gorilla.

Aron, 21

Did you make it to the end? Yes.

How many attempts did it take you? 11.

What were the biggest difficulties? The game was very laggy on my computer. I struggled to finish it as I constantly either crashed or got killed by the gorilla

What time did you complete the level in? 1 minute 55 seconds.

What would incentivize you to invite your friends to play this game? If it was less laggy and if I could play against them in multiplayer.

John, 23

Did you make it to the end? Yes

How many attempts did it take you? 6.

What were the biggest difficulties? Crashing into things and having to start over. I also had to do a double take and make sure I was turning the right way on the first hill going over the bridge.

What time did you complete the level in? 1 minute 43 seconds.

What would incentivize you to invite your friends to play this game? I get a score at the end but I can't do anything with it, if I could like share that with my friends or buy new items for my car with it or something.

Ernst, 23

Did you make it to the end? Yes.

How many attempts did it take you? 7.

What were the biggest difficulties? Crashing and getting stuck on objects. The gorilla kept killing me at the start of the race.

What time did you complete the level in? 1 minute 46 seconds.

What would incentivize you to invite your friends to play this game? I like to set high scores in games, so if it had leader boards I would want to play against my friends.

Lee, 22

*****Note, this participant received only the April 5th build.

Did you find all the cargo in the setup phase? I don't know how much cargo I was supposed to find.

Did you make it to the end of the race? Yes.

How many attempts did it take you? 3.

What were the biggest difficulties? It seems as though if I crash once I am basically dead. I could never escape the gorilla

What time did you complete the level in? 1 minute 41 seconds.

Did you see any other problems with the game? I didn't know what I was supposed to do at times. I would like the menu to have nice wooden buttons.

2.6.2 Five Second Test

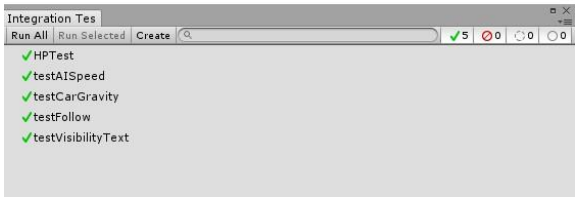
This test was done for the website. I asked one participant (the website is very simple, it does not require vigorous testing) to look at it for 5 seconds and try to remember what he could see. I then asked him some questions:

Aron, 22

1. **What was the name of the website?** Cargo Cruiser
2. **What was the page about?** A driving game
3. **Rate the page between 1 and 5?** 5, I liked the art style.
4. **What element of the page did you focus on most?** The car.
5. **What can you do on this website?** Download the game.

2.7 Evaluation

All integration tests passed without issue.



One test case failed as shown in section 2.5.

From reading the results of my customer testing, I identified a couple of problems and evaluated them by rating them as catastrophic, minor, severe or intermediate. This then enabled me to plan solutions and implement them in my next development phase.

Minor: Not game-breaking, usually cosmetic or visual requests.

Intermediate: A shortcoming in the game, but one that the player can work around themselves.

Severe: A problem that seriously hinders the game's playability/re-playability.

Catastrophic: Game breaking, a problem that makes a player totally unable to play

Problem: The car is difficult to handle	**Rating:** Intermediate
Description:	When the player reaches high speeds, they have difficulty turning the car and slowing down, this leads to them spinning out or crashing into the environment.
Solution:	Fine tune variables Steer and Accelerate. Also, alter the Rigidbody's Drag and Angular Drag.

Problem: The	**Rating:** Severe

score has no purpose	
Description:	When the player finishes the level, they can see their score and that is essentially the end of the game. Nothing happens after and thus there is no incentive to replay it.
Solution:	Setup a lead board system and allow players to see the high-scores and upload their own.

Problem: The game was laggy	**Rating:** Catastrophic
Description:	Ideally, the game should be playable on most systems. Serious lag (low frames per second) will make the game unplayable. The lag was reported to occur in the racing level.
Solution:	Delete excess trees to improve performance.

Problem: The context of the game is never explained	**Rating:** Minor
Description:	It is unclear what the objective of the game is as no context is given to the player. This is a problem as the player may not realize they can throw cargo out of their vehicle, for example.
Solution:	Set up a scene before the game begins to explain the objectives of the setup phase and racing phase.
Problem: The player has too much time in the setup phase	**Rating:** Intermediate
Description:	It is far too easy to find 5 boxes in 60 seconds, the setup phase level is small, so testers were left waiting around after finding the boxes for approximately 20 seconds for the phase to end. The player should be feeling pressure in this phase, not comfort.
Solution:	Easy fix, reduce the timer to 45 seconds, slightly increase the distance between the spawn points.

