

**Design Document for:**

# OOber Taxi

**Revolutionising the taxi game!**

“We’re not trying to reinvent the wheel with this one”™

All work Copyright ©2018 by Carré

Written by Jack Matters, Cordell Smith

Version # 2.00

Thursday, May 31, 2018Table of Contents

[OOber Taxi 1](#_Toc515620367)

[Design History 5](#_Toc515620368)

[Version 1.00 5](#_Toc515620369)

[Version 1.10 5](#_Toc515620370)

[Version 1.20 5](#_Toc515620371)

[Version 1.30 5](#_Toc515620372)

[Version 2.00 5](#_Toc515620373)

[Game Overview 6](#_Toc515620374)

[Philosophy 6](#_Toc515620375)

[Philosophical point #1 6](#_Toc515620376)

[Philosophical point #2 6](#_Toc515620377)

[Philosophical point #3 6](#_Toc515620378)

[Philosophical point #4 6](#_Toc515620379)

[Common Questions 6](#_Toc515620380)

[What is the game? 6](#_Toc515620381)

[Why create this game? 6](#_Toc515620382)

[Where does the game take place? 6](#_Toc515620383)

[What do I control? 7](#_Toc515620384)

[What is the main focus? 7](#_Toc515620385)

[What’s different? 7](#_Toc515620386)

[Feature Set 8](#_Toc515620387)

[General Features 8](#_Toc515620388)

[Gameplay 8](#_Toc515620389)

[General Features 8](#_Toc515620390)

[Gameplay 8](#_Toc515620391)

[The Game World 9](#_Toc515620392)

[Overview 9](#_Toc515620393)

[World Feature #1 - Randomly Generated Mission Locations 9](#_Toc515620394)

[World Feature #2 - Inhabited by AI 9](#_Toc515620395)

[World Feature #3 - Inaccessible Areas 9](#_Toc515620396)

[World Feature #4 - Randomised Obstacles 9](#_Toc515620397)

[The Physical World 9](#_Toc515620398)

[Overview 9](#_Toc515620399)

[Key Locations 10](#_Toc515620400)

[Travel 10](#_Toc515620401)

[Scale 10](#_Toc515620402)

[Objects 10](#_Toc515620403)

[Time 10](#_Toc515620404)

[Rendering System 11](#_Toc515620405)

[Overview 11](#_Toc515620406)

[2D/3D Rendering 11](#_Toc515620407)

[Camera 11](#_Toc515620408)

[Overview 11](#_Toc515620409)

[Camera Detail #1 11](#_Toc515620410)

[Camera Detail #2 11](#_Toc515620411)

[Camera Detail #3 11](#_Toc515620412)

[Camera Detail #4 11](#_Toc515620413)

[Game Engine 12](#_Toc515620414)

[Overview 12](#_Toc515620415)

[Game Engine Detail #1 - Physics Engine 12](#_Toc515620416)

[Game Engine Detail #2 - AI Engine 12](#_Toc515620417)

[Game Engine Detail #3 - Texture Manager 12](#_Toc515620418)

[Water 13](#_Toc515620419)

[Collision Detection 13](#_Toc515620420)

[Lighting Models 14](#_Toc515620421)

[Overview 14](#_Toc515620422)

[Lighting Model Detail #1 14](#_Toc515620423)

[The World Layout 15](#_Toc515620424)

[Overview 15](#_Toc515620425)

[World Layout Detail #1 - Random Procedurally Generated Cityscape 15](#_Toc515620426)

[World Layout Detail #2 - Mountainous Terrain Border 15](#_Toc515620427)

[Game Characters 17](#_Toc515620428)

[Overview 17](#_Toc515620429)

[AI Pedestrians 17](#_Toc515620430)

[AI Vehicles 17](#_Toc515620431)

[User Interface 18](#_Toc515620432)

[Overview 18](#_Toc515620433)

[Assisted GPS 18](#_Toc515620434)

[Car Info 18](#_Toc515620435)

[Timer 18](#_Toc515620436)

[Minimap 18](#_Toc515620437)

[Musical Scores and Sound Effects 20](#_Toc515620438)

[Overview 20](#_Toc515620439)

[3D Sound 20](#_Toc515620440)

[Sound Design 20](#_Toc515620441)

[Single-Player Game 21](#_Toc515620442)

[Overview 21](#_Toc515620443)

[Single Player Game Detail #1 - Missions 21](#_Toc515620444)

[Single Player Game Detail #2 - Free Roam 21](#_Toc515620445)

[Story 21](#_Toc515620446)

[Hours of Gameplay 21](#_Toc515620447)

[Victory Conditions 22](#_Toc515620448)

[Multiplayer Game 23](#_Toc515620449)

[Overview 23](#_Toc515620450)

[Max Players 23](#_Toc515620451)

[Servers 23](#_Toc515620452)

[Customization 23](#_Toc515620453)

[Internet 23](#_Toc515620454)

[Gaming Sites 23](#_Toc515620455)

[Persistence 23](#_Toc515620456)

[Saving and Loading 23](#_Toc515620457)

[Character Rendering 24](#_Toc515620458)

[Overview 24](#_Toc515620459)

[Character Models 24](#_Toc515620460)

[Character Textures 24](#_Toc515620461)

[Extra Miscellaneous Stuff 25](#_Toc515620462)

[Overview 25](#_Toc515620463)

[Day/Night Cycle 25](#_Toc515620464)

[“Features Implemented/Not Implemented Appendix” 26](#_Toc515620465)

[“Known Bugs Appendix” 28](#_Toc515620466)

[“Special Features Appendix” 29](#_Toc515620467)

[“Objects Appendix” 30](#_Toc515620468)

[“AI Appendix” 31](#_Toc515620469)

# Design History

During the design process of our game engine, the ideas that we intend to bring into fruition will be documented with the intent to allow us to examine the success and failures of our project. Beginning with version 1.00, we will be adding smaller 0.10 increments or even 0.01 indicating minor changes pertinent to the project. When a significant milestone is reached, this will be set during the project planning stage, it will be indicated by intervals of 1.00.

