Flocking & Steering 2D Racing Simulation/Game - “Road-X”

Final Report

Submitted for the BSc (or MEng) in
Computer Science

April 2016

by
Nick Ignat Smirnoff

Word Count: 13595

Contents

1 INTRODUCTION 2

2 AIM & OBJECTIVES 3

Objective 1 – Replay Ability (Passive Objective) 3
Objective 2 – Graphics 3
Objective 3 – Application 3
Objective 4 – Gameplay & Mechanics 3
Objective 5 – Advanced/Extra Features (Secondary Objective) 4

3 BACKGROUND 5

3.1 XNA Game Studio 5

4 TECHNICAL DEVELOPMENT 6

4.1 Graphical Assets 7
4.2 The Camera 9
4.3 Game Objects 11
4.4 Lanes & Object Placement 13
4.5 Player Score 16
4.6 Game State and User Interface 17
4.6.1 Main Menu 18
4.6.2 Playing 21
4.6.3 Controls 24
4.6.4 Pause 26
4.6.5 Game Over 28
4.6.6 Exit 34
4.7 Audio 34
4.8 Testing 35

5 EVALUATION 36

5.1 Project Achievements 37
5.2 Further Work 38

6 CONCLUSION 39

APPENDIX A: INITIAL PROJECT PROPOSAL DESCRIPTION 40

APPENDIX B: DEVELOPER TEST INSTRUCTIONS 41

APPENDIX C: USER TEST FORM 42

APPENDIX D: USER TEST RESULTS 43

APPENDIX E: TIME PLAN 47

REFERENCES 48

1 Introduction

This document will outline and follow on from the previous introductory reports used to outline the initial ideas, concepts and plans for the project. 'Flocking & Steering 2D Racing Simulation/Game' Project, now referred by the game title 'Road-X' is essentially a 2D road crossing game where the player takes the comical role of a chicken and overcomes the mass traffic that is generated in front of them.

With no real end to the game the main focus is the score that is presented to the player and increased by the intervals of 1 for ever progressive step they make going forward on the treachery roads. This was the original intention for the direction of the project and the fundamentals that moulded the game; challenging single player experience with a constant drive for replay ability. 'Easy to learn hard to master' type approach was the key when it came to making decisions on the development of the game, where the player can pick the game up from their first experience and is drawn to the game and keep re-playing to achieve higher scores for self-achievement and/or to compare and compete with other players, which is obviously a very difficult concept to grasp and is sought after by any game developers but the uncontrolled variable of the players likes, dislikes, opinions makes this a hard goal to achieve and the best any developer can hope for is the majority of players that do play the game experience this.

The idea was to recreate an 'over the top view' experience game of simple mechanics; the player used directional movements to avoid and overcome different objects and obstacles, this introduces a skill element to the game where the user is required to react quick enough and in the correct way or lose the game.

Aesthetics play into the aim, as in order to get the player wanting to come back to the game, the game required to firstly be pleasing to look at, secondly being clear to interpret what everything is; the player will be required to make fast reactions and in the correct directions to overcome obstacles and proceed in the game, if the obstacles are hard to work out it would greatly affect the gameplay and the overall player experience [1].

The PC target platform has been maintained, as well as the developing tools that did undergo a large amount of consideration and compared against each other with pros and cons the outcome of which was for the project to be developed mainly using Visual Studio 2015 under XNA C# programming language, with obvious exceptions for graphics and any external assets of which have been created with Adobe Photoshop CC.

[1] *Refer to Appendix A.*

2 Aim & Objectives

2D arcade style game where the player controls a chicken to overcome waves of traffics to achieve the highest possible score before succumbing to an eventual unavoidable mistake, thus giving a high score which can be then be used to compare with other players and/or a goal to beat next time.

Objective 1 – Replay Ability (Passive Objective)

Although not required for the completion of the game, replay ability is something that will have an effect on the actual mechanics and features of the game at every step. The aim is to find a balance in simplicity and engagement of the game with the player. Keep the game simple enough so the player can instantly pick up the game and keep them engaged with different features that if they should fail they will want to replay the game to do better and progress further.

Objective 2 – Graphics

Graphics will play a big role in the success of the project, as not only is it what the user interacts with but it also is what sets the foundations for all the upcoming objectives. The game has to be aesthetically pleasing and clear as to what everything is, the environment and object play a big part in the actual player objective; the player is required to identify the dangers of the upcoming obstacle and attempt to react fast enough and in the correct way to proceed with the game.

Objective 3 – Application

Using the different assets created from the previous objective; the development can be initiated on the game application. This objective will consist of the creating the Windows form for the actual game and implementation of the different scenes and elements that will be used to make up the physical state of the game. The ‘Main Menu’ will be the first scene that the user will be greeted with upon running the game application and will consist of different elements to guide the user through the different options and start the game. In-game the user can be put through different scenes whether that be graphical terrain or the different obstacles.

Objective 4 – Gameplay & Mechanics

Taking the vehicle graphics (initially a chicken) from ‘Objective 2’ and adding gameplay mechanics to the object; this objective will define the manoeuvrability the player has over the vehicle they are controlling. The controls will be very simple to relate to the ‘easy to learn’ approach of the game. The user will only need the 4 basic directions (up, left, up, down) to overcome the different obstacles, and to an extent, add a skill factor to the game that will relate the ‘hard to master’, as the player will need to rely on reaction, timing and moving in the right direction to proceed in the game.

Objective 5 – Advanced/Extra Features (Secondary Objective)

This objective will become the focus of the project should there be time remaining after a working build of the game has been achieved, as a secondary objective it is not as important to the requirement of the game and the initial features implemented to make the game run. Thus the points in the objectives are here to provide insight on the ideas and features considered and are in no way to indicate the confirmation of implementation; some points will be delicate and have the potential of having a negative effect on the game, which is why they are not required and very unlikely that all will be used due to this fact.

- Lives – This is game changing as it will allow the player to fail a number of times before they achieve their final score.
- Power Ups – Adds another element to the game: luck/RNG (Random Number Generator), can be a controversial feature. However, measures can be implemented to counter the potential negative element by only allowing a limited number of power ups per play through or giving a power up every score milestone.
- Additions to obstacle library – Add extra obstacles that will generate in the game.
- Additions to vehicle library – Add extra vehicles the player can play as.

3 Background

Flash games used to be really popular to kill time in the past and although that still holds true the player base has largely moved platform, a big percentage of the lightweight/time killing games are now played on mobile devices. Although the aim isn't to follow this trend, it is a good source to get inspiration and recreate the different elements that made these lightweight games so addicting.

There have been a number of different inspirations and ideas for the author, the first mentionable one being a driving spin off from the very popular mobile game named 'Temple Run' which is a 3D endless running game where the player runs away from a rolling ball, score is earned by picking up gold and overcoming obstacles such as gaps and overhangs by jumping, sliding and strafing in the appropriate directions. Another notable mention would be of a simple flash game that the author played in the past named 'Crazy Taxi' where the user avoids traffic by steering with the left and right arrow keys, controlling the speed with the up and down keys and jumping with the space bar.

The overall inspiration for the game comes from the similar mechanics and elements of the 2 games mentioned above and from personal experience with the simplistic style of games that pull the player in and create an incentive to play and get better/higher scores.

3.1 XNA Game Studio

The XNA toolset was first announced in March 24, 2004 and the initially released 2 year later on March 14, 2006. XNA 2.0 and 3.0 were released soon after in the years following years of 2007 and 2008; Regrettably after all the focus and yearly updates to the toolset XNA started to decline from this point and what was going to be the next update named XNA Game Studio 4.0 later turned out the be the final real version of the tool set which was released on September 16, 2010. The "End" of the toolset was confirmed by a Microsoft Developer Promit Roy in 2013, stating that XNA is no longer actively being developed.

Fortunately, over the course of its short life time XNA gathered a lot of flexibility and features that set it apart from other toolsets and game engines and for what it is worth at the final stage this freeware toolset XNA Game Studio 4.0 was the chosen development environment for the project "Road-X".

This choice will inevitably be contested by the now regularly supported, open source, more advanced and spiritual successor to XNA, MonoGame (2014); the simple reasoning behind dismissing MonoGame is that it doesn't actually use the XNA Framework and it is just a re-implementation of the Framework, something that is widely more recognized and is more directed at Microsoft devices. The goal for MonoGame is to allow XNA developers on Xbox 360, Windows & Windows Phone to port their games to the iOS, Android, Mac OS X, Linux and Windows 8/10, as well as PlayStation Vita, Xbox One and PlayStation 4. This does not correspond with the target platform of the project and will compromise some features that will possibly require a keyboard.

4 Technical Development

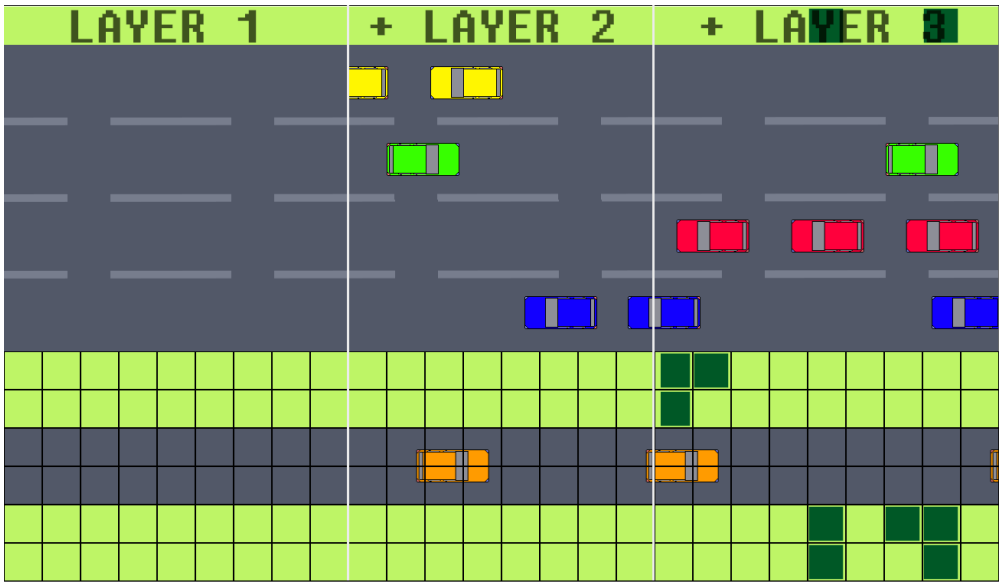
The design plan of the application makes the application start by presenting the player with a main menu of which the options are to play the game or to simply exit. On play the player is dropped right into the game and processes are set in motion to have the player avoid traffic and proceed as far as possible before essentially losing the game at which point the final scene is presented where the player has their score displayed to them and is able to play again or to exit. This design is simple to coincide with the image of the project to focus around replay ability; the fact the game is so lightweight only benefits the aim and allows the user to keep playing simply and without delay should they wish.

The technical Development process of the project underwent a major overhaul at which point the second revision development state began. There have been a number of changes for the project both in between the two revisions and in comparison with the plans of features but the motive and focus of the objectives stay the same. The intent of the project has always been focused around the main objectives and any technical decisions that would impact the application were made with the first and passive objective in mind; with this in mind, caused the project to evolve into the second revision build of the application after the completion of the first revision.

Every point and objective was covered within the first build of the application however to strive further and achieve a greater potential for the project the second revision was able to take a leap in further development and include further features that were not originally planned. This does still comply to the initial thoughts for the potential features of the project during the planning phase; it was noted that not all potential features can be included not for the technical development reasons, but more so for the mechanics and game play features of the game. Adding something like extra lives to the game will add a whole new element to the gameplay and can potentially hurt the project rather the improve the experience; as these features are labelled as potential extra content they were decided during the further development phase and after a working build of the game has been complete. This section of the report will outline in further detail of the different features that have been developed within the application and constantly compare between the two revision of the game.