Problem: The game is still paused after pausing and clicking retry	**Rating:** Catastrophic
Description:	If a player pauses the game and clicks restart, the game will remain paused. This is a huge issue as it will force the player to exit the game.
Solution:	Set the time scale to equal 1 in the Awake() function of the script.

The testing done was evidently of great use from a technical perspective. The problems identified are ones that I may never have seen had I not set up the testing environment and conducted the various tests.

3 Conclusions

Advantages: The objective of the game is to introduce new, exciting mechanics to a racing game and the document shows the approach that was used to solve this from a logical and technical perspective. The leader boards add a competitive aspect to the game, enabling players to compete with each other and giving it re-playability. The game was vigorously tested by both myself as a developer and with end-users, this greatly improved the game as a result.

The visual design of the game is very impressive and the level is highly detailed. The different image effects add a layer of photo realism to it. The use of monkeys as characters instead of humans makes the universe of the game more enjoyable and interesting, it is not to be taken seriously. The addition of the AI racing car makes the player feel less alone.

Progression is an important part of a game, it should get harder as a player becomes more experienced to keep them invested in gameplay. The rubber-band mechanic of the gorilla makes it so a skilled player will have to drive without error to make it to the finish line, one crash will be fatal.

Disadvantages: A drawback of the level being so highly detailed meant there was time for only one. The AI racing car does not always complete the path, often times it will crash along the way, this is due to the nature of the complex geometry. The frame rate can be unstable on weaker machines, I improved the frame rate as much as I possibly could but old machines will have to turn down graphical quality.

Opportunities: The project has massive room to grow, this is discussed in detail in section 4: Further development/System Evolution

Limits: The game is single-player only, this limits the re-playability value.

4 Further development / System Evolution

The most striking way this system could evolve is through implementing forms of progression or adding new challenges. As per the details of this document, there is no real progression to speak of. Time permitting, it would be fitting to implement progression in the form(s) of:

- Money: If the objective for racing players is to retain as much cargo as possible, there should be a reward for doing so. The system could accommodate this if it is made possible to use players' score from a race to buy upgrades for their vehicle (improve speed, improve weight capacity, cosmetic changes etc.).
- Multiplayer: Beyond leader boards, if players could actually play against each other it would really enhance the value of this game. Four players racing in cars and one player as the gorilla trying to stop them.
- Cargo weight and value: There is no variation in the cargo you can collect in the setup phase, I think it would be a nice addition if cargo would weigh down the vehicle and be worth more score points.
- The AI can be improved with an iterative learning algorithm. For this project I had to fine-tune the position of every node, just one being out of place would cause the car to crash. If there were an algorithm that could play the game by itself and improve gradually it would be far easier and result in a better AI.
- Second level: A second level was something I wanted to create but did not have time due to the high detail standard I set for myself. The second level will take place under the sea, with the shark as the antagonist.
- Maze mode: The second level, instead of having a setup phase, could present players with multiple paths to the finish line and have them collect cargo on the way there, while avoiding the chaser.
- Last man standing (winner takes all): Instead of having a finish line, the jungle race could loop until there is only 1 racer left. The chaser would have to constantly gain speed, to ensure all racers are eventually eliminated. This synergizes well with the money aspect discussed previously.

Otherwise, if the project were to have extra resources, there would be a general improvement in all aspects of its quality. Developing without funds means one is restricted to basic versions of programs like Unity for example. With a professional license the possibilities become greater with the large amount of assets, live support and improved graphical quality, or even a more powerful game engine.

5 References

1. "Unity - Unity - System Requirements". *Unity*. N.p., 2016. Web. 11 Dec. 2016.
2. *Theaceofspaceblog.com*. N.p., 2017. Web. 6 May 2017.