## Version 1.00

Version 1.00 is the initial start on the game design document. An example game document has been used as the foundation for this one, adding/removing/altering the different parts to suit our game design. The parts that were added/altered were done so in another font colour, making it easier for the other members of the team to review what has been added/altered and compare it to what was already there from the example. Once all members have had a chance to see the changes, the example sections will be removed. This will be done at a later date. Many things were changed, and the rest was left untouched until it is decided whether we need to alter or remove them.

## Version 1.10

Version 1.10 consisted of the group coming together to work on the current Game Design Document. Things changed include:

* Editing sections that were added in the previous version, and completing them.
* Completing other sections that had no information given.
* Removing sections that don’t apply to our game (Level Editor, etc).

## Version 1.20

Version 1.20 is the final version before our first submission. The next version will most likely be after we have started on our game engine, and have more information on what we have done and how we have done it. Changes in this version include

* Completing all relevant sections
* Reviewing existing sections, and doing minor changes to wording
* Adding a note to sections that may/will be added in future versions of this document
* Final read through of entire document

## Version 1.30

Version 1.30 is the first alteration to this document for submission of Assignment 1. Changes in this version include

* Added a ‘Features implemented/not implemented’ appendix
* Added a ‘Known bugs’ appendix
* Added a ‘Special features’ appendix
* Changes to the ‘Game Engine’ section of document

## 

## Version 2.00

Version 2.00 is the final version for our second submission. All relevant parts of this document have been updated to the changes made since version 1.30.

# Game Overview

## Philosophy

### Philosophical point #1

This game is based on an idea that has been on the market for quite some time. It isn’t attempting to achieve anything new or innovative, but rather to take ideas from several already existing games and to incorporate them into our own game.

### Philosophical point #2

Our game will run on a game engine that we build ourselves. For this reason, the game won’t be anything fancy, but rather something that demonstrates the effective use of the different components of our game engine. The sole purpose of our game is show the functionality of the game engine that we have designed and built ourselves.

### Philosophical point #3

We wish to create a successful game engine using the creational patterns of Factory Method, Abstract Factory, Prototype and Singleton as well as using the structural patterns of Facade and Flyweight.

### Philosophical point #4

We wish to make our game engine portable enough to work on many systems and not just high end devices. This way if we are successful in our end goal, we can have a lightweight engine that can be reused in further projects or even used as examples for future endeavours (portfolio use).

## Common Questions

### What is the game?

The game is based on the chaotic life of a an OOber driver who is trying to obtain a 5 star rating. The player will have a 3rd person PoV that controls the vehicle they drive. The player must pick up and deliver passengers to their desired location in a timely and safe manner. The game will follow a storyline, so until the player has received a satisfactory score for each mission, they will be unable to proceed and must retry the current mission. The mission score will be based on the average of all customer’s satisfaction levels, which will be determined by a set algorithm combining the overall happiness values of the successful or unsuccessful customer fares. Obstacles will be generated to add a degree of difficulty which may cause you to deviate from your objective or reduce the overall satisfaction of the ride which you as a skilled driver must find a way to overcome.

### Why create this game?

We have decided to create this game because we thought the idea of a 3rd person driving game would be something new to develop as none of us had done so in the past (mostly 1st person PoV). This genre of game will allow us to challenge ourselves as well as explore different concepts to do with camera work, following the model car that is the main character of the game (the OOber Taxi car) which allows the player to develop sentimental value for as it will be predominantly what they will see throughout the play through. In conjunction with the task of creating a robust, reusable game engine, this game will be the showcase of our project.

### Where does the game take place?

This game takes place in a randomly generated city that is roughly based off of the city of Perth. The size of the city will roughly be one square kilometre, allowing a big enough area for the player to play in. There will be other cars and pedestrians on the streets and sidewalks, controlled by AI.

### What do I control?

The player will have control over the main character while they are behind the wheel. While behind the wheel, they will be driving a car and will be subject to forces such as acceleration, braking, and turning. The player will have access to a menu that allows them to accept their next mission/request.

### What is the main focus?

The main focus of this game is to transport the customer to their desired location in a timely and safe manner. The faster the player delivers the customer to their location, the higher that customer’s satisfaction rating will be. The satisfaction rating will decrease the longer the player takes to deliver them, and for causing accidents by hitting other cars, hitting buildings or pedestrians, or by any other means of causing an accident.

### What’s different?

This game is not original as what we are creating isn't something new, and has been inspired by a beloved game called “Crazy Taxi”. However, what we do add ourselves is the entertainment of a scenario that creates the experience of driving a car fast through a busy city in order to deliver your customer and earn a 5 star OOber rating. This kind of realistic entertainment that includes challenges such as roadblocks and avoiding pedestrians is thrilling and fun, as it is something that most people will never experience.

# Feature Set

## General Features

Large world

Driving game

3D graphics

Randomly generated cityscape

3rd person POV

## Gameplay

Driving a car

AI controlled vehicles and pedestrians

Customer satisfaction score

Visible damage on vehicle

Audio from other vehicles and pedestrians

Assisted GPS showing direction (not path) to customer drop off point

Story driven experience

Free roam feature

Minimap feature  
Menu for accepting missions/requests

v2.0

Following the previous releases. The Carre game engine now has a more updated list of general features and gameplay features that are noted below.