4.1 Graphical Assets

The plan for the project was complete and the development phase of the project began with the creation of the initial graphics assets; the background which consisted of lanes split up into roads and grass strips, this was the bases of which the cars will be driving on and randomly generated bushes be placed for the player to traverse each lane while avoiding cars and working a way round the bushes. The player mechanics of the game operated under a square grid, the player would move between each square in any of the 4 basic directions (up, down, left, right) at the beginning each square was declared to be 50x50 pixels. With this information the background (first layer) was created using Photoshop and grid layout of square matching the declared size. The background size and thus the window size was established to be 1300 pixels (26 Steps) wide and 750 pixels (15 Steps) high.

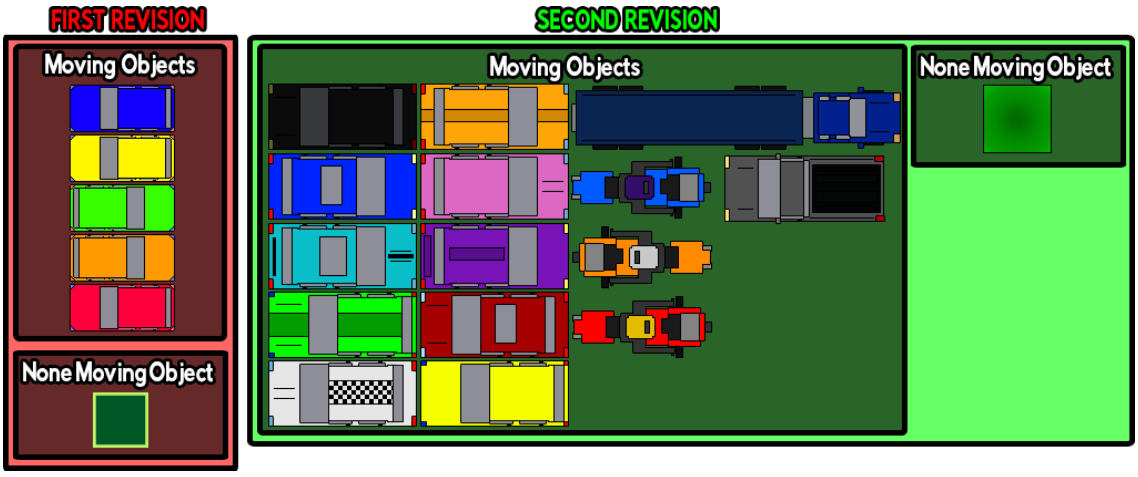


As visible from the image representation of the grid and the 3 different layers of assets that will make up the first revision of the Project. The very first layer is just simply the world background, the second layer adds the moving objects in this case the different cars and finally the third layer adds the none moving objects in this case the randomly spawned bushes. With the simplified mechanics of the first revision the Texture of the bushes would be randomly generated every time a new game is launched however would stay in their place throughout the current game session.

The second revision of the Project began with optimization and further feature development in mind; the very basic funcionality was in place so the first thing for optimizing the game was ‘Power of Two’ image dimensions. The original layers and order of the assets stay the same but the sizes of the different assets adapt to that of a compatible size. The size of the ‘World Background’ Image was changed to 1024x1024 pixels and to reflect this change the size of the squares that make up the ‘grid’ were increased in size to that of 64x64 pixels. Using a ‘Linear Wrap’ Sampler State, it become possible to continuously recreate the ‘Word Background’ Image vertically in a tiled effect.

```
// Begin - Draw using LinearWrap and the scroll effect
spriteBatch.Begin(SpriteSortMode.Deferred, null, SamplerState.LinearWrap, null,
null, null);
// DRAW - World Background
spriteBatch.Draw(
worldBG, // Texuture - World Background
Vector2.Zero, // Position
new Rectangle(0, (int)(-scrollBGY), 1024, 1024), // Rectangle with variable on the
Y-Axis which will depend on the character movement
Color.White); // Colour
spriteBatch.End();
```

This further advancement within the second revision allowed a great deal of further features to be implemented that even surpassed the aims and objectives of the project. The graphic asset palet between the two revisions is vastly different; there are a wide variety of assets available within the second revision of the game that is incompatible with the first.

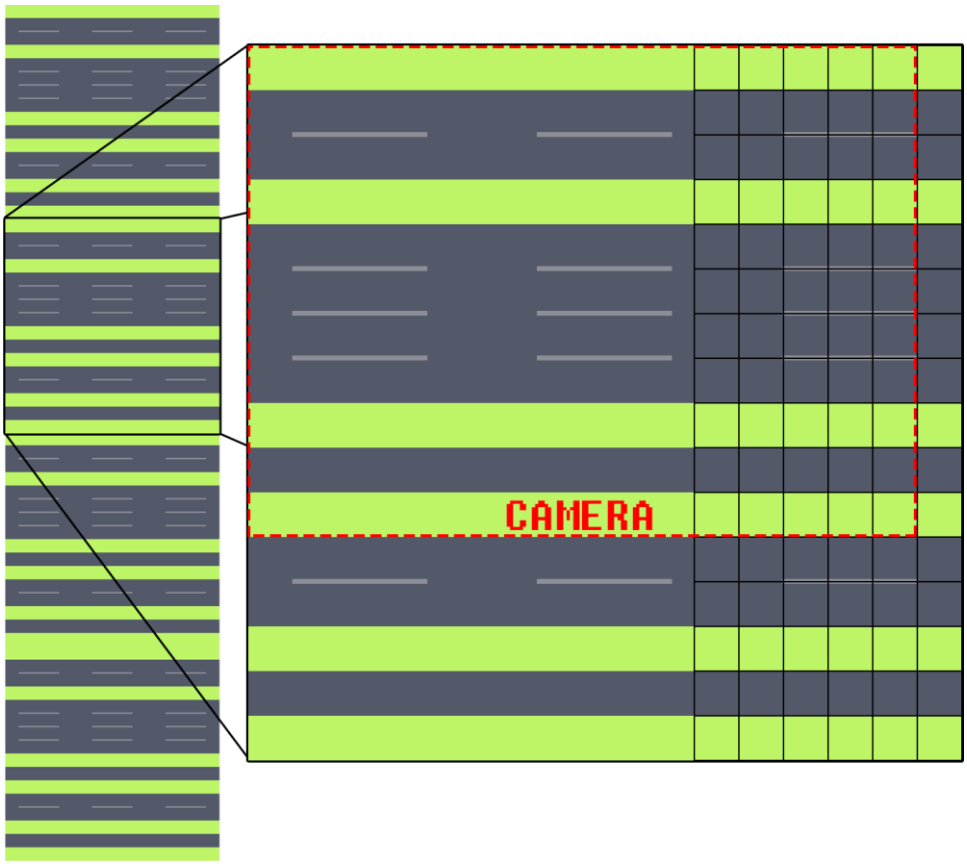


The technqiues used in the second revision, greatly improve on the initial implications of the first revision of the project, these changes and improvement will be reflected with comparisions throughout this report. With the changes the aims and objectives of project are achieved and in some cases surpassed with implementation of various objects of different charactistics; as visually represented from the graphcis palet comparision above.

4.2 The Camera

The first revision of the Project did not have a Camera type view and was just simply the full view of the background image, the player was constrained in staying within the window with the boundaries declared to the size of the background image/window. However the 'Top' boundary had a different set of mechanics compared to the other three sides, in which made this revision of the game 'endless' in its own right; upon reach the top most row/lane and inputting the 'Move Up' command the user will be teleported along the Y-Axis all the way back down to the bottom of the window while maintaining the X-Axis and allowing the Player to carry on accumulating higher scores, this coincides with the plan and scope of the game to keep it endless to the ability of the player, and as long as the player has to the skill to out-manoeuvre the moving vehicles and find a path round the bushes they can keep accumulating higher scores.

Due to the fact the background is 'limitless' and scrolls vertically a camera type view is required of which will be difined as the size of the window. Furthermore as the game enviroment is made out of a grid of which the player moves between the different squares, it would be essential for there to be an odd amount of squares horizontaly in order to acompany a 'middle' column for the player. It would have been mathmatically impossible to have an odd amount of 64x64 squares going across a 1024x1024 image therefore a column is cut from view. Therefore the defined size of the camera and thus the window for the second revision is 960 pixels across and 704 pixels high. This not only compliments the new sizes of textures and images in accordance to the 'power of two', but also the slightly decreased size from the first revision allows the game to be ran on older monitors that are not quite High-Definition.



The variable that defines the background image Y-Axis can be manipulated on ‘Move Up’ command to move the all of the contents that has been drawn to be moved down an entire row (64 pixels) while leaving the player character in place, this wrapped in a ‘If’ statement to be triggered if the user enters the ‘Move Up’ command while half way up the screen will give the effect of movement forward and be be carried out as along as the player is still in the game.

```
// IF Player moves up half the screen (Hit the Boundry)
if (playerCharacter.Location.Y < getWindowHeight / 2)
{
	// ADD to scrollBGY which will update the Y-Axis of the World Background
	scrollBGY += 64;
}
```

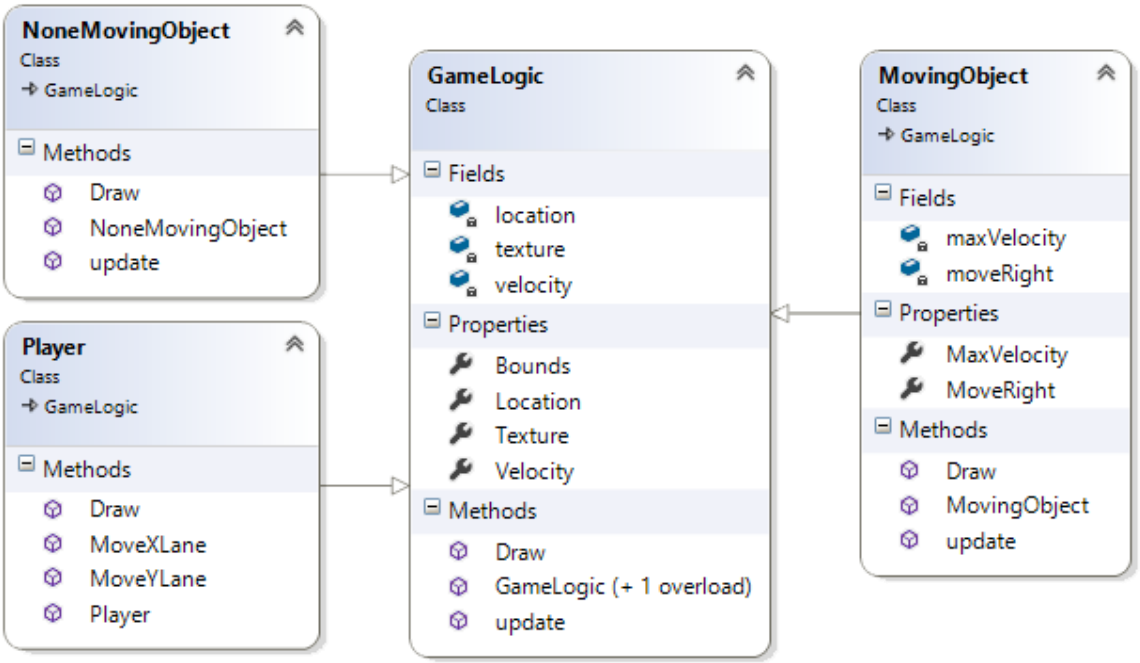
With these mechanics despite the player constantly moving forward and everything gives the effect of moving down, it is interesting to understand that at no point does the player character actually move in the upper half of the screen nor out of the X and Y-Axis values of the lower half of the window and with the world boundaries stopping them from moving out of the camera window (Left, Right, Down) the player character is forever within the range of 0 – 960 pixels in the X-Axis and 352 – 704 pixels in the Y-Axis.