6 Appendix

6.1 Project Proposal

PROJECT PROPOSAL - CARGO CRUISER

Sean Trant, x13332576, seantrant@gmail.com

BSc Hons in Computing

Gaming and Multimedia

19/10/2016

Objectives

- Setup phase
- Load cargo and chaser sets obstacles and traps
- Race against other players/AI, maintain as much cargo as possible and go fast to win
- 2 maps (jungle and underwater)
- Chasers: gorilla and shark

The idea behind this project is a racing game in which a player's vehicle will have a cargo compartment with cargo inside. Racing other players to the finish line, the player must keep as much cargo inside the vehicle as possible as score is not just reliant on who finishes first, but also who has the most cargo remaining.

There will be a chaser in the race, currently plans for a gorilla in a jungle map and a shark in an ocean race. The chaser will pursue players through the race and attempt to take them out of it. If the chaser gets close to a player, the player has an option to throw some of his loaded cargo away to go faster.

Before the race begins, there will be a setup phase in which the chaser will have a certain amount of time to lay traps and move the obstacles of the track to his own preference. During this time, the player can choose from a pile which cargo to load into their vehicle, some being heavier than others but contributing more to the score at the end.

Background

The idea stemmed from a simple project from 3rd year, a racing game in which a player would lose points when cargo fell out of the vehicle. Upon completion of the project, I felt unsatisfied, like my team and I had merely scratched the surface of a good idea. So, from here I expanded and developed it to the point where it was entirely upgraded.

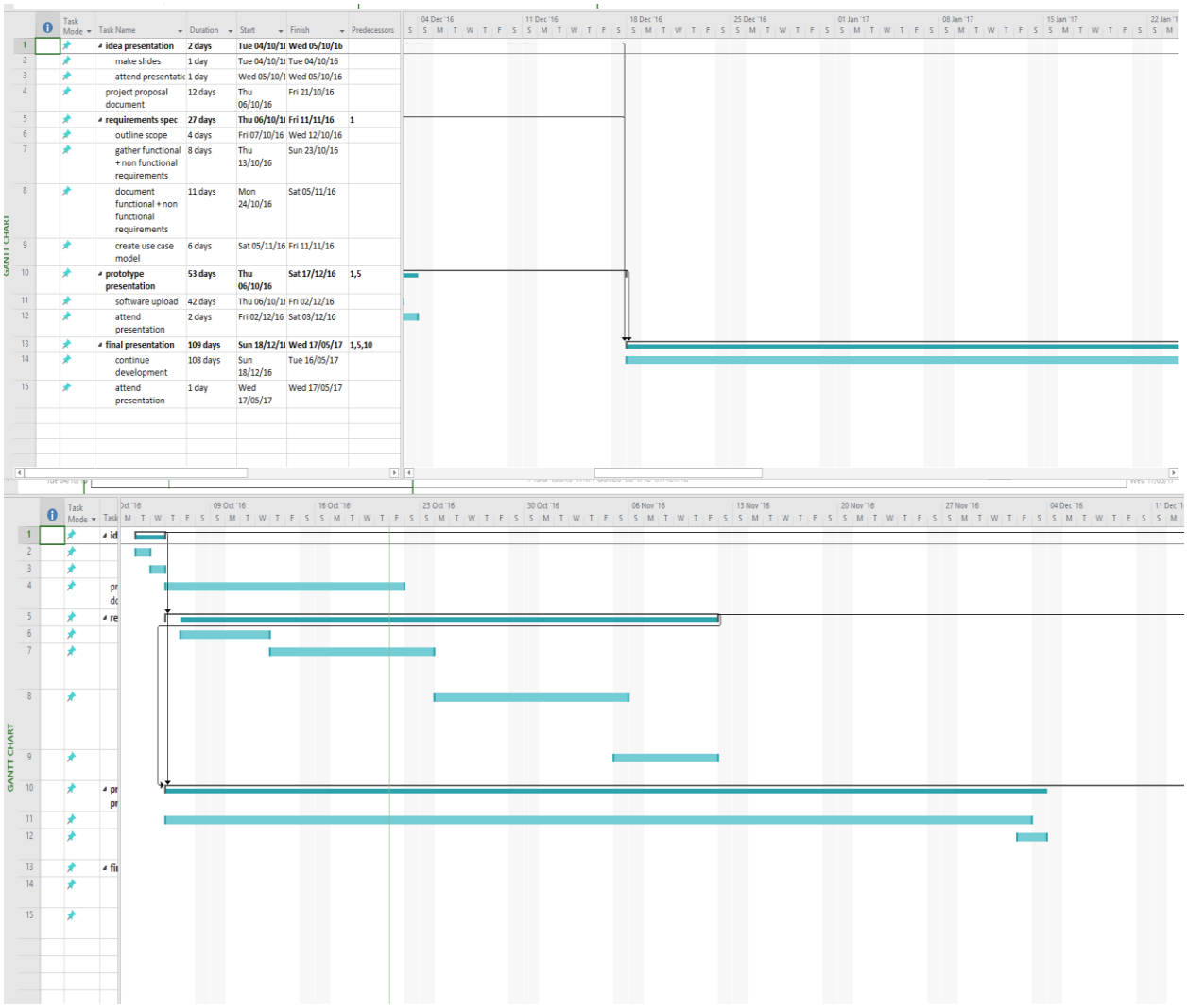
The highly popular mobile game Temple Run showed that the idea of being chased could add a fresh improvement to the standard racing game. I then found this thought reinforced by a multiplayer game called Speed Runners, in which 4 players race each other in a 2D setting continuously until the camera closes off behind them and begins eliminating players from the race.