## General Features

Driving game

3D graphics

3rd person camera

Static and dynamic collision

Camera and player movement

## Gameplay

Driving a car

AI controlled pedestrians

Rocks player can collide with

Pedestrians player can collide with

Free roam

# The Game World

## Overview

The game world is a randomly generated cityscape that contains various high rise buildings and smaller buildings (ideally we would like to replicate the city of Perth as one level). The size of the city will roughly be one square kilometre, allowing a big enough area for the player to maneuver around the world without feeling like they are restricted. This cityscape area will be surrounded by a varying mountainous terrain however it will not be accessible (solely for visual aesthetic). There will be other cars and pedestrians on the streets and sidewalks, that will have simple AI pathing to replicate a realistic pattern of movement.

v2.0

Due to time constraints, we were unable to implement a randomly generated world and a cityscape, and there are no inaccessible areas and randomised objects. No mission locations were implemented either due to not having the time available to implement a game story. We were able to implement simple AI using a pedestrian model, that moves between waypoints, and to place obstacles/objects on the loaded terrain.

## World Feature #1 - Randomly Generated Mission Locations

The world will be randomly generated for each customer that you pick up. This ensures a new experience for each customer that the player must deliver, and creates a new visual for the player each time. This increases the replayability of the game as it will be a new experience each time.

## World Feature #2 - Inhabited by AI

The world will be inhabited by AI in the form of vehicles and pedestrians. Both forms of AI are included to create a realism to driving in a central business district. The player must be aware of the other vehicles on the road, as well as any AI walking along the sidewalks or crossing the street.

## World Feature #3 - Inaccessible Areas

The game world will also include inaccessible areas outside of the city landscape. These areas will include bodies of water (an ocean on one side of the world, or a small lake), as well as distant mountains. These areas are there to create a realism to the scenery in which the player will see when they are near the edge of the city, instead of empty distant fields. These areas will be inaccessible as they are only there for the visual effect, and driving out of the city is not a feature of the game.

## World Feature #4 - Randomised Obstacles

Road blockages such as car crashes or road works will be randomly generated throughout the map to keep the player from venturing too far from the objective as well as posing as a challenge to add a level of difficulty to the gameplay.

## The Physical World

### Overview

The physical world will include buildings ranging from small houses to big multistorey business buildings. The world will be generated in a way that there is an order to the buildings that are near each other, so you don’t see a single story house being placed next to a thirty storey building. The AI pedestrians will be based off of 2-4 different models, and the AI vehicles will be using 2-3 different models.

v2.0

Due to time constraints, the physical world is pretty bare. We were able to get a player model, a rock model, and a pedestrian model loaded in, but were not able to create a city landscape, have a day/night cycle, and the scale of the pedestrians compared to the player is not correct.

### 

### 

### Key Locations

Each randomly generated game world will have a CBD area, as well as general housing neighbourhoods. The location of these area will vary with each game, and sometimes may not be included at all. Other key locations include areas such as parks, gardens, and shopping districts.

### Travel

The player travels by car. They will have control over the car, and so they will have control over accelerating, braking, steering, and other motions involved in driving and handling a car. There is no plan to add a fast travel system, as this defeats the purpose of the game.

### Scale

The scale will roughly be 1:1 with the real world. This creates realism in the game, so that vehicles aren’t bigger than houses, and pedestrians don’t look like they wouldn’t fit in a vehicle.

### Objects

The different objects found in the world will include things such as the different road signs (stop sign, give way sign), light and power poles (street lights and sidewalk lights), houses and other buildings, the other vehicles (driving around the city, parked, or car crashes/accidents), walls and fences (walls around a park, fences in front of houses in residential neighbourhoods), and most any other object you see while driving through a city. Each object is there to create the scene, but will also become objects that the player could hit. We are yet to decide if some objects, such as fences and signs, will be destructible or not. This would create a more realistic experience, but we may not have the time to do this.

v2.0

Due to time constraints, the number of objects found in the world is significantly lower than what we would have liked. We only had a few to demonstrate the capabilities of our game engine, not to create a visual world.

*See the “Objects Appendix” for a list of all the objects found in the world.*

### Time

Given enough time (excuse the pun), we intend to have a day/night cycle. The speed at which time passes in game is yet to be decided. If time does not permit, then there will be no time in-game. Instead, it will just be instances of day/night that last until the world is randomly generated again.

## Rendering System

### Overview

The rendering system will follow an architecture that adopts the fat interface approach. This is such that in addition to the core services that are apart of the interface, it also offers a rich set of services that satisfy the common needs of the rendering system. Entities will be created from an abstract base class that allows for Update(), Render() and Draw() calls to be made. The Rendering System will have full control over the interfaces that handle any and all entities.

### 2D/3D Rendering

An abstract class that will comprise of behind the scenes OpenGL code utilising 3D draw calls.

## Camera

### Overview

The camera will be set in a third-person point-of-view. Multiple distances from the players vehicle may be incorporated, to allow the player to pick a distance that suits them. The camera may also change its position and/or orientation depending on the different scenarios the player vehicle is placed in. A third-person camera would make our driving game less realistic, but it would make gameplay simpler for the player.

v2.0

In version 2.0, we implemented a third person camera that follows the player as they move. The look direction is changed manually using the left/right mouse button. The camera distance is able to be changed by using the mouse scroll wheel. We were unable to implement any other camera features due to time constraints.