4.3 Game Objects

The application mechanics are relatively simple to grasp the fundamental workings of each object, these objects types can be split up into 3 main section classes; player, moveable objects and none moveable objects. The player class mainly consist of the controls of which the player can take to move the player object across the screen, the moving objects will be things like cars and vehicles that the player will be required to avoid and will consist of mechanics to generate the object and make it move across the screen while maintain a bounding box to act as collision algorithm to check if the player has intersected/collided with a vehicle thus losing the game. Next up, none moving objects which will start of by bushes and be the obstacles the player needs to overcome when not on the road as these bush objects will be unpassable and will leave the player to have a set route for the challenge.

Due to the fact there is a lot of constant movement and updates to the different object classes at any point of the game especially the positioning of each object, the way in which it has been designed is of extreme importance in order to track where an object of a particular class is and how it is generated. This statement holds even more truth in the second revision of the game as the whole element of gameplay changes to a more technical stand point in which every single element on screen is moved at times with a single player movement.

With so many interlocked processes it is essential to have an appropriate tracking and updating measures for each class and in doing so keep the process to a minimal while achieving all that is required for the objects to operate under the given conditions, ultimately to keep performance up and make reactions more effective.



The Game Logic architecture describes how an object is inherited from, and further expanded to perform the necessary actions, that allow each object to function while maintaining the desired properties and variables of which can be manipulated with ease depending on the different game factors. Both None Moving and Moving Object classes factor in a Library of objects that come with their own individual parameters and are split depending on whether the object has movement within the game or not; as to the reasoning behind why not have the None Moving Objects simply have 0 velocity and be stored in the Moving Object Class, the player interacts differently upon colliding with a Moving Object in comparison to a None Moving Object and the separation provides a greater definition between the two types of objects and how to handle the update method.

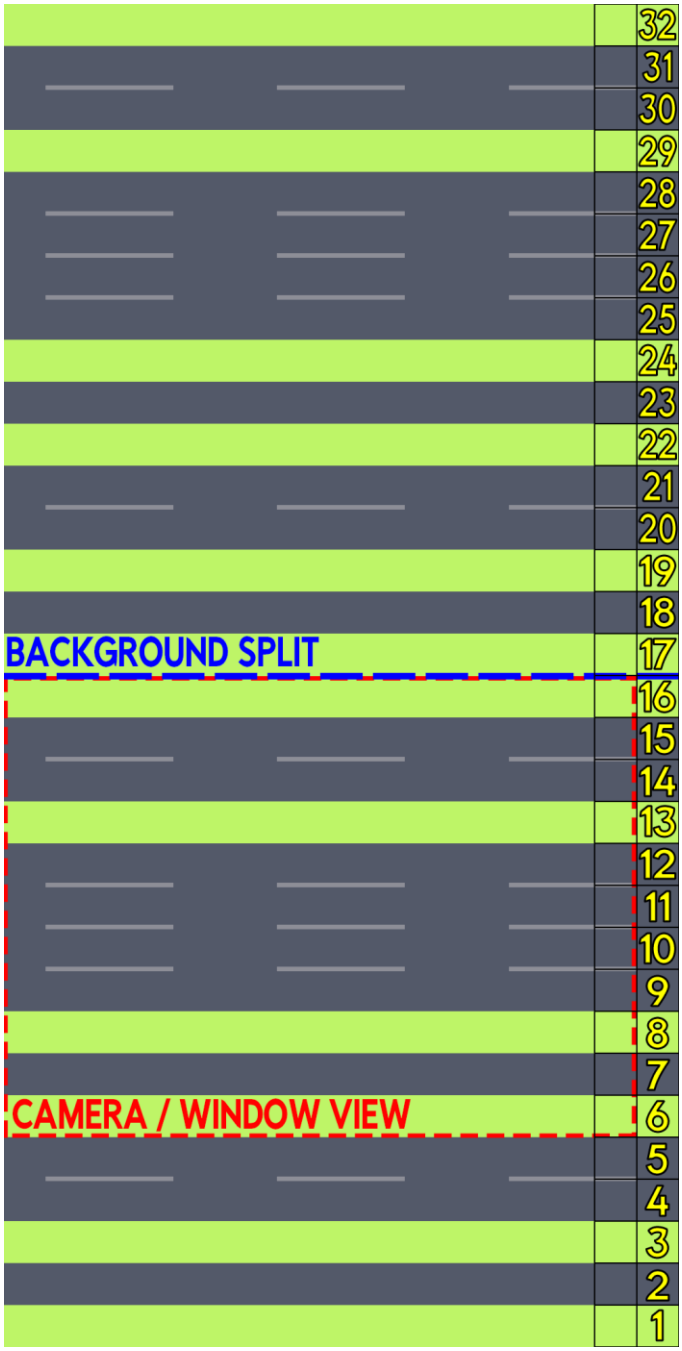
None Moving objects due to their lack movement as suggested in the name require no extra properties other than that, that are already provided by the Game Logic. While on the other hand objects that do have movement as such Moving Objects factor in a direction parameter “MoveRight” which of course only requires a true of false argument in order to determine in which direction the Moving Object will be moving (Left or Right), “MaxVelocity” declares the maximum speed of which the object moves allowing the potential of ever varying speed of certain objects.

The two object libraries that accompany their counterpart Object classes are likewise split on whether or not the object moves respectively; these library lists are there not only to parse in the given objects and its parameters but also give extra potential of further development and addition of objects that can be added to the game in the future; of which the Moving Object Library has proven to be a great benefit with the contents of which include a wide variety of vehicles all moving in varying directions and at varying speeds.

The Player Object is focused on the singular entity of the player on the ‘board’ their position, movements, texture and Bounds of which are already included within the Game Logic and do not require further properties. The distinction between the different objects and their parameters provides precise actions and outcomes of certain situational interaction between the player and the Objects within the game. The “Bounds” parameter changes the outcome of the interaction when the Player Bounds collide with a Moving Object as that results in the game being over, versus the other option of collision with a None Moving Object of which can be a Bush and simply results in a reversal of a move.

4.4 Lanes & Object Placement

With the necessary assets and game objects created, the next planned stage was the placement of objects on the different lanes, this section will outline the process involved with the lane placement and generation of objects used within the game and will be mainly focused on the second revision of the game with minor references to similar mechanics within the first revision to compare. As previously mentioned, the game world is set out in a 'grid' this results in rows of the grid being represented as Lanes. These lanes are laid out on the Background image and the different object classes are placed on top of the image, positioned so it is in accordance with the defined lanes environment; grey lanes represent a road meaning that Moving Objects will be placed on these lanes, green lanes represent grass meaning that None Moving Objects will be placed on these lanes.



To further follow up on the process of the world background being drawn in a tiled vertical format. A game 'set' is made up of two of the world background images being stacked on each other; there is only one game set being drawn on the screen at any time and each game set is maintained throughout the game.

With reference to the image to the left, it will be possible to give a detailed description of the game mechanic process. The numbers along the right of the image represent the lane numbers, the blue line represents the split between the two background images that make a game set as previously mentioned, and finally the red box outlines the game window and thus what the player sees at any time. The image illustrates the accurate positing of the window upon launch of the game.

The player is spawned in the middle of 'Lane 6' and has the entire of the bottom half of the camera window to move around without any internal effect. 'Lane 10' is the soft-border for the player this means when they move up to 'Lane 11' they over step the border and the camera mechanics kick in; the camera window will move up

one square/one lane, by moving all the content down a square; this has been described in the 'Camera' section in this report previously, what needs to be noted here is the visible lane range the user sees has been shifted, from the initial visible lanes being between 6 and 16 it has been shifted up the Y-Axis to be between the ranges of 7 and 17. The shifting process is a loop based on the actions of the player, the process is repeated accordingly based on the player overstepping their soft-bound. The lane that disappears from the bottom of the camera that lane contents is relocated along the Y-Axis by the length of a game set (if each background image is 1024 pixels high, the length of a game set is 2048 pixels in height). In accordance to the illustration when the player moved the camera up to the ranges of 7-17 the previously visible 'Lane 6' is now out of view and the contents of the lane is relocated in the next 'Lane 6' (2048 pixels up). This happens every time a lane goes out of range of the camera, as it is not in the interest of the game to allow the user to go back down, the contents of the previously viewed lanes become redundant and are relocated up, to the upcoming game set. The whole process is an endless loop and each lane that makes up a game set is set in advance for the next loop round the user makes as long as they are alive and moving up.

The contents of each lane is recycled once the lane has been surpassed by the camera to maintain the best performance. If the contents maintained on it assigned lane, there would be an ever growing number of redundant objects that still operate and use the machines resources off camera and in no way impact game one they have been surpassed and thus will require the list of objects to keep growing the further the player got in the game. This was not an issue in the first revision of the game as explained in previous section the window was never manipulated and therefore no real use of a camera type view, meaning that all the lanes were always on screen and the process never changed; once the player reached the top of the screen they were teleported back down to proceed in achieving higher scores.

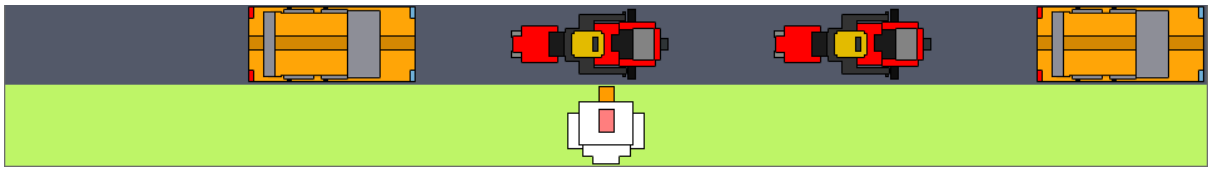
Although each lane is recycled the generation differs each time so there is no real pattern to any lane other than that of the graphics/objects used. The world background is tiled and is made up of 16 lanes, it is to be expected the objects will be same every 16 lanes. This holds true to the type of objects but the moving objects graphics differ slightly and follow a pattern of graphics used per game set rather than per world background image; this means although 'Lane 2' and 'Lane 18' are the same lane roads on the background image, they can hold different graphics of moving object and thus different parameters, however they will always stay true to what they are in that numbered lane; once 'Lane 2' will go off camera and be moved up a game set it will still be 'Lane 2' and still hold the same graphic and object. To end the point, despite the lanes objects maintaining a pattern; the generation of the objects and the spawn of objects will differ, this holds true to none moving object which change generation placement every time a lane is drawn. A counter is used to reiterate through all the 32 Lanes before resetting back at 1 to maintain the loop, with this process it becomes a matter of adapting the appropriate lane at the appropriate counter value. Every time the camera moves and a lane goes out of view, a value is added to the counter and a method is carried out based on the new value of the counter. Furthermore, to create the illusion of player movement upon the addition to the counter a process is carried out to go through each object in the application, moving and none moving and move them up a square (64 pixels) to re-align with the world background movement.

A full game set is drawn at all time, the none moving objects, are all placed in on the game set, once a lane goes out of view and the lane is moved up a game set the placement of the none moving objects in that lane is changed. Using the illustrated example one ‘Lane 6’ drops out of view and the contents of ‘Lane 6’ are moved up a game set to the next ‘Lane 6’ the placement of the objects initially used will be different. Similarly, with the moving objects once ‘Lane 7’ is relocated a game set up to the new ‘Lane 7’ the order of which the objects spawn in that lane will not be same as it was in the previous lane prior to the relocation. This is done through timers and random generators.