In the initial idea proposal, I was advised to add more original elements to the game. This made me think of a setup phase, which could allow players to alter the race track and select different pieces of cargo to load onto their vehicle, which would affect its weight and speed, but also the value at the end.

Technical Approach

The technical approach at this pre-development / planning stage is largely subject to change, but I am currently looking at developing the game using an engine called Unity. Unity is a free, cross platform development engine so it is simple to mount projects on both computer and mobile. The central language for this engine is C#, and I will be using Autodesk Maya to make models or other environments and animations.

Project Plan



Technical Details

Unity, C#, javascript, Autodesk maya

Evaluation

I plan to test the project under the following conditions:

- Does it satisfy the functional requirements that drove development in the first place?
- Do the functions complete within an acceptable timeframe? Is the game stable?
- Does it run on all intended platforms?

These conditions will make it easier to identify problems and address either the problems themselves or unachievable requirements.

6.2 Project Plan

The project plan is pictured in the proposal.

6.3 Monthly Journals

6.3.1 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: October

My Achievements

This month:

Created the model for the vehicle that will be used in the first track, done using Maya 2016, I am happy enough with the result. I do not need to animate the vehicle itself, as the vehicle script I use in Unity uses the models wheels as “GameObjects” and physically rotates these to drive.

I also started looking at the logic behind the functionality of tossing cargo out of the vehicle. Initially I wrote in English what I want to accomplish and then began to code it. What I did this month was create a script with a Score attribute and an ArrayList. When the spacebar button on the keyboard is pressed, 10 points are deducted from the score and the latest entry of the ArrayList is removed (I know this is working as there are no errors/crashes, but I currently can’t display the list on the screen).

By default Unity will run methods you call in an “Update()” function, which occurs in every frame, and the game will run at anywhere between 30 and 60 frames per second, so this was a problem as the score would go from 100 to 0 with one button press. To solve it I added a time delay so that this function can only occur once every 3 seconds when the spacebar button is pressed. This is a central mechanic of the game and is nearly finished.

Intended Changes

Next month:

The most pressing task to complete within the next week is the requirements specification.

Earlier, I went over how I began to code the functionality of tossing cargo out of the vehicle. What is left from this perspective is playing an animation with the button press, and then displaying the ArrayList(may change to the Stack type) on the screen and removing entries along with the score.

The vehicle model is finished, but I still need to export it to Unity and make it a drivable entity, however I have done this before so this won’t be a time-consuming problem.

I need to make a model for the gorilla, animate it and then make it an AI controlled entity in Unity. This is something I would like to have somewhat completed in time for the prototype presentation as proof of concept.

Supervisor Meetings

Date of Meeting: 26/10/2016

Meeting Summary: The first supervisor meeting took place on this date. It was a group meeting in which we discussed our ideas as a group and what technologies we were using. The most meaningful advice I took from this at the time was to treat the prototype

presentation as a proof of concept, as in code enough functionality to show you can actually program what you described in your proposal. This was useful as initially I only planned to create models and a race track.

