horizontal line

Grand Traffic Auto

Technical Specifications

**14th April 2019**

# 

# 

# 

# 

# 

# Project Team

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| Robert Pennefather | 21511164 |
| Daniel Gonzalez | 22024722 |
| Martin Porebski | 21498791 |
| Jacqueline Soon | 21719676 |

# 1.0 Overview

In the following sections the selection of the tools and strategies will be detailed, clarifying the use of the Unity engine as framework, architecture of the system, data handling, in-game choices, testing method, deployment and documentation for future developers.

# 2.0 System Design

## 2.1 Game Engine

The team has decided to build the game on the Unity engine. This is because using an existing game engine will reduce the development time. It will also aid with the overall design and allow for a clear and flexible structuring of the project. In particular, game engines provide significant support and a large number of tools tailored to building reliable games. They also assist in learning the specifics of game-development as the team has no experience in this field. Using Unity will allow the team to conform to industry development standards for the engine, which will assist any future developers. This is consistent with the requirements outlined in section 5.0 of the functional specifications.

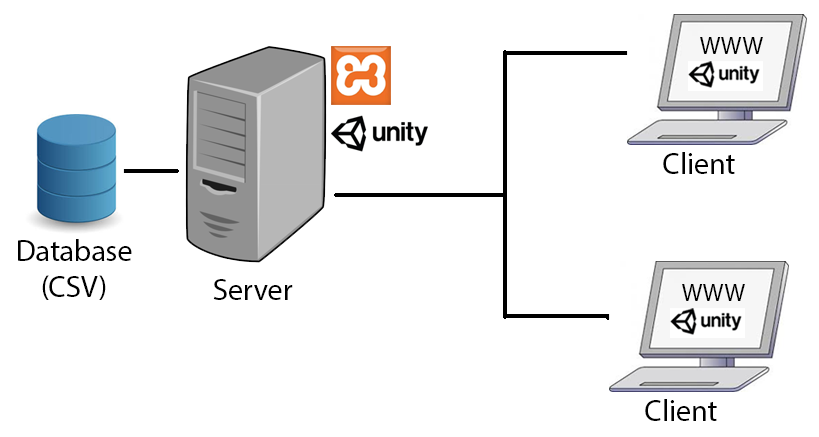
The team narrowed down the options for the game engines to: Unity and Unreal Engine. Both of these options were selected as they are made to build 3D games each having a large community; with tutorials, discussion boards, and question forums if the team needs support. Unity was selected as it is an open source framework which is tailored specifically to making higher-level, small to medium size, 3D games. Unreal provides a more complex framework that allows for more lower-level operations. However, this makes it more complicated and time-consuming to use as it is more suitable for triple A games that require complex features and physics. Furthermore, Unity uses the Mono framework which runs C# code, which the team is more comfortable with, as opposed to Unreal’s C++.

Unity is cross-platform which will allow the team to deploy to a wide range of client machines including the most popular: Windows, Linux and macOS. Additionally, Unity offers deployment into a browser application (Javascript powered WebGL) which the team plans to utilize, if the performance of the game is not impacted.

## 

## 2**.2 Architecture**

The intended overall architecture will consist of a server machine and multiple clients connecting to it. This way, the server will be in charge of the framework and can coordinate the user machines. It will set up the desired scenarios and collect the survey data from players. This server can be hosted on AWS virtual instance or given the opportunity the system can be deployed on the virtual servers available at UWA, which that would reduce network latency and privacy concerns associated with the user data.



The server will host a simple website that will gather the required survey information and direct users to a game instance, as previously configured by the admin using a configuration file. Considered technologies involved:

* Nginx web server with ASP.NET on .Net Core (C#)
* Nginx web server with ReactJS and/or NodeJS (JavaScript)
* XAMPP web server with CodeIgniter framework (PHP)

However, due to the simplicity of the requirement, the XAMPP-PHP-Bootstrap framework was selected as it will allow for a quick development of a form application. The website will gather the data from the survey and assign players to the available sessions.

## 2.3 Data Storage

A database is required to gather and store data from each experiment (game run), such that it can be queried later by the researchers. The team has considered both a relation SQL and a non-relation Mongo database. As the client is more used to relation-like “rows” and because it will be a consistent structure, there is no need for a flexible database and team will be using a relational SQL database.