Firstly, to address the ‘random generators’ used, computers are deterministic machines and without special hardware it is impossible for a machine to generate a truly random number. With that, my ‘random generator’ arguable is as random as possible within the limitation of the requirements. Acquiring a ‘seed’ based of the local time of the machine allows the random generator to generate a value based of the acquired seed of which will be different every instance of the application with the assumption the local time of the machine is operating as intended. Using the random generator, it becomes possible to generate a number and multiply it by the length of a square (64 pixels) to achieve a random placement on a lane for a none moving object.

```
// Generate a seed based on time and use the seed to get random X
int seed = unchecked(DateTime.Now.Ticks.GetHashCode());
Random rndSeed = new Random(seed);
int rndX = rndSeed.Next(0, 15);
```

This process is done per “bushLaneY” and can be repeated with each newly generated ‘grass’ lane to generate a random placement of none moving objects. In the first revision this process was also used to generate the moving objects however this occasionally caused cars to spawn within a few milliseconds of each other and overlap, this could be fixed by establishing a checking method that disables the feature for a short time after spawning a moving object or simply moving an object further along the X-Axis if it overlaps another object. The moving object process was changed in the second revision to use timers, which allowed the ability to monitor the length of time which passed since the previous spawn of an object and manipulate the spawn rate of each lane independently.



This was a far greater implementation over the first revision as not only did it allow more control over every independent moving object lane but also allowed the ability to integrate a random number generator which could handle the small possibility of spawning a secondary graphic into the lane, which of course is far more suitable then handling the core spawn mechanics.

4.5 Player Score

The Player accumulates score based on the lanes they surpass; each new lane forward is an addition to the score. Both revisions require a variety of rule sets to allow the score to function as intended and disallow the player from taking advantage based on the game environment. There are two main variable integers that maintain the score to that of the intended values; The player score value and a temporary reverse value.

In both of the game revisions, the score mechanics are integrated with the player controls. Each time the player executes a command that moves the player down, the value of the temporary reverse value increases by 1, on the flip side it decreases by 1 when the player moves up, assuming the value is not 0; under these circumstances the player is moving back up the lanes that they previously moved down in. In the case, the temporary reverse value is 0 and the player moves up, the score is increased by 1 as the player has moved up to a new lane.

Additional rules are used to stop the user from taking advantage to manipulate these values when moving against objects and bounds that are not passable for the player; window boundaries and none moving objects. When moving up and down in the direction of an impassable object or boundary, the score mechanics should not be carried out depending on the chosen direction. For example, if the player attempts to move down into a bush (none moving object) the mechanics to add 1 to the temporary reverse variable should not be carried out due to the player not actually moving down and being stopped by the object. Without such rules in place the user can continuously accumulate score by moving up against a none moving object and not actually moving while score mechanics still functioned.

4.6 Game State and User Interface

There are a number of game states used within the workings and navigation of the game. It is important to have a variety of game states to change between the different 'modes' of the game, which allow the organisation of the different objects and elements to be drawn at appropriate times and are not overlapping one another. Overall, the use of game states allows the different navigational options available within the game to function and to be differentiated between one another. The different features and drawn elements are able to be separated by a switch method, the different elements and mechanics of which the game operates are switched based on the active game state.

Both revisions of the game declare a list of enumerated keywords which are distinct type that consist of a set of named constants called within the enumerator list. Using this enumerated list, a Game State based of the keyword can be assigned. The first revision of the game uses the following Game States; 'MainMenu', 'Playing', 'Controls', 'GameOver' and finally 'Exit'. This holds similar to the second revision of the game with only a minor change, the "Controls" game state which is removed and replaced with a new game state, 'Pause'. These different game states all hold their own related objects and User Interface elements which will be outlined in this section.

The aim for the user interface was to have a minimalistic interface to only include the required information for the player at any time; with this in mind placement becomes very important and has to be considered under all environments and situations that the game has to offer. Ease of use is paramount when creating successful apps and websites and holds true for my project, a great deal of time has to be spent in the design of each element in order to achieve an aesthetically pleasing look and get the required information across.

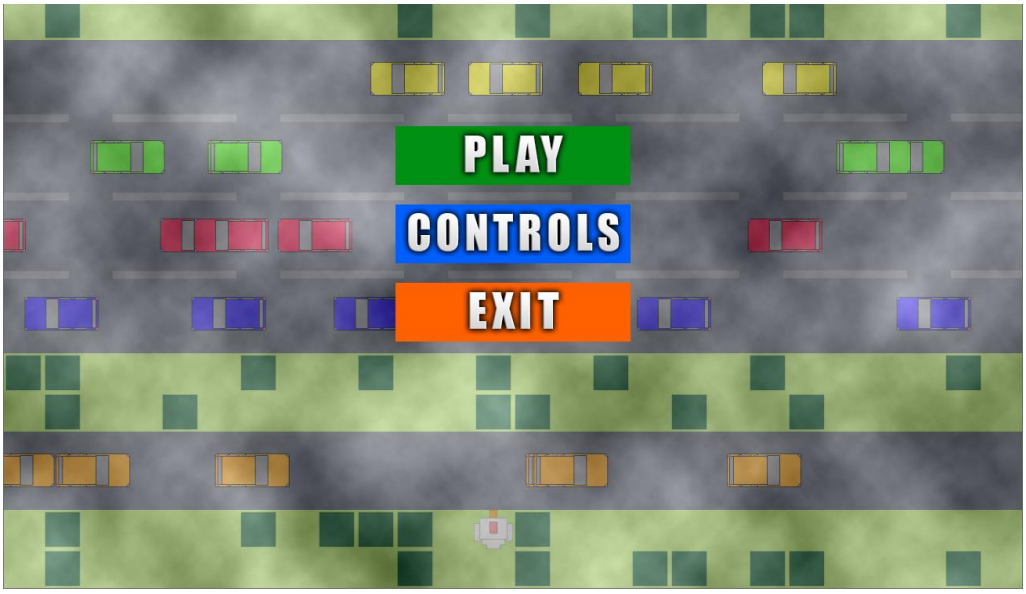
Aesthetics are very important when it comes to a User Interface as this is the element of the application that the user will be using to make choices and get information about the current gaming session. The user will spend a large amount of time using these Interface elements and thus require them to be understandable and have a balance of appeal; the element should not constantly draw the users eye as to not distract them while still maintaining a reasonably appealing look for the time the user does interact with the element, it looks like it belongs and not out of place.

The user should be able to achieve all the desired actions and requests without requiring external help and feel lost. All the information and choices available to the user should be easy to understand and navigate through different options to achieve the desired result. It is important that the user gets what they want with the least amount of effort and options to go through; this factor has been a big design focus for the project. There have been a few elements that have been adapted in between the different revisions of the application.

4.6.1 Main Menu

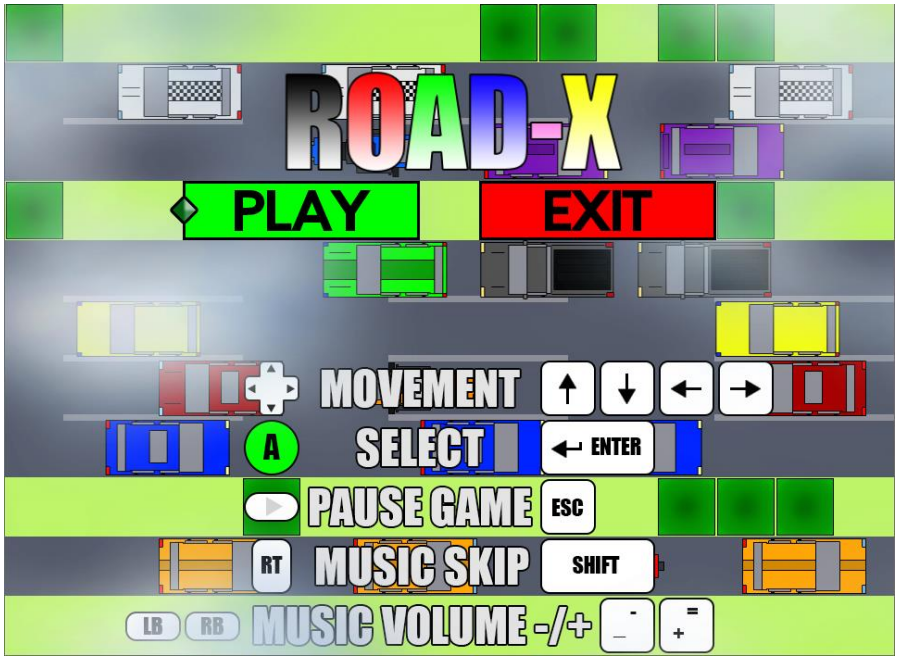
The ‘Main Menu’ game state is the initialised game state of the application, meaning that upon launch this is the first game state the user will be greeted with. The objects and elements used in this game state will be directly related to the navigation of the game; by being presented with specific elements the user will be presented with a choice of which will allow them to navigate to other game states of the game.

The User Interface elements on the Main Menu changed the most through the two different revisions. Initially the Main Menu consisted of an instance of the game running in the background, the moving cars and the generated bushes are all visible under a static grey ‘cloud’ overlay image. This held true for the second revision with a change to the ‘cloud’ overlay image which used the same technique of tiling the image endlessly as the background image did with an exception of an additional variable to the X-Axis as well as the Y-Axis and both variables to be increased at the same time by small amounts within the update method giving a consistent smooth scroll effect. The scrolling cloud effect in the second revision adds an immersive effect upon the launch of the game. Both revision however do include the cloud overlay for the purpose of portraying a clear omnipresent effect upon the player; the player looks down through the clouds onto the ensuing dangers with the challenge of guiding the chicken across the never ending lanes.



The above image is a screenshot of the ‘Main Menu’ taken from the first revision of the game.

Within the Main Menu of the first revision the user is then presented with 3 options; ‘Play’, ‘Controls’ and ‘Exit’. These options are textured images that act as a button through a dedicated button class and are used in both revisions. As can be seen from the image there is nothing else that is available to the user other than that of the 3 different options.



The above image is a screenshot of the 'Main Menu' taken from the second revision of the game.

The second revision Main Menu contains many more Interface elements however less intractable elements as there are only 2 available buttons. Noticeably, the 'Controls' options that was presented in the first revision is not available and the information that would have been provided in the 'Controls' game state is simply displayed on the Main Menu of the second revision. The 2 available buttons are also colour coded to that of a traffic light as to encourage the player to select the 'Play' option.

Both revisions of the game use the same button class that runs an update method which constantly checks for the boundary of the mouse pointer to intersect with that of the boundary box for the textured image, this allows the control of when a button is being hovered over by the mouse. Upon hovering over the alpha channel value of the texture creates a loop and decreases in size from the maximum value of 255, and increases in value from the minimal value of 0, once the user stops hovering over the texture value automatically increases to 255 if it is not already. When the 'button' is in the hovered over state and the left mouse button is clicked that means a selection has been made and passes a isClicked state which will alter the game state depending on the selection.

```
if (mouseRectangle.Intersects(rectangle))
{
        if (colour.A == 255) down = false;
        if (colour.A == 0) down = true;
        if (down) colour.A += 3; else colour.A -= 3;
        if (mouse.LeftButton == ButtonState.Pressed) isClicked = true;
}
else if (colour.A < 255)
{
        colour.A += 3;
        isClicked = false;
}
```

To the left of the ‘Play’ button there is a green rhombus shaped image which represents what is currently selected this is more so for the controller support of the game but also benefits the ease of use of the interface as the enter on the keyboard is able to make the selection as well. By pressing the directional keys on either the keyboard or controller the user is able to change the selection and move the green ‘selector’ image in between the different options. This also gives the ability to the player to never even use the mouse should they choose not to. Furthermore, more directly aimed to the keyboard user as the button availability of a controller is always within reach, the arrow keys are the keys used to operated movement, this results in the hand placement of the keyboard user to always be in that area of the keyboard. By allowing the keyboard user to make selections with the enter key greatly increases the ease of use and decreases the effort required to make selection this partnered up with the fact each selection takes less time the user will be more inclined to get into a state of pressing the enter key to retry the game upon a game over game state.