### Camera Detail #1

The camera will in most scenarios be behind the player vehicle, creating a third-person point-of-view. The camera will be far enough back to provide the player with a decent view of the game world, but not so far back that the player vehicle seems ‘too far away’.

### 

### Camera Detail #2

The camera will ‘swing out’ when turning a corner. This will allow the player to see what is around the corner briefly before turning. This feature will make it easier to navigate around the city and to see what obstacles there are around a corner, instead of having the camera always directly behind the vehicle.

### Camera Detail #3

If the camera will pass through an object/building/model, it will instead start to move above the player vehicle. It will continue to move above the vehicle until it is directly above it, at which point it will stop. This situation will happen mostly when the player is reversing, and gets too close to a building. Once the player starts driving forward again, the camera will return to its position behind the vehicle.

### Camera Detail #4

If the player wishes to see what is behind their vehicle, the camera will move from behind the vehicle to a position in front of it. It will retain the same distance from the vehicle that it previously had, but will now instead show what is behind the player vehicle. This camera orientation will be used when the player presses the corresponding key/button for a rear-view-camera.

## Game Engine

### 

### Overview

The Carré Game Engine will comprise of various components that will be coupled together to create a robust and reusable system for the purpose of entertainment. The components that will be included are the 3D Graphics Engine, Game system, Input system, Animation system, Collision Detection and Navigation System, Physics Engine, AI System, 3D Model and Imagery Manager, Audio System, Game World Engine and Game Control Engine. Throughout the design phase of the engine, we will look to add layers of abstraction to various components allowing for ease when switching libraries such as OpenGL and PhysX.

v2.0

Due to time constraints, we were unable to implement an Animation system, Navigation system, and Audio system.

### Game Engine Detail #1 - Physics Engine

The physics engine is going to use the 3rd party software ‘Bullet Physics API’ to handle all physics calculations. This is easier than creating our own physics engine from the ground up as we do not currently have the knowledge nor the time to complete this task. ‘Bullet Physics’ is a well known physics engine that has been around for several years. We will not be using every feature that it contains, but only those that we need. ‘Bullet Physics’ allows the creation of many different types of rigid bodies from simple shapes, such as boxes and spheres, to more complex shapes such as height fields and convex meshes. It also allows the creation of dynamic and rigid bodies, and the addition of gravity as an acting force that is able to be altered. This API has everything we need, so we will be using it.

### Game Engine Detail #2 - AI Engine

v2.0

The AI engine is split up into AllStatesFSM, ComputerAI, State, and StateMachine classes. The StateMachine class handles the transitions between the different states, and keeping track of the current and previous states. The State class is the template virtual class defining the Enter, Exit, and Execute sections of each state. The AllStatesFSM class handles the implementation of the Enter, Exit, and Execute sections for each different state. The ComputerAI class handles all the data related to the AI. These 4 classes all work together to create the AI engine.

### Game Engine Detail #3 - Texture Manager

v2.0

The Texture Manager class is responsible for the loading of all textures in the engine, and storing them in a meaningful way for access during initialization and game play. All textures are stored externally, with their relative file paths stored in a lua script. The script is used to load the textures into the Texture Manager, where they are given a textureID. The loaded textures can be accessed where they are needed, as the Texture Manager uses a singleton design. When accessing the Texture Manager, it will always check to see if the requested texture is loaded first. If it is, it will return the correct textureID, otherwise it will load the texture into the Texture Manager then return the textureID. This prevents the loading of a texture multiple times.

### Water

*Not implemented.*

### Collision Detection

Collision detection is done in each simulation step, and is handled by the ‘Bullet Physics API’. All forces and collisions are calculated, after which the resulting location of each game objects rigid body is passed back for drawing. In the case of non player controlled game objects, they are all under the effect of gravity and any other forces acting upon them. In the case of the player controlled game object, a force is applied in the direction that the object/camera is moved via input from the player and then cancelled out after all physics calculations have been completed to prevent unwanted movement.

## 

## Lighting Models

### Overview

Lighting will be implemented within our game engine as we look to achieve a pleasant and somewhat realistic world environment for the user to navigate through. The main lighting model used will be the Phong reflection model (or Phong illumination).