For the prototype it has been determined that storing the data using a CSV file is sufficient. This is because the implementation will be quicker and it can be used to simulate a relational database without requiring to setup an SQL server and communication system.

The configuration files to create custom sessions will be stored as JSON files. This is because the data can be stored without structure and it will be simple to edit values.

## 2.4 Game Physics

Unity has a built in physics engine, the rigidbody component applies physics to an object and can allow for gravity, friction, drag, collisions with other models and assortment of other useful features. Using a physics-based system will allow the player to steer, accelerate and brake the vehicles in a natural way. The game will also provide manually coded feedback to the user through the rotation of the wheels and steering wheels when turning as well as sounds to indicate acceleration and braking.

Since the game is made to simulate a real environment the team wants to stop vehicles before collisions with other objects. The team believes it is important to include invisible barriers and a larger invisible collision box to detect potential collisions and stop them before the assets visibly collide for the player.

# 3.0 Testing Strategy

The code will be tested with one of the tools Unity offers which is the Unity Test Runner, it uses the open-source unit testing library NUnit and allows tests to be run in Edit and Play Mode.

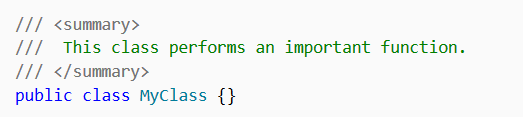
Tests in Edit Mode need to focus on the functionality of the survey, since it runs directly in the editor and does not invokes Monobehaviour methods Awake and Start, which are called after objects have been initialized and let them pass information between them. Play Mode tests will be required for in-game interactions.

The Unity Test Runner does have limitations regarding WebGL, so tests will have to be written keeping in mind that the UnityTestAttribute is not supported. This type of unit test allows to yield from test so that frames are skipped when the test runs.

# 4.0 Documentation

The code behind the project will be thoroughly commented to increase the software quality. This is because it will aid when writing tests (to determine desired functionality) and allow easier collaboration between the team members and any future developers.

Hence, the team will be able to take advantage of a documentation generator. The C# .Net framework offers the ability to automatically create documentation files from comments at compile time. This means that the team will have to strictly follow the syntax and the commenting practices that are specific to this framework. In particular the XML documentation comments have to include inputs, outputs and side effects of components and functions.

****

Furthermore, Atomineer Pro extension for Visual Studio can help the team automatically create relevant XML comments by praising through the code. The members can then adjust and reformat the comments to bring the up to a standard.

# 5.0 Deployment & Maintenance Strategy

The team will install all of the required software on the server machine, which includes XAMPP, Unity and the selected database. The proper process to install the system will be described in the documentation, listing prerequisites and tentatively a script will be prepared to auto-install them on the server and client machines, if the game is to be in the browser, no installation will be required. The inclusion of an auto-install script will depend on it’s attainability.

There is no maintenance strategy required. However, if the game is updated in future iterations, the new version might have to be downloaded on client machines. Hence, an installation script or an auto-update within the Unity framework will be accounted for.

# 6.0 Performance Criteria

The client machines, where the game will be played, need to be powerful enough to render assets and run the game smoothly and without any frame lags. This means that they need to meet the minimum requirements of:

* Processor: Intel or AMD, Duo Core, 1.6 Ghz.
* Memory: 2 GB RAM.
* Free Hard Drive Space: 10GB.
* Compatible mouse, keyboard etc.

The server allocated by the client (Chao) needs to be able to handle at least 50 unique connections at once. Hence, the above requirements are also the minimum specifications for the server. To be able to handle a multiplayer framework and/or more people, more resources will have to be allocated to the server. The team has proposed a virtual-machine to be used as the server from AWS or UWA IT. This will allow performance to easily be scaled-up and tailored to only use the resources that it needs.

# 7.0 Project Management

The project will be hosted on Github. This will enable the team to simultaneously work on the code and share the project with the client. Git will also provide version control which is essential to any project. In particular it will store the code in the cloud and enable on rollbacks.

The team will use Trello to efficiently organise and manage the progress of the project. It enables on group collaboration through creating cards (tasks) and checklists which make it easy to track progress of the individual members as well as the overall plan. Trello also has additional applets which make project management easier including TeamGantt.

Slack and Facebook chat will allow the team to efficiently communicate and coordinate meetings and tasks.

# 8.0 Appendix

## 8.1 Players & Vehicles