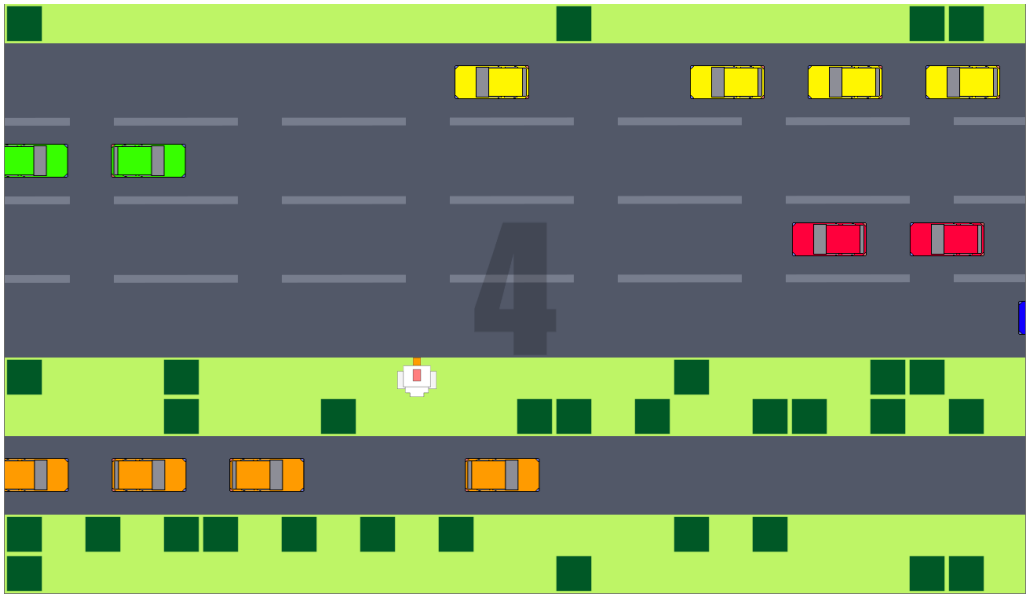
Moving to the Road-X title which is present in only the second revision of the game. By using the same technique as that of the buttons and cloud overlay the title size is able to be manipulated through a variable value, the Road-X title size changes through a loop of the size value that is able to increase and then decrease, based on the value itself.

```
if (roadXTitleIncrease)
{
	roadXTitleScale += 0.001f;
	// IF - The scale gets equal to or more than 1.2f
	if (roadXTitleScale >= 1.2f) roadXTitleIncrease = false;
}
else if (!roadXTitleIncrease)
{
	roadXTitleScale -= 0.003f;
	// IF The scale gets equal to or less than 0.9f
	if (roadXTitleScale <= 0.9f) roadXTitleIncrease = true;
}
```

Furthermore, there is an extra User Interface element that is not present in the screenshot of the second revision due to the fact the High-Score value is 0 and has not been set yet. Upon playing the game for the first time the when the player achieves a score that is above 0 the initial High-Score value will be set, at any point that the High-Score is beaten the new Score will become the new High-Score. Once the player launches the game again, with a High-Score value that is greater than 0, the user will be greeted with a message that displays their personal High-Score.

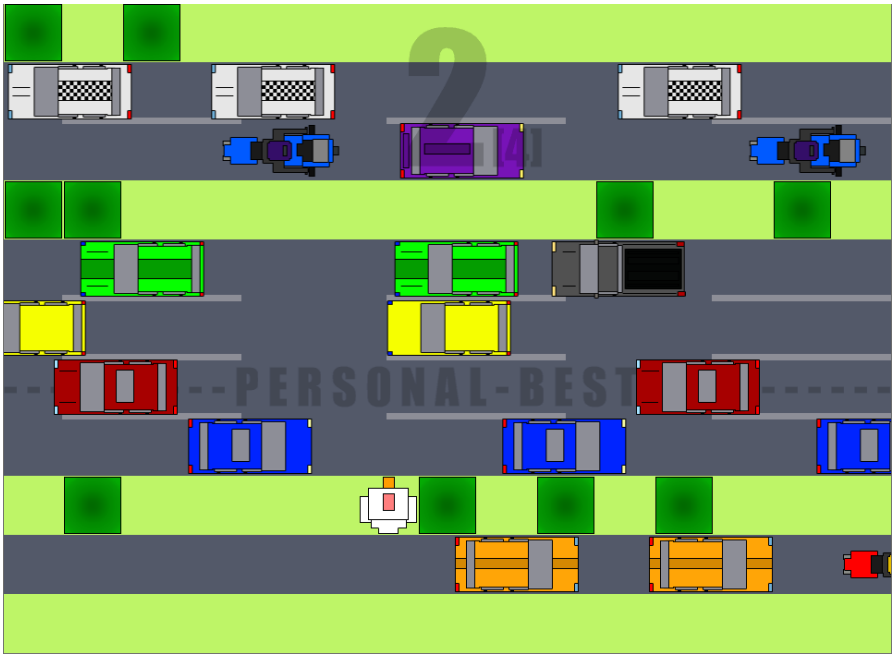
4.6.2 Playing

Upon making the selection to play the game, the application game state will be changed to 'Playing' this causes the mechanics of the game to be activated and all functionality related to actual game play will be in effect. This game state will consist of the core features and mechanics of the application and will be the game state where the user will spend most of their time will running the game. The 'Playing' game state contains the most minimalist User Interface design throughout the game. Due to the nature of the game mechanics that the heavy focus on reactions and pathing that is enforced upon the player, it is essential that there is nothing stopping the user having a harder experience than that of the challenge they are placed in within the game environment.



The above image is a screenshot of the 'Playing' game state taken from the first revision of the game.

Really driving the minimalistic design forward is the first revision of the game; the only real User Interface element that is on screen is the player score that is displayed in the very middle of the screen with a reduced alpha channel. The score is always up and acts as the background, with the effect of the transparency acting as though the score is edged into the road of the background image. All the moving objects are able to drive over the value without having any effect on the user experience. The value of the score is updated in real time to coincide with the movement of the player; with every new lane forward a point is added to the score value. A mention has to go to the positioning of the element; although there is only one User Interface element on screen it still requires a large amount of consideration for the design and placement. The placement of this element is particular and not to the tastes of the average arcade game of where the score is usually found in one of the corners of the screen out of the way of the player but always available whenever desired. The placement choice for the design of this scoring element in my project allows the value to always be in sight of the player while still having access to the actual gameplay elements, as to not distract the players eye to one of the corners of the screen, away from the deadly elements that could potentially hurt the players game session.



The above image is a screenshot of the 'Playing' game state taken from the second revision of the game.

The second revision of the game only adds one extra element. As explained previously at no point does the player character actually move in the upper half of the screen the player character is forever within the range of 352 – 704 pixels in the Y-Axis, this results in the upper half of the window 0 - 352 pixels in the Y-Axis to be player-free space. The score has been moved up to the top half of the window to occupy the empty space that is available. Additionally, the layering of the element has changed, the score is no longer in the space between the Background Image and the moving objects, as it has been moved to the very top layer and is visible on top of any object that is placed in the window.

Unlike in the first revision of the game the score has an addition parameter that is displayed alongside it. As visible from the image above the score is displayed as "2" with a smaller value of "[4]" to the right of it. Similar to the High-Score element on the Main Menu on the first play through of the game the second parameter, the value within the open and closed brackets is not visible. Only after the player has achieved a High-Score value that is greater than 0, will the parameter be added to the score and reflect the value of the players High-Score. The current player score is the value that is always displayed in the bigger font size, and is the first value in the string; similar to the score in revision one is updated in real time to represent the player score, the High-Score value is always constant throughout a play through of the game and is updated only when the player achieves a new High-Score.

The layout format of the score partnered up with the High-Score underwent a number of changes as it proved difficult to convey the information and differentiate between the player score and the player High-Score in an intuitive way that doesn't contradict the placement choices and reasons that were outlined in the first revision; not place elements in the corners of the screen to redirect the players gaze. It was established from the start to have the values together as they are both linked to score, to set up a distinction between them both a size difference was introduced. The initial format used had a slash (/) to further

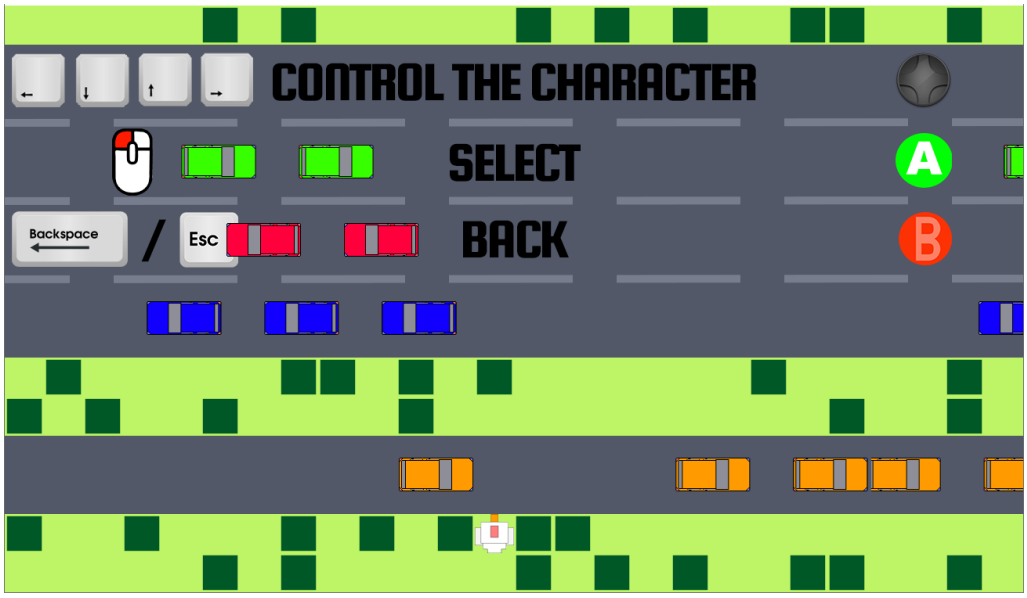
separate the values, so using the values from the image above; <Big “2”> <small “/4”>. Although this format did convey the desired information it had an effect that can be viewed as leading the player on; the fraction view gave the impression that the user had to achieve the full fraction which is equivalent to the players High-Score to achieve something. That was obviously not the desired impression to give the player thus the symbol that set the values apart was changed to a colon (:) to appear as <Big “2”> <small “:4”>. This was worse than the first view due to the values being displayed in this format made it seem like the element was showing a time, although it wouldn’t take long to realise what looked like the hour value was increasing with each move forward it still doesn’t help the user experience. The second to last format removed the symbol altogether <Big “2”> <small “4”>, this format relied on the distinction of the sizes to make it clear to the player the difference between the two values and although this format might have worked with information that did not consist of pure numeric values, it made the contents look like a whole number and combined the two values, which is clearly not the intended implication. The settled format was the open and closed bracket <Big “2”> <small “[4]”>, this separated both values and did not have implicit effects of the previous formats as mentioned.

A further note on the position of the new scoring element; a variable is used to determine the placement. A variable is required to keep the placement of the element in the middle of the screen; each value has a different width and with an ever changing score and an ever growing High-Score changes the entire width of the contents thus without a varying position can offset the origin of the element. By getting an updated total width of the element, divide it by two to find the half way point and using this value to displace the positioning along the X-Axis the User Interface element that displays the score and High-Score can always be drawn in the middle of the window.

The extra User Interface element that is added in the second revision can be easily seen in the screenshot; the line that goes across the lane outlining “PERSONAL BEST”. This is yet another element that is ‘Hidden’ on initialize under a statement that does not draw the object if the High-Score value is 0 and not set. Upon setting a High-Score the element that is layered directly above the background image as the score used to be in the first revision, is drawn. This User Interface element as the contents suggests outlines the lane the player achieved their High-Score, the position of which can be determined through the knowledge of knowing each square that makes up a lane is 64 pixels high that multiplied by the value of the High-Score allows the element to be drawn on the lane of the High-Score and gives a visual representation of the High-Score in game other than that of a value.