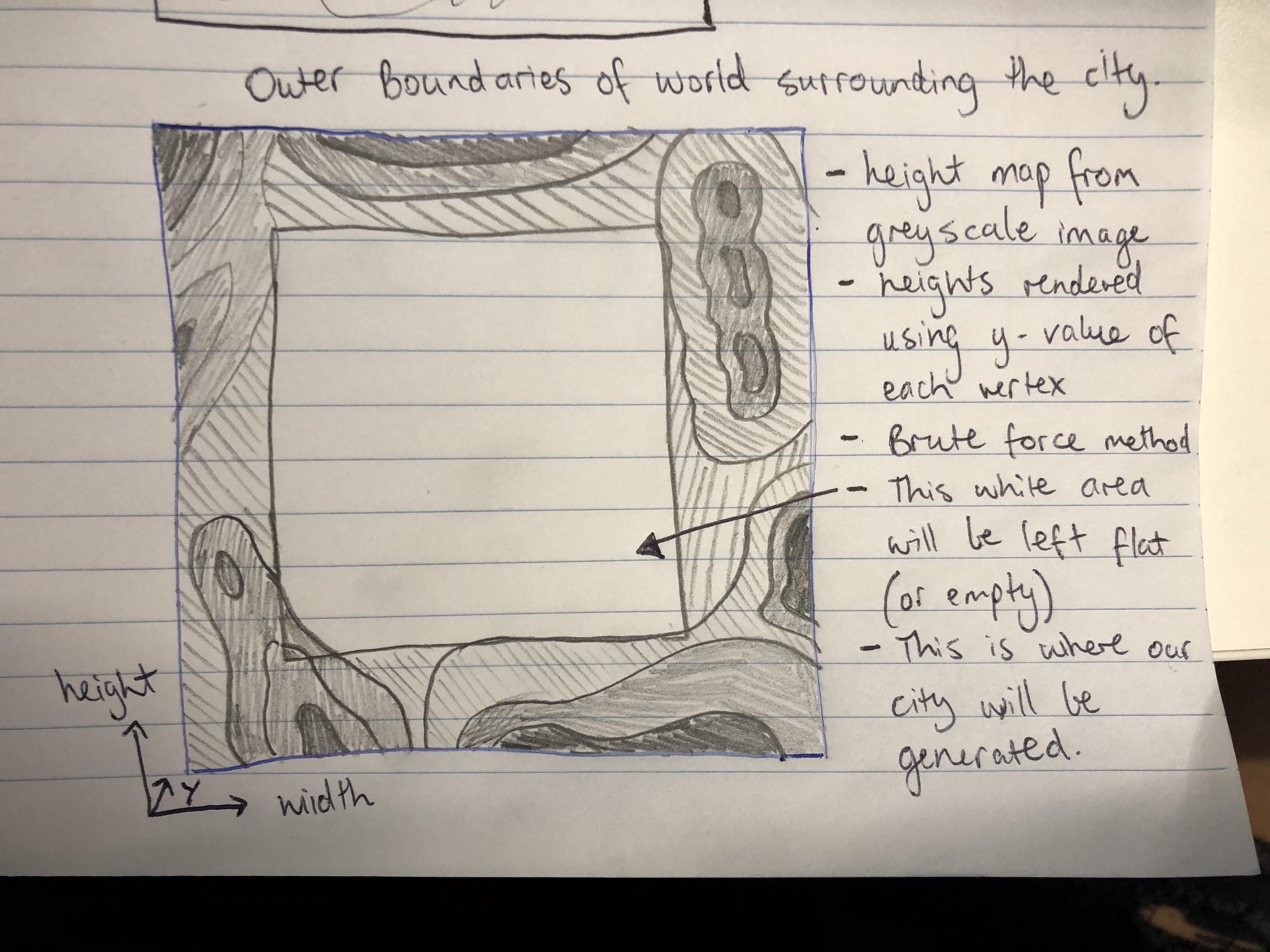
### Lighting Model Detail #1

Phong Reflection: This technique is an empirical model which consists of local illumination points on a surface. Using a combination of diffuse reflection from for rough surfaces and specular for shiny surfaces with small, intense specular highlights and ambient light to account for the small amounts of light scattered about the entire scene. These combination of model details will be used to give our models and environment a more appealing look.

# 

# The World Layout

## Overview



## World Layout Detail #1 - Random Procedurally Generated Cityscape

The cityscape that the player will be allowed to explore (the white square) will be procedurally generated and textured to allow for the creation of simple square high rise and low level buildings. Each time the world is re-created, the streets and building placement will always be different to what it was prior.

v2.0

The terrain is generated using a heightmap. The heightmap is a custom .raw image file that is read in as bytes and stored in a data structure. Each corresponding pixel represents a scale from 0-255 of black and white. This is then passed to a bruteforce class that reads in each data segment and creates a terrain block with that height value, white being the highest and black being the lowest. The bruteforce class then generates a mesh of the terrain that can be used in the game engine.

## World Layout Detail #2 - Mountainous Terrain Border

Surrounding the playable area will be a brute force generated terrain (topographic map around white square) This will be used to demonstrate the use of terrain and add aesthetics to our world.

v2.0

The custom .raw image created has a mountainous surrounding with a flat square in the center to where the city is to be generated.

**World Layout Detail #3 - Skybox**

To simulate a never ending sky, we will be using a skybox depending on the time of day and weather. Whether the skybox changes during play, or is the same for the entirety of an instance is dependant on if we have the time to create a day/night cycle.

v2.0

We failed to implement a skybox feature in version 2.0 as we did not have the time and manpower to do so.

**World Layout Detail #4 - Textures**

v2.0

Each terrain contains a corresponding texture that is applied to each cube that is generated that makes up the entire mesh. When creating the terrain object, you also specify the texture you wish to be applied to it and it gets stored in the terrain object along with the rest of its data. When the rendering occurs, the texture is applied to the terrain.

# Game Characters

## Overview

The main player model in the game is a vehicle. Depending on the health status of the vehicle it will have different models applied to it, the more damage the vehicle takes will be reflected with damaged vehicle models.

v2.0

The main character of the game is a taxi vehicle. It has the ability to be controlled by the player with a 3rd person camera that follows it. The character can collide with objects using the physics engine as well being able to follow the mountainous terrain showing a slightly realistic driving experience.

## AI Pedestrians

The AI pedestrians will be based off of 2-4 different models. They will use a path finding system to simulate pedestrians walking along the sidewalk and crossing the road at intersections. They will react to the player vehicle if it gets too close, by moving/jumping out of the way. There will be no situations where it is possible to hit a pedestrian.

v2.0

The AI pedestrians of the game move between set waypoints, choosing a waypoint at random. They are all of the same model. They have no collision and do not follow the terrain, but they will continue to move between the different waypoints, showing a slightly realistic pedestrian behaviour.