6.3.2 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: November

My Achievements

This month:

Last month I left myself with a couple of critical tasks to complete before the prototype presentation:

- Requirements Specification – I felt that this was finished to a strong standard after sending it to my supervisor for feedback. Documenting the user requirements and use case model also left me with a couple of new ideas that I should attempt later on.
- Introduce the gorilla chaser to unity – I modelled the gorilla in Maya 2016 using image referencing and added a running and attacking animation to it for demonstration in the prototype. The animations are controlled using a method in C#. The gorilla will currently follow the player object, the running animation is assigned as idle (i.e. if nothing else is happening, the running animation will loop). The distance between the gorilla and the player is recorded, when this distance is less than 2, the attack animation will play.
- Display and update an ArrayList on screen – I was able to display the ArrayList on the UI canvas in Unity using the GUI.Label function in Unity's C# library.
- Introduce the vehicle model to unity as driveable – This was done as expected. The script for making an object a driveable vehicle requires four wheel objects (meshes) from the model. It also requires four "Wheel Colliders". These are game objects which act something like axes for the models wheels to rotate around. We then use the Input Axes Vertical and Horizontal to control our direction in steering. The meshes and colliders are stored in arrays which we loop through to keep consistency between all of their positions and rotations.

Intended Changes

Next month:

The most pressing task at the beginning of next month will be finalizing preparations for the prototype presentation. While I feel as though I have enough of a foothold to demonstrate my idea, I would like to carve out the beginnings of the first track to improve the visual appeal of it. Of course, I will also need to make slides for the presentation as well as complete the Technical Report to the best that I can.

While this is something that is not likely to be rectified next month, it should be mentioned now. The gorilla model has separate pieces for every body part, arms, legs, head, etc so the animations do not look as fluid as I want them to. It is reminiscent of

something like a Lego piece. To fix this I will have to start the model from scratch (or delete everything except the torso) and make the whole thing in one piece, and subsequently learn how to use the bone and joint features in Maya to animate it.

Supervisor Meetings

Date of Meeting: 28/11/2016

Meeting Summary: In this meeting I informed the supervisor of what I had accomplished so far (discussed above and in the previous monthly report). I also did not need help at the time, but thought I had to setup meetings anyway to attain the supervisor communication marks. My supervisor told me that she could simply be updated via email, which I will do from now on, unless I need help with a coding problem. I was also advised to demonstrate in the presentation that, although I had a good chunk of work done, I still had a significant amount left to do.

6.3.3 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: December

My Achievements

This month:

The objective for December was finalizing preparations for the presentation and do the presentation itself:

- I made my presentation slides. I stuck to the template and felt the slides were not the strongest, however I used feedback from my supervisor to improve them somewhat.
- Last month I said I wanted to create the beginnings of a track and this is one thing I set out to do. However, it took a lot longer than anticipated to do even a small amount of the track while ensuring it had an acceptable level of detail. The reason for this is Unity does not accommodate the creation of complex shapes or models, i.e. hills, trees, so I had to use Maya to create individual parts of the track and then import them back into Unity. The best way to describe how this works is to think of it like a jigsaw, I just need to make all the pieces and stick them together to create a race track
- While not planned for, I decided I had time to create a small scene in Unity to demonstrate in the presentation the setup phase that takes place before the race. In the scene I created a bundle of different shapes to take the form of cargo. I created a script which allows the player to click and drag on these objects to move them to the vehicle. Rather than physically load the cargo onto the vehicle through animations, I left a slowly spinning object on the ground which will be used as a designated area to drop cargo for the vehicle. When cargo is dropped in this space it will disappear and increase an integer displayed on the side of the screen.

Checklist – Pressing tasks

It is apparent that I will need to dedicate more time to the level design. As mentioned previously, the whole ten seconds-worth of track I had in the presentation took a long

time. This may become faster with the more road and hill piece models I have, as they can be recycled numerous times.

I need to look at making the gorilla a playable character, this was something I completely forgot to mention in the presentation.

I focused too much on speaking about how I needed to create AI in the presentation and massively overstated its importance. If multiplayer can be implemented, then AI will be mostly redundant.

Need to introduce sound to the game. This is a very important task as sound will make the game that bit more engaging. Additionally, I need to create the character that will sit on the player vehicle and throw cargo out of it. These tasks are related as the character on the vehicle will be how I deal with audio and visual signalling in chase situations.

As expected, I still need to remodel the gorilla. I realize this may not be that worthwhile from a marking perspective so I will focus on the other tasks first.

Supervisor Meetings

Date of Meeting: none

No supervisor meetings, however I did communicate with my supervisor via email. This was to receive feedback on my prototype presentation slides. The feedback I received was useful and helped me improve them.