4.6.3 Controls

The game state dedicated to displaying the controls to the player is only relevant to the first revision of the game. As outlined in the 'Main Menu' section all the information relevant to the controls of the game is combined within that game state in the second revision. With the limited amount of information that is presented in this game state, it doesn't take long for the user to understand all that is required to operate the game. Once the user has seen this information this game state becomes redundant, and is of no further use. For that reason, this game state was removed in the second revision of the game and all the information relevant was simply displayed on the main menu, readily available to the player upon launching the game and with no further requirement of navigation.



The above image is a screenshot of the 'Controls' game state taken from the first revision of the game.

After selecting the 'Controls' button in the first revision of the game the user will be taken to an instance of the game where the controls are laid out across the road and the cars in the top most 3 road lanes are driving over the User elements. This user element is a simple '.png' image with transparency that is layered above the world background image and contains the relevant information. As to allow the user to read and understand the information laid out on the road the spawning mechanics of the relevant lanes are paused, so as to transition into this game state the already spawned cars in the top 3 road lanes drive across the screen and leave the lanes free for the users viewing.

This game state was an instance of the actual 'Playing' game state as it held the majority of the mechanics with minor tweaks that as mentioned stopped the spawning of the lanes that required visibility and further stopped the game state changing to 'GameOver' upon object collision; the player would instead be repositioned back in the starting position. This allowed the player to an extent simulate the game and learn the controls before taking on the game for real. Within this instance of the game the user is presented with information that outlines what keys and buttons are used in the game and what they do. One of the aims and characteristics of the project is easy to learn hard to master; there is very little to the game

as far as controls go, of which the user only uses the directional keys or buttons depending on what input device they choose, as it was planned to have a controller support.

Despite the control elements being displayed in different game states there is no real change in the information provided by the element due to the controls being maintained. The second revision of the game with the introduction of audio being implemented within the game achieved a couple extra controls associated with audio control. These controls allow the user to skip the currently playing audio track and change the volume of the track up to the point of muting. The design of the element within the second revision consist of personally made graphics which follow a consistent style, with a placement that is similar to that of the first revision of the game, where each button/key is placed on a game lane.

4.6.4 Pause

This game state is what replaced the 'Controls' game state in the second revision of the game. Even though the first revision of the game did have a feature that allowed the player to pause the game, the game itself did not have a dedicated 'Pause' game state. The pause feature in the first revision of the game simply took the player back to the Main Menu of the game, this still worked as a pause due to the fact the players score and position was saved across the game states, when the player hit the 'Play' option they would be placed back in the position they were with the same score when pausing the game. The player was able to pause the game at any point during the 'Playing' game state and even go into the 'Controls' game state instance and re-familiarise themselves with the controls and practice more before jumping back into the playing game. In the first revision of the game the user can achieve this command by either pressing the backspace key or the escape key on the keyboard or the start button on the controller, due to this feature not acting as a generic pause and rather taking the back to the main menu this feature was never labelled as 'Pause' and as can be seen in the previous section was labelled as 'Back'. The second revision of the game, with the implementation of a dedicated 'Pause' game state and labelled as such, the player would command the game to pause by pressing either the escape key or the start button depending on the input device.

The above image is a screenshot of the 'Pause' game state taken from the second revision of the game.

As can be seen from the image above the 'Pause' game state takes User Interface elements from some of the previously mentioned game states within the second revision. The entire design is very similar to the 'Main Menu' design and maintains a consistent style used throughout the design of the application. It should be noted that this pause feature functions as generally expected like the standard pause, used in other games; all the game objects used while playing are frozen, and player movement has been disabled.

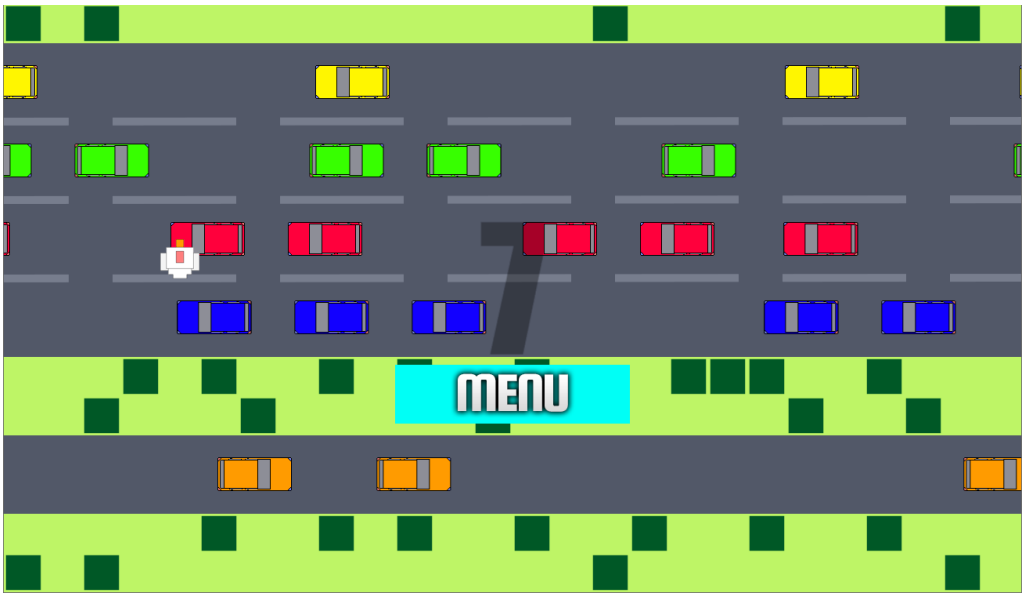
The “Road-X” title image uses the same mechanics used in the Main Menu that allows the image to increase and decrease in size within a loop, creating a breathing effect. The buttons are also very similar to that of the Main Menu only now the play button resumes the playing session rather than initiates one. The “PREVIOUS BEST:” text will display the players High-Score that will reflect that of the value displayed within the ‘Playing’ game state to the right of the live score as explained in the appropriate game state previously; this feature is always visible and is the original way to view the players High-Score before the implementation of the extra parameter onto the live score element in the ‘Playing’ game state, although this element within the ‘Pause’ game state was still maintained it would not be in the players best interest to be required to pause the game when the wish to view their previous High-Score. The “PERSONAL BEST” line on the road is still visible within the ‘Pause’ game state, and operates under the same rules as that of mentioned previously and is only drawn on the occasion the player actually has a High-Score which would be a value greater than 0.

4.6.5 Game Over

The 'GameOver' game state is the only game state that cannot be directly navigated to, this game state occurs when the player eventually succumbs to the challenge of the game and gets 'hit' by a moving object.

```
// FOR EACH Moving Object in List of Moving Objects
foreach (MovingObject movingObj in lstMovingObject)
{
        // IF Moving Object Intersects with Player
        if (movingObj.Bounds.Intersects(playerCharacter.Bounds))
        {
                // SET the Current Game State to Game Over
                CurrentGameState = GameState.GameOver;
        }
}
```

A check is carried out in the update method that goes through every object in the moving object list and, checks if any of the moving objects come in contact with the player. Eventually when the player bounds intersect with a moving object bounds, the game state is instantly changed to 'GameOver' and the mechanics and draw methods are instantly switched to comply with that change. This process is the same across both revisions of the game, however the elements and available features that are presented to the player drastically change between the two revisions.



The above image is a screenshot of the 'GameOver' game state taken from the first revision of the game.

As can be seen from the above image there is very little difference between the previous game state 'Playing'. When initialised the 'GameOver' game state acts as a pause and stops all moving elements, the game world stops and the player is no longer able to move. The only real addition to the window elements is the "Menu" button that is placed in the middle of the window underneath the Score. There is only one navigational choice available and that takes the user back to the 'MainMenu' game state where the user is prompted to make another chocie as to what they want to do.

The remaining element and design choices is the update in the layering of assets. The player asset is placed on top of the Moving Object layer and gives a visual representation of where the player got hit by the object. Lastly, the score element that used to be layered directly above the world background, gets moved in the layers and follows suit of the player asset and is placed above all moving objects to make it more promanant as game mechanics are not in play so the only real information that needs to be made available to the player is the result of the game session.

The above image is a screenshot of the 'GameOver' game state taken from the second revision of the game.

The second revision has a lot more to offer in comparison to the first revision not only does the statement hold true for the overall application, but the 'GameOver' game state has a lot more to offer to the player; although not instantly visible to the player. Once again there are elements that have a visibility variable, however unlike the previously described elements that only require a High-Score value greater than 0, these hidden elements are varying on the player actually acquiring a new High-Score.

Starting from the aesthetic design of the different User Interface elements, it can be easy to notice the style is once again maintained, following the colour code of the appropriate surroundings and themes of the game. There are again two buttons that are available to the user as always one is to exit the game, and another is titled "Retry" this button re-initialises the methods used in the game and places the user right back into the "Playing" game state completely skipping the Main Menu, for fast and easy access right back into the action of the game. Working similar to that of the 'Pause' game state the High-Score string displaying the players "PREVIOUS BEST:" value is positioned directly underneath he buttons, also visible in the background the High-Score lane visual representation element is maintained across the game states.

The final score element goes through a big change, of which the technique is not used anywhere else in the game. Firstly, the physical informative values that it provides no longer display the players High-Score in small font to the right of it as it does in the 'Playing' game session, due to the fact the High-Score is already presented with a further description. Furthermore, it should be noted that the placement is exactly that of the placement within the previous game state which would be 'Playing' so upon the switch of game states the score sees a smooth transition effect into what is visible in the images used in this section. The new design technique used with this element comes down to an adaptive colour scheme that is based on the progression and score of the player with the consideration of their previous High-Score value. This subtle feature might not be instantly noticeable to the user as it changes based on each game session. When the user enters this game state and they did not reach a new High-Score, a method is used to change the colour of the Score value presented, by determining how close the player came to the previous High-Score.

Should the player enter the 'GameOver' game state with a score that is within the range of 30% of their High-Score value, they will be presented with a final score element in the colour red.

The colour then changes to orange should the score be in the range of up to 70% of the High-Score to show progression.

The final colour turns green should the final score not beat the High-Score and not fall under the previously mentioned categories.

The above image is a screenshot of the 'GameOver' game state taken from the second revision of the game.

Should the player achieve a new High-Score the final; they are presented with a number of new elements and a new feature that allows the submission of the achieved High-Score. The Final score element displays the achieved score in a yellow colour to separate itself from the other score states as mentioned above, the text that displayed the "PREVIOUS BEST:" is changed to further inform the user of their achievement by now reading "NEW PERSONAL BEST!".

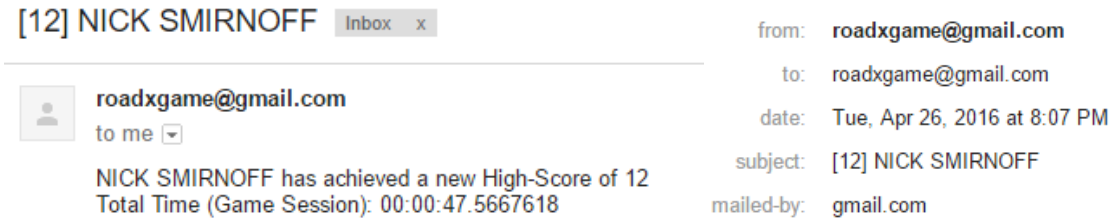
The High-Score submission feature with its elements are now visible and available to the user. In green colour it encourages the user to “TYPE YOUR NAME”, this element adapts to the users input and allows any keyboard character and space characters to appear of screen, naturally any mistakes can be deleted through the backspace key. Underneath where the user types their name, is an element that states how to submit “PRESS F1 TO SUBMIT YOUR SCORE!”. The submission will parse in 2 variables; the achieved score value and the chosen name typed in by the user, within the appropriate element mentioned above.