## AI Vehicles

The AI vehicles will be based off of 2-3 different models. Similarly to the player vehicle, there will be different models applied for the current health status of the AI vehicle They will use a path finding system to simulate vehicles driving in the city, obeying laws such as stop lights and driving on the left.

v2.0

AI vehicles were not added to version 2.0 due to time constraints.

# 

# User Interface

## Overview

The user interface will have a minimap situated in the top right corner of the screen, as well as an arrow pointing in the direction the player must go (assisted GPS), which will be in the top centre of the screen. In the bottom left of the screen will be information regarding the car such as current speed, current damage to vehicle, and other important information. There will be a timer in the top left of the screen showing how long it has currently taken to transport the passenger, or how long is left to transport them.

v2.0

Due to time constraints and manpower, we elected not to implement a user interface and instead work on other more important components.

## Assisted GPS

The main directional component will be a directional arrow situated at the top of the screen in the center. This will give the player an indication of what direction they need to be heading in order to pick up their next customer or drop them off but will not give them a specific route, allowing for dynamic play in the procedurally generated world.

## Car Info

In the bottom left of the screen will be information regarding the car such as current speed, damage to the vehicle, and other important information.

## Timer

A different timer will be shown in the top left depending on what state the player is in.

If the player has a passenger in the car, the timer will display the duration of the trip to transport the passenger and the maximum amount of time available before the passenger decides to exit the vehicle. If the player is on their way to a fare, the timer will display how much time they have to pick up the next passenger before they cancel and order another driver.

v2.0

The timer has not been implemented in current release.

## Minimap

The minimap will show the surrounding area to the player. This will allow the player to get an idea of what streets are coming up that they may turn down, but won’t be large enough where they can plot their entire course to the pick up/drop off location.

v2.0

The minimap has not been implemented in current release.

# 

# 

# 

# Musical Scores and Sound Effects

### 

## Overview

Audio samples will be taken from royalty free audio resources such as Freesound.org.

This includes such effects as the car engine, collision sounds, tyre screeching, miscellaneous ambient sound as well as a potential music selection from a car radio.

v2.0

Due to time constraints, this feature was not implemented.

## 3D Sound

*Not implemented.*

## Sound Design

*Not implemented.*

# Single-Player Game

## Overview

The single-player game experience involves picking up customers and delivering them to their desired location. The player may drive how they want, but too much damage to the vehicle will render it broken (or failure on the customer satisfaction), and taking too long will result in a low customer satisfaction rating. Apart from the story, the player will be able to freely drive around the randomly generated world. There will be an option in the menu to create a new world if the player wishes.

Describe the single-player game experience in a few sentences.

v2.0

Due to time constraints, we were unable to implement a story or missions to the game. Free roam is included.

## Single Player Game Detail #1 - Missions

Initial Mission: The player will be required to pick up their first customer from a specific destination that is easily identified by the objective indicator (big arrow). When the player picks up the customer, they will then be required to take them to their destination. There will be some minor obstacles that can’t be avoided to teach the player that the car can take damage which affects the happiness of the customer. When the destination is reached, the customer will give their rating according to factors of the ride such as time and comfort. The initial mission is used to teach the player about what the ultimate goal is, and how to successfully win or lose.

Game Start: The game will start with the player having to complete a series of successful pick up and deliveries in an allotted time (will be later defined). After each successful drop off, a rating will be given and the average of these ratings over the mission time will be what your final score will be and added to your OOber rating. When the timer is complete, and you successfully complete a number of pickups and drop offs, the next level will begin with a new city generated.

## Single Player Game Detail #2 - Free Roam

Free roam will allow the player to explore the randomly generated world. This mode is mostly for if the player just wants a relaxing experience of driving around a city, looking at all the visuals and the bustling city experience. There may or may not be the option for the player to complete random pickups and dropoffs. If this is added, the score for each successful transport will have no bearing on the players score in the main story.

## Story

The character will be tasked with picking up and transporting customers in order to achieve a 5 star rating for their OOber score. The story will allow the player to increase their overall star rating by providing them with customers, or missions, that they can attempt. There is two directions we could take with this. The first is that to keep to the story, the player will be unable to move onto the next mission until they achieve a satisfactory score on the current mission. This will be enforced so that when the story is completed, the player will have achieved the 5 star rating they were after. The second way is to have each mission accessible for completion to the player, but until they achieve the satisfactory score in each mission, they will not achieve their 5 star rating. Each mission will have its own small story, that will add to the overall story. The number of missions is yet to be decided.

## Hours of Gameplay

*Not implemented.*

## Victory Conditions

The victory condition for each mission is to achieve a satisfactory customer score. The mission score will be based on the average of all customer’s satisfaction levels, which will be determined by a set algorithm combining the overall happiness values of the successful or unsuccessful customer fares. The actual formula used is yet to be decided.

# 

# Multiplayer Game

## Overview

For version 1.xx, we don’t intend to have multiplayer functionalities, however this may be a feature that may be included in future versions of the game engine. In this case, the game design document will be updated to include all information about this feature.

v2.0

This feature was not implemented in version 2.0.

## Max Players

*Not implemented.*

## 

## Servers

*Not implemented.*.

## Customization

*Not implemented.*

## Internet

*Not implemented.*

## Gaming Sites

*Not implemented.*

## Persistence

*Not implemented.*

## Saving and Loading

*Not implemented.*

# Character Rendering

## Overview

We plan to render the player vehicle using a model loader and following a rendering pipeline. The models used will be one of a certain model type (.obj, .fbx, .3ds, .max, .blend), and there will be different models for the different state of the player vehicle (how damaged it is).

v2.0

The Carre game engine uses a third party library called Assimp Open Asset Import Library. This library has the capability to recognise many different file types and read the data in using what they call aiScenes. A scene may contain one or many objects that contain one or many meshes with their respective data attached and they are all linked through a linked list data structure. The extensive library allows us to read the character model in through its relative path, then to be stored in a data structure.