6.3.4 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: January

My Achievements

This month:

This month did not play out as intended, I managed to do a significant amount of level design although the first track is still mostly unfinished. I also made the gorilla a playable character with a third-person character control script, however after brief testing I have decided to scrap this and go for a first-person setup instead. The reason for this is the third-person script I have does not allow the camera to be controlled with the mouse, making for very awkward navigation, additionally as the gorilla is so big you can't see where you are going from a third-person perspective.

Intended Changes

Continue dedicating time to the level design, it must not be left till the end of development.

I need to look at making the gorilla a playable character, this was something I completely forgot to mention in the presentation.

I need to research multiplayer in Unity. I also am aware of an approach for the AI which is making the level's surfaces a "nav mesh" allowing the computer to direct objects through it, however I feel multiplayer is a better approach.

Still need to introduce sound to the game.

Supervisor Meetings

Date of Meeting: none

No supervisor meetings, however I did update my supervisor of my progress via email.

6.3.5 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: February

My Achievements

This month:

Continuing last month's trend, this month I completed a lot more of the level. I learned new techniques in level design, which in hindsight would have sped up development if employed earlier. Unity has a built-in terrain designer, which I was aware of but did not consider trying to use, it is a far better solution to the earlier approach of making individual track pieces in Maya and importing them to Unity.

I updated the cargo tossing system. Cargo can now be seen leaving the vehicle when the toss cargo button is pressed. To do this I created a simple box object in Maya and imported it to Unity. I then made this object a prefab in my assets. I then created a script which assigns "force" to the given object. Force is a part of the Rigidbody component that enables it to move. I then instantiate this class in the TossTest script, which updates the score every time the spacebar is pressed. Finally, I use the Shoot() function to handle the launching of the cargo object. This uses time and a speed float.

I made the gorilla playable in first person, the movement controls are there as I am currently using the default Unity first person controller. However, I am hesitant to proceed with this deliverable in case I decide to go ahead with the remodelling of the gorilla (this is dependent on time).

I looked at making AI controlled characters, however there are complications with the "nav mesh" system. Basically, you have to set the environment to "walkable" and "not walkable" so the AI knows what route it can take to the set target. This is a problem as I have both racing cars that must obey the road and a gorilla, which can move independently around the map. Multiplayer seems like the better option.

Intended Changes

Continue dedicating time to the level design, it must not be left till the end of development.

Finish the vehicle and its scripts by adding the monkey assistant character.

I need to research multiplayer in Unity.

Still need to introduce sound to the game.

Supervisor Meetings

Date of Meeting: none

No supervisor meetings, however I did update my supervisor of my progress via email.

6.3.6 Reflective Journal

Student name: Sean Trant x13332576

Programme (e.g., BSc in Computing): BSc in Computing, Gaming and Multimedia

Month: March - April

My Achievements

This month:

From the period of March 1st to April 11th (time of writing this journal), I completed a very large amount of work.

- I added the finish line to the race level, when the player crosses the finish line I disable the car, disable the gorilla and enable various text fields so the player can see what they scored.
- I completed the level design. I am quite happy with the level design in its current form, it is not perfect, but it I have achieved the high level of detail that I wanted.
- I completed the setup phase level, 5 boxes are randomly spawned between 10 spawn points and the player is given 60 seconds to find them.
- I added 3 soundtracks to the game, they will play on the different levels.
- I used maya to model a driver for the car. This driver is a small monkey, which is fitting seeing as the player is chased by a gorilla. I also animated the monkey's arms and the vehicle's steering wheel so they turn whenever the player uses the turn keys.
- I enhanced the graphics of the game, adding effects like bloom, colour correction and ambient occlusion, this greatly increases the visual appeal of it.
- I added an AI racing car to the game, this turned out to not be overly difficult, to accomplish it I set up a path of nodes with a script of gameobjects. The AI will try to reach the first node, and then move on to the next, it will repeat this until it reaches the last node.

Intended Changes

Any changes from this point will be based on the testing feedback I get from customers and the testing I perform myself. The main task I need to work on now is testing, as this is worth 10% of my grade and is an important part of any project's lifecycle.

Supervisor Meetings

Date of Meeting: none

No supervisor meetings, however I did update my supervisor of my progress via email.