The submit High-Score feature works of the ‘F1’ key press and has a number of outcomes that is displayed to the user depending on the outcome of the submission. The outcome is really only dependent on the user having an internet connection, to allow the submission to be sent over the internet.

Should the user have a stable internet connection and choose to submit their High-Score, the try method should successfully carry out the submission method. The user will be notified on a successful submission, by seeing the bottom most element that informed the user on how to submit their score change to the status “SENT!” in a green colour.

In the case the user attempts to submit the score again after a successful submission, the status element will once again change to “YOU HAVE ALREADY SENT YOUR SCORE!” this will be displayed in red to deter the user from continuously attempting to submit the score. Without this counter measure the player will have the ability continuously submit the same score and essentially spam the submissions.

Should the try method occur an error, the catch method will alter the status element to display an error message that will contain the most common problem which is the requirement of an internet connection; “FAILED TO SEND – CHECK CONNECTION!”. The element changes to a red colour to alert the user that their action was unsuccessful, and should they still want to submit they have to do as suggested and check their connection.



Due to budgeting and legal security constraints, the submission is sent via E-Mail to a dedicated E-Mail address created for the purpose of maintaining player submissions. The method sends an E-Mail to its own address of ‘roadxgame@gmail.com’. The subject is made up of the submitted High-Score value contained within a set of brackets, followed by the name of the submitter. The content of the E-Mail has a particular format and displays 3 different variables achieved through the user client. The first variable is the contents of what the user typed in the name field element mentioned above, followed by standard text “has achieved a new High-Score value of”, this is where the High-Score value achieved by the submitter will be displayed. The last variable parameter is for further data on the play session and is not required within the actual submission, this displays how long the user had their client up and running before achieving and submitting their High-Score; this is set in the following format: hours-minutes-seconds with milliseconds. With access to the inbox a filter can be ran on the value of the High-Score and the greatest value can be determined; with that process it is possible to find out who submitted the highest score and how long they had their client up and running to do so.

4.6.6 Exit

The ‘Exit’ game state does not hold any drawn elements, on either of the revisions. The reason behind this game state is to carry out any necessities to save any data before closing the actual application. There is no window that is dedicated to this game state and visually the game instantly closes upon selecting the “Exit” button.

In the first revision of the game due to there being no external mechanics involved with the game process, and no further requirements needed to be made as far as saving the players score go, the ‘Exit’ game state simply closes the game. Within the second revision of the game, the High-Score is saved to a local file in binary format as to not be manipulated by the player. As a failsafe procedure, the program does a check and save of the data involved within that external file, to make sure the player never loses their High-Score progress the contents of the file is checked against the new player data available so the possibility of a new High-Score, is saved to the file to be loaded up on the next play through before the game actually closes.

4.7 Audio

Audio as planned the final feature implementation, to be carried out upon all physical game mechanics and elements completion, was developed with the second and final revision of the project. This section marked the final development stage of the project, using the already developed game mechanics and operations the process can be complete to integrate audio to each desired command that the game will process.

Extending the use of the Random Number Generator that as previously mentioned used for the spawning of objects, a single sound bite can be selected out of a library of similar sound clips, is used to play upon a character move command; this small library consists of 4 variations of a ‘Chicken Clucking’, with every move the player makes a random ‘cluck’ is played. Upon a collision with a moving object, a scenario that switches the game state to ‘GameOver’ a loud chicken sound is played to further symbolise the response. Furthermore, using the same technique every time, the user is on a road there is a chance of a ‘car horn’ audio clip being played which also has a library of 4 variations.

To further the aesthetics and overall feel of the game, audio is added when changing and selecting options of the buttons within the game and give a further audible tactile experience to of which the user benefits from. It is easily distinguished through the audio when a user makes a change in their selection of buttons and also upon the actual selection.

There is a small library of background music, which consists of 3 royalty free music tacks, that play in a loop during gameplay. This audio stream is able to be adapted to the user, commands are implemented that allow the user to skip the currently playing music track and also increase or decrease the sound level. Upon ‘GameOver’ game state a dedicated sound track is played, which stops and goes back to the original playlist upon starting the ‘Playing’ game state through the selection of the retry button.

4.8 Testing

Development testing was done throughout the project after each and even partial implementation of a feature. This relocated a considerable amount of time from the test plan and was distributed amongst the different development phases. A week was still allocated for testing, however due to the development testing already been complete with each step of implementation, this week was given for public tests and data to be gathered based on the week long experience with the application. Public tests were carried out upon the completion of the final revised build of the game with the purpose of finding bugs and glitches, as well as getting dedicated feedback on the project. The project was given out to a small group of 5; which volunteered for the role.

The directions given to the 'Testers' can be found in Appendix B. These outline the instructions of installing the necessary components that are required to run the game ("Microsoft XNA Framework Redistributable 4.0" and "Microsoft .NET Framework 4.5") and finally the physical direction which consist of extracting the contents of the .zip file provided and which file to run. Further notes where included that outlined some of the features and working of the game that allowed for in depth tests to be carried out.

The questionnaire survey and results are referenced in Appendix C and D respectively. This step was for the testers to give further information about their computer and personal experiences with the game and allowed the chance of submitting suggestions for further implementation of the game. This was advised to be carried out after a few days of having the application on their machines, allowing the testers to answer the questionnaire with the potential of having a moderate amount of experience and exposure to the game.

It can be seen from the results, that backup the claims of success made within this document, as the completion of the different aims and objectives have been proven to be unanimously true; the game was simple enough to grasp but engaging enough to keep the users playing and even on an above average basis. Comments made about the game show the game has the potential of being marketable, and already show promise for the future with good suggestions being made for further development of the project.

5 Evaluation

The initial time plan for the project has been followed a fair amount quicker than expected due to a small portion of the application and project being more finalized upon entering the different steps and issues that have not been foreseen fixed in methods that discontinued further support in that planned section; that is to say the project has become more optimized for time with each passing section to the point a working proto-type of the application has been achieved and the objectives have been met with each step.

2 Using the first revision of the Game place player on screen with movement
2.1 Added Scrollable Background which scrolls based on Player input via arrow keys
2.1.1 Added Boundary to stop the Player from moving out of the desired area of the screen
2.1.2 Background scrolls based on the character moving forward after hitting the top boundary
2.2 Added Moving Objects (Vehicles)
2.2.1 Added None Moving Objects (Bushes)
2.2.2 Added Different Game States (MainMenu, Playing, Paused, GameOver)
2.2.3 Tweaked Velocity and Speed of Vehicles in the Moving Object category
2.2.4 Changed the method of which none moving objects are maintained when the game is reset (player went through 'Retry')
2.2.5 Fixed bug in the method Bushes use to deal with collision which allowed players to get out of the world boundary by pressing combos of movements against bushes
2.2.6 Added Controls overlay to the Main Menu Game State
2.3 Added High-Score
2.3.1 High-Score has a read and write function to maintain value outside of the game
2.3.2 Added Controller support
2.3.3 Changed Controls Overlay to reflect Controller Support
2.3.4 Added hidden feature "Crtl + R" will reset High-Score (main purpose for testing but will keep within the game)
2.4 Added Extra Lanes
2.4.1 Filled extra Lanes with new Vehicle Lanes
2.4.2 Changed the Graphics of the different vehicles
2.4.3 Added feature where a lane has a chance of spawning a another type of vehicle
2.4.4 Tweaked Velocity and Speed of Vehicles in the Moving Object category
2.5 Added sound to the game
2.5.1 Added Background Music
2.5.2 Added Skip Background Music feature on Button press (Keyboard & Controller)
2.5.3 Added Volume Up and Down for the Background Music on Button press (Keyboard & Controller)
2.5.4 Changed Controls Overlay to reflect the relevant sound feature
2.6 Added a High-Score Submit feature (E-Mail is sent to a dedicated email address containing the submitted High-Score)
2.6.1 Added a 'Text-Box' feature that allows the Player to type their name before submitting their High-Score
2.6.2 The Player name is included in the E-Mail sent containing their High-Score
2.6.2 Added Try Catch method in case the user attempts to submit score without an internet connection or if some external problems come up the user will be notified
2.6.3 Added Total Time of Game Session to the E-Mail containing Players High-Score

For the time availability, a second revision of the game was developed with more in depth mechanics and features that allowed the project to excel in the planned aims and objectives. The biggest time optimization consisted of de-bugging/testing which had been going on passively with each coding step and thus resulting in a constant working revision step of the application.

The project made a shift of features and a few planned extra features that the writer had voiced in the initial report as to be unconfident about the implementation but the features should definitely be reviewed; has been accomplished and such features as additional live is too much of a gameplay change to implement that is to say it would not be difficult to implement but the implications of the feature would have too much of an impact on gameplay in a negative way, of which the concern has been there from the planning stage and has come to pass that the decision of not implementing lives and power-ups of such things for the greater experience and enjoyment for the player.

5.1 Project Achievements

The first objective of the project to find a balance in simplicity and engagement of the game with the player. Replay ability, keep the player engaged with different features that if they should fail they will want to replay the game to do better and progress further. This objective being a passive objective was the reasoning behind the majority of development and feature choices that were made when developing the application; to keep certain features away from the game and implement others. This objective has been achieved unanimously across all the game testers, this objective is the goal for the majority of application released in the world and with the conditions of achievement for this type of objective being very limited as to what the developer can influence it bring about an astronomical success to the project, managing to achieve this objective.

Achieving all the necessary chosen advanced an extra feature is worthy of an achievement for the project; adding extensive library that can be expanded on to create further objects in the future. However, managing to reach further and develop a means of connectivity between the players and have the ability to house a community around the simple application is required to be mentioned; as something as simple as a High-Score that was not even planned to be taken further and be connected through the internet provides a whole new level of completive play and achievement to the game.

5.2 Further Work

With extra time the project could be further developed in a number of ways, the first feature that could be implemented as requested by a tester; additional character variations. This feature would simple be a graphical texture variable and can be changed to that of the players choosing, furthermore this can be integrated with achievements thus allowing players to unlock new characters every time they unlock an achievement within the game.

With a budget, a server could be established and used to store High-Score data and have an on demand ready High-Score leader board table, to be available to the user, allowing them to view their rankings and the overall top players to compete with and aim for the top.

Ultimately the game can keep growing with an infinite possibilities of features that can be implemented to improve on the user experience, with the possibility of being accepted to one of the application market places that are available to the public today. With a best case scenario being listed in the stream database, allowing access to millions of users and simple ability to push updates the application can grow infinitely and carry on evolving based around the community, especially with the steam forums and feedback, implantation of features can revolve around what the players want within the application.

With reference to the “User Test Results” in Appendix D, further suggestions have been given by players that have experienced the game for a length of a week. Showing a small number of potential features that can be implemented within the application to further develop the experience and what the application has to offer. These suggestions brought on by the start of a player base for the application has the potential to grow the community that can be based around the game and increase the likelihood of being accepted and purchased from a marketable source, due to the positive interaction with the consumers.

6 Conclusion

In conclusion this reports used to outline the initial ideas, concepts and plans for the project. A project that strived to achieve the aims and objectives that have been planned from the beginning, one of which that carried out each stage of development with the objectives in mind. Has officially reach a revised working build, one that fits all of the criteria laid out from the start of the project, and has evolved past a point that was expected and planned.

Originally complying with the aims and objectives on a basic level, with basic mechanics that consisted of a static background that spanned across the size of the window, with generic car graphics used as moving objects which occasionally overlapped one another due to the spawn mechanics. Bush textures spawning as none moving objects that are randomly generated one time during the launch of the game and maintain their position, and finally simple score system that let the player repeat the same layout of obstacles consistently until, game over. Using that first revised build of the game as a stepping stone, achieving an astonishing result with the second and final revision of the game which adapted on the basic functions and features, to expand the user experience. With such features as a variety of graphics used for the moving objects which both vary in speed, size and quantity which over extends the second objective of the project. Further development in the application, allowing the use of the graphics to be cleverly adapted in the positioning and layout to be recycled in a way that doesn't make the game feel repetitive and very little influence on pattern like behaviours due to the integration with player mechanics that allow the user to have a forward motion of the entire window and move the elements placed on screen depending on the player progression, all contribute to the third and fourth objective. Finally, the addition of an extensive library of objects that can easily be increased, a High-Score system with internet capabilities, and a controller support withstand the test of the fifth objective that challenged the project to advanced features; more so to the point of completion of this objective, includes the fact some features were not even mentioned in the possibility of being introduced in the planning phase and were later developed and integrated within the project that allowed the object to excel further then expected.