## Character Models

v2.0

The characters model is loaded and stored in a data structure that contains all the mesh data, vertex data and material information. Using modern openGL shaders, each character model has its own VAO containing all the necessary data in a VBO. The VAO is then enabled uses a universal Default.shader to render the models to the screen. The .shader file contains both the vertex and fragment shader C code for the GPU to use.

## Character Textures

v2.0

The characters textures are loaded in using stbi\_image. This is an image loading library that is contained in a single header file allowing us to read in multiple types of images. When loading in the characters data, you must also define what texture it uses. The texture is read in as a relative file path and the data is stored in a Texture member variable for the model of the character. When the character is rendered, the texture is applied to it using modern openGL shaders with the corresponding UV mapping to make the texture appear correctly.

# Extra Miscellaneous Stuff

## Overview

This section contains miscellaneous stuff that we plan to include in the game, but does not fit in any other section of this document.

## Day/Night Cycle

Given enough time available we intend to add a day and night cycle with a rotating directional light around the game world. When the directional light reaches a certain point the skybox will gradually switch texture and street lights will be faded in to create the effect of a night time scene.

v2.0

This feature was not implemented due to time constraints.

# 

# 

# “Features Implemented/Not Implemented Appendix”

This appendix provides a full list of every feature that is implemented in our game, and every feature that is not implemented.

Implemented features:

* 3D game in first person perspective (will change to third person for assignment 2)
* Camera is able to move forward and back (but not strafe left or right)
* Moving mouse changes the camera view direction
* Displaying of wireframe and solid view of terrain
* Objects that camera can collide with
* Collision detection for camera and simple shapes (spheres, box shapes).
* 15 NPC’s (cars with no textures)
* Doxygen comments

Non-implemented features:

* Storing of game data in a data structure
* Game data stored in external files and loaded by script
* Game objects not implemented from base class called GameObject
* Physics Engine not behind a facade (bound to Bullet Physics API)
* Displaying a heightfield terrain
* Multi-texturing, detail map, and lighting for terrain
* Camera doesn’t move across terrain (terrain not implemented)
* Splash screen with group members and their names
* Strafing left and right (we decided not to implement this as we are making a driving game, and strafing wouldn’t make sense. We will add steering instead to Assignment 2)

v2.0

Following the previous releases. The Carre game engine now has a more extensive working list of features that are noted below as well as any non-implemented features.

Implemented features:

* Third person camera that follows the player
* Player able to move forward and back, and turn side to side using the ‘WASD’ keys
* Camera angle/direction able to be changed using mouse by holding the mouse1 button down for angle around player and mouse2 for pitch
* Camera can zoom in to player by using the mouse scroll wheel
* Displaying of wireframe and solid view of models using the ‘Q’ and ‘E’ keys
* Static objects that player can collide with
* Dynamic objects that player can collide with
* Static and dynamic collision for models and player
* 20 NPC’s
* Storing of game data in data structures
* Game data stored in external files and read from script
* Loading and generating of a terrain
* Player and AI follow terrain height
* AI pedestrians (simple AI)
* Input engine
* Graphics engine
* Texture manager
* AI engine
* The terrain generated is (128, 000 x 128, 000) units with 10 units = 1m it is 12800m x 12800m
* Camera zoom feature using mouse wheel
* Doxygen comments

Please Note: A number of these features may have been a requirement for Assignment 2 and due to being a 2 person group, we were unable to implement some of these features.

Non-implemented features:

* Animation Engine
* Menu system and UI
* AI not loaded by script
* FSM not loaded by script
* Water feature
* Multi-texturing, detail map, and lighting for terrain
* Splash screen showing group members and their name
* Saving and loading of games
* Characters can be organic or inorganic or a combination of both
* Game story
* Audio Engine
* Day/Night Cycle
* AI vehicles
* City terrain/landscape
* AI communicating amongst each other
* Inaccessible areas
* Randomly generated terrain
* Support for multiple window types
* ‘Smart’ AI (AI only move between waypoints)
* Rotation of AI based of movement direction
* Skybox
* No visible ending
* Player can not die/fail
* Rotation of player to simulate driving up a hill

# “Known Bugs Appendix”

This appendix provides a full list of every known bug in our game, how to recreate them, and possible solutions.

Known Bugs:

* AI pedestrian model uses same height value as dynamic pedestrian model. If dynamic model moves up a hill, the AI matches its height, giving it the effect of floating in the air
  + To recreate, just push a dynamic model up a hill
  + Possible solution is alter the code so each models uses it own height value
* Pedestrians are dynamic and move when collided with.
  + To recreate, move into a pedestrian not controlled by AI
  + Solution is to change the collision on the pedestrian so it doesn’t move when collided with
* Camera moves through terrain when changing angle
  + Move camera angle down, and it will go through terrain
  + Possible solution is to limit the angle to what the camera can be moved, so it can’t pass through the terrain

# “Special Features Appendix”

This appendix provides a full list of every special feature in our game.

Special Features:

* We do not currently have any special features. We were more concerned on completing as many of the assignment requirements as we could instead of focusing on making any special feature. For assignment 2, we will endeavour to create some to showcase our game.

v2.0

For version 2.0, we believe that we don’t have any special features. Due to limits in the time remaining and manpower available, we elected not to spend time implementing any special features. We decided to spend our limited time implementing the core features of our game engine. Of all the features we were able to implement, we do not believe that any of them are special, rather they are standard features to any game engine, and therefore this section is left blank.

# 

# 

# “Objects Appendix”

This appendix provides a full list of every object our game contains.

Animate objects:

* Taxi (Player)
* Knight pedestrian

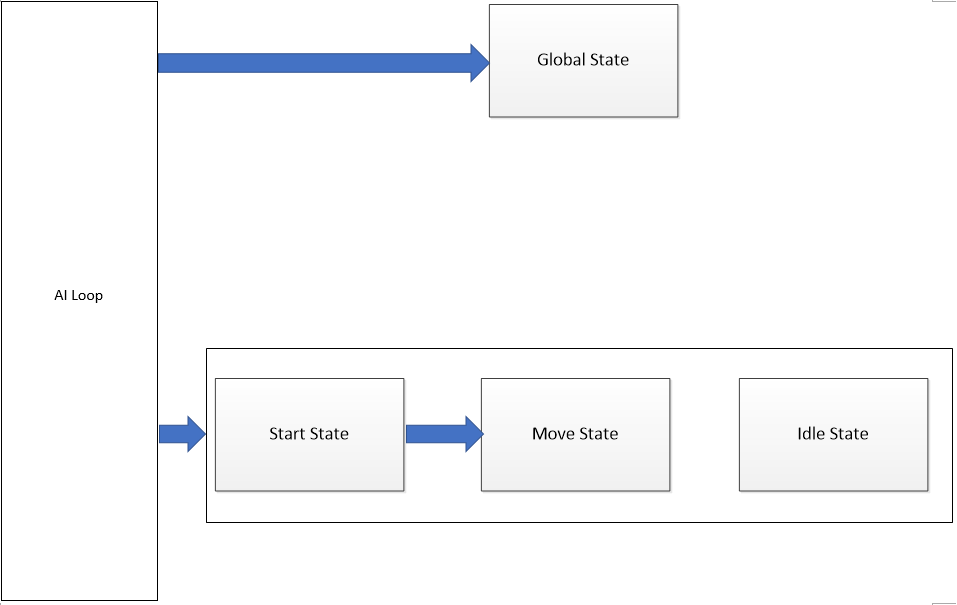
Inanimate objects:

* Rock

# 

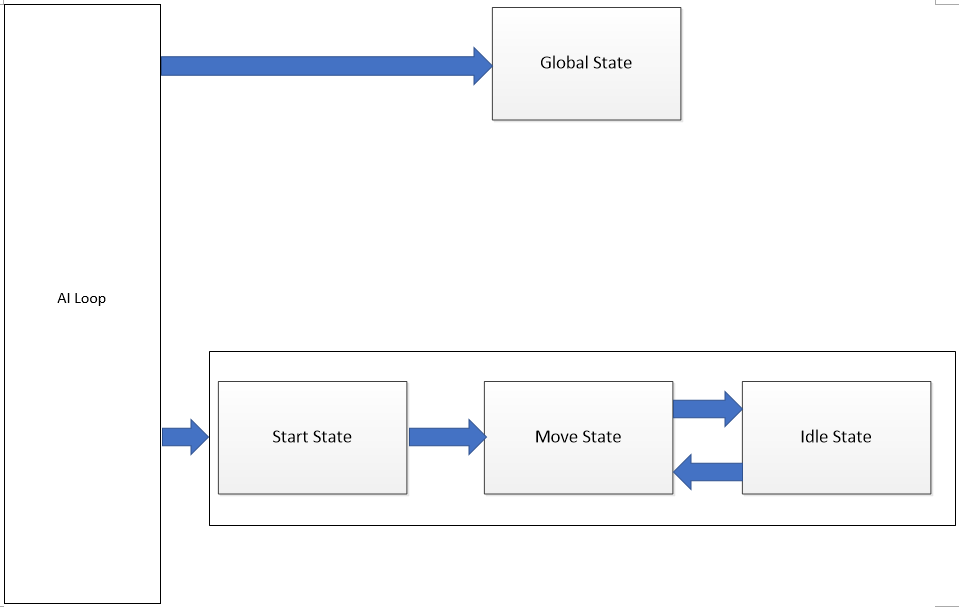
# “AI Appendix”

This appendix provides a diagram of the AI state transitions, and explains what happens where.



*Diagram 1 - Final Demo AI State Transitions*

This is what the AI state transitions ended up being for version 2.0.



*Diagram 2 - Final Demo Planned AI State Transitions*

This was the planned state transitions for version 2.0 if time permitted.

Global State:

Global state would be checked each AI loop. For the final version, it is empty. For the planned version, it would check if it had collided with anything. If so, it would pause its current state, move back from what it had collided with, and try another path. Once it completed that task, it would return to its previous state.

Start state:

Start state is where the AI starts. It sets all the initializations, then transitions into the move state. This state will not be entered again.

Move state:

Move state is where the AI picks a random waypoint and proceeds to move towards it. Once the waypoint is reached, it would pick another waypoint to travel to. In version 2.0, this state is never exited. For the planned version, it would move to the Idle State if stopping at a traffic light or crosswalk.

Idle State:

Idle state is where the AI is not moving. It is waiting for a condition to be met to continue moving. If version 2.0, this state is not implemented. In the planned version, this state would be entered when a vehicle is waiting at the traffic lights, or pedestrian waiting at a crosswalk. This state would be exited on the green signal.