With further hard decision made during development, ones that potentially could have jeopardised the experience that was strived with the features mentioned above. The consequences of which were thoroughly considered and declared to be too much of a risk of having a negative impact on the experience, reasoned against some of the initial concerns that were issued during planning stage with some advanced features, notably power up and extra lives which would of evolved the gaming experience drastically and foreseen to not be in a positive light; with the inclusion of which would have brought an unwanted element of luck and RNG to the game with the powers and allowed users to greatly boost their scores with mistakes through extra lives. Overall the correct decisions were made and the project progressed in the desired direction of which complies with all that was set out to be achieved and finished with additions to the expectation.

Appendix A: Initial Project Proposal Description

Project Description

The aim of this project is to create a 2d racing game or simulation in which computer players are simulated using flocking and steering algorithms to avoid each other and stay on the track. Computer controlled cars will modify their speed and direction in order to avoid collisions with other cars whilst travelling around the track as fast as possible, whilst adhering to the same limitations as would be imposed on any human player.

Appendix B: Developer Test Instructions

->REQUIREMENTS
1. "Microsoft XNA Framework Redistributable 4.0" - https://www.microsoft.com/en-us/download/details.aspx?id=20914
2. "Microsoft .NET Framework 4.5" - https://www.microsoft.com/en-b/download/details.aspx?id=30653

->HOW TO INSTALL
1. Extract contents from "[2.6.3]Road-X.zip".
2. Run "RoadX.exe".

-THINGS TO NOTE-
1. You start with no High-Score (as you should).
    - High-Score is presented in places of the screen as long as the High-Score value is over 0.
    - This means elements of the game are invisible on first/clean launch.
    - The High-Score is saved as binary in the same folder as the game files (folder you extracted the contents of the .zip to)
    - The High-Score file is "HighScore.dat".
    - Due to the implementation of the data being in binary it should be a lot harder to edit the information of this file.
    - On the "MAIN MENU" Pressing "Left Ctrl + R" will set the High-Score to 0.
    - Alternatively to reset the High-Score simply delete this file while the game is closed.
2. Submit High-Score Feature requires an internet connection as it sends an E-Mail to a dedicated E-Mail address which contains your input (Type Your Name) and the score achieved.
    - Your Personal E-Mail and information is not touched the E-Mail is all dedicated and sends it to itself, which I can access and see the scores.
3. There is no real end to the game you need to get as far as possible.
4. I will be creating a report based on this project.
    - Should you be reading this then your task of which you accepted would be to "Test" this current build of the game and provide (if any) feedback.
    - The General Feedback questionnaire will be provided.

Appendix C: User Test Form

Road-X

1. What is your Operating System?

2. How easy was it to Navigate the application?

1 - I'm Lost	2	3	4	5 - No Problem
○	○	○	○	○

3. How easy was it to understand the different Interface Elements and Layout used?

1 - There was buttons??	2	3	4	5 - No Problem
○	○	○	○	○

4. How often did you find yourself playing the game?

Once.	A few times.	I'm a regular.	More then average!	I'm addicted!
○	○	○	○	○

5. How long is the usual play session?

One attempt.	A few attempts. (0-10 mins)	Many attempts. (10-20 mins)	I need to beat my High-Score! (30-60 mins)	Send Help! (60+ mins)
○	○	○	○	○

6. In your opinion is this application marketable?

○ No

○ Yes

7. Any further suggestions or comments?

Done

Appendix D: User Test Results

Q1

What is your Operating System?

Answered: 5 Skipped: 0

Responses (5)

Showing 5 responses

windows 7

windows 7

Window 8.1

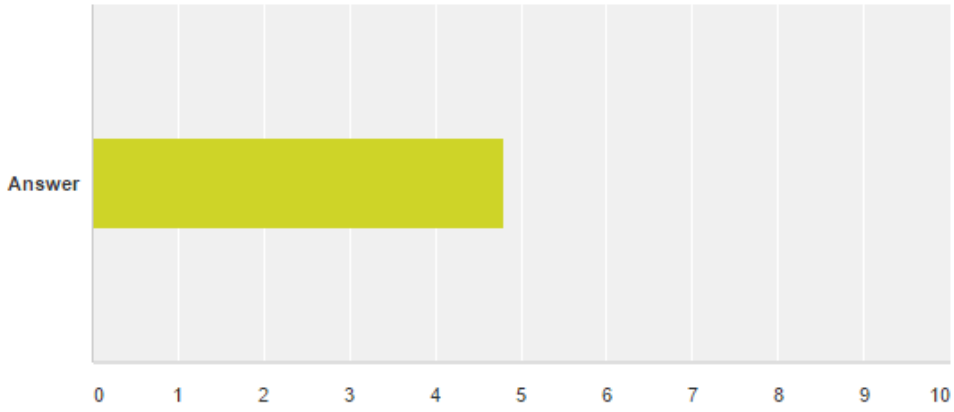
Windows 10

Windows 10

Q2

How easy was it to Navigate the application?

Answered: 5 Skipped: 0

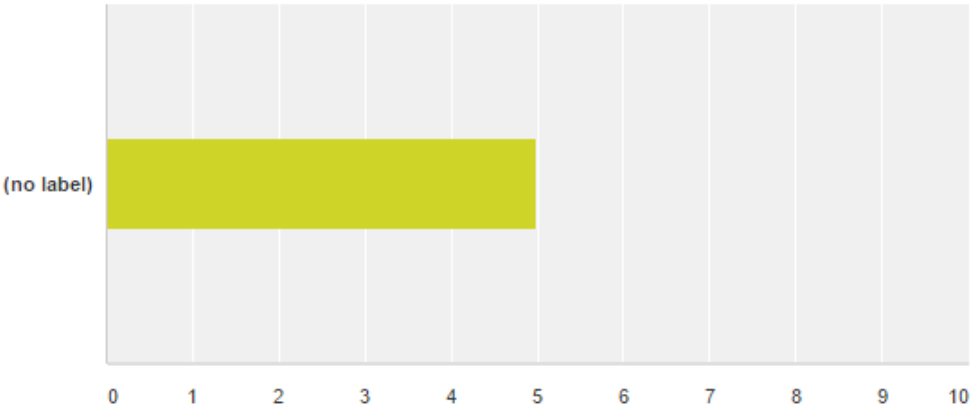


	1 - I'm Lost	2	3	4	5 - No Problem	Total	Weighted Average
Answer	0.00% 0	0.00% 0	0.00% 0	0.00% 0	100.00% 5	5	4.80

Q3

How easy was it to understand the different Interface Elements and Layout used?

Answered: 5 Skipped: 0

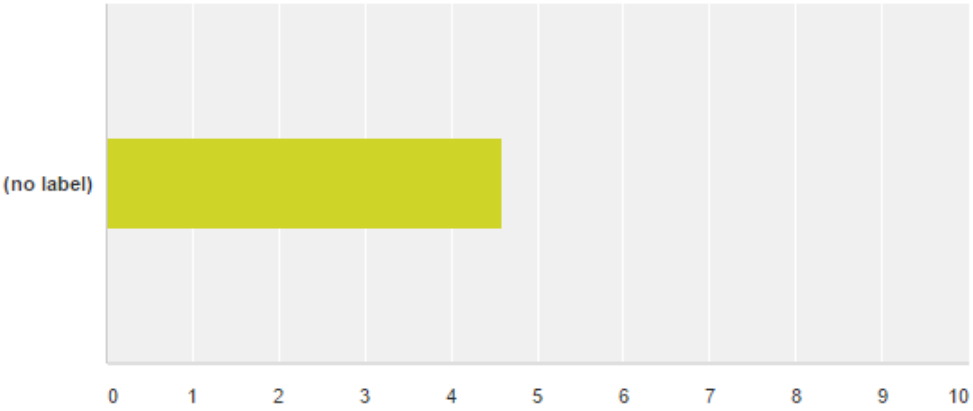


	1 - There was buttons??	2	3	4	5 - No Problem	Total	Weighted Average
(no label)	0.00% 0	0.00% 0	0.00% 0	0.00% 0	100.00% 5	5	5.00

Q4

How often did you find yourself playing the game?

Answered: 5 Skipped: 0

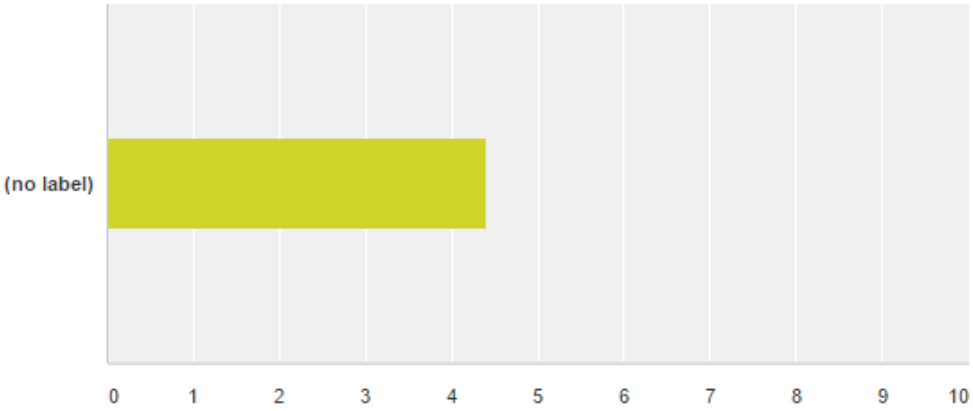


	Once.	A few times.	I'm a regular.	More then average!	I'm addicted!	Total	Weighted Average
(no label)	0.00% 0	0.00% 0	0.00% 0	40.00% 2	60.00% 3	5	4.60

Q5

How long is the usual play session?

Answered: 5 Skipped: 0



	One attempt.	A few attempts. (0-10 mins)	Many attempts. (10-20 mins)	I need to beat my High-Score! (30-60 mins)	Send Help! (60+ mins)	Total	Weighted Average
(no label)	0.00% 0	0.00% 0	0.00% 0	60.00% 3	40.00% 2	5	4.40

Q6

In your opinion is this application marketable?

Answered: 5 Skipped: 0

Answer Choices	Responses	
No	0.00%	0
Yes	100.00%	5

Responses

Responses (5)

Showing 5 responses

app stores £2.49

Steam £2

Chrome store £2

steam £1.50

Steam ~£3


Q7
Export
Any further suggestions or comments?
Answered: 4 Skipped: 1
Responses (4)
Showing 4 responses
new background/locations
add character unlocks
add achievments
View able High-Scores.

Appendix E: Time Plan

		University Calendar Weeks																
#	Task Name	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
1	Research																		
2	Architecture																		
3	Graphics Assets																		
4	Testing																		
5	Application Form																		
6	Mechanics																		
7	Interim report																D		
8	Gameplay																		
9	Extra Features																		
10	Final Report																		
11	Demonstration																		

References

Anon, 2014. *Crazy Taxi.* [Online]
Available at: http://www.knowledgeadventure.com/games/crazy-taxi/
[Accessed October 7 2015].
Holland, C., 2013. *Steering/Flocking Racing Game.* [Online]
Available at: http://www.s174768869.websitehome.co.uk/CraigWebsite/steeringflocking-racing-game/
[Accessed 2015 October 7].
Holt, C., 2014. *Behind the success of Temple Run.* [Online]
Available at: http://www.techhive.com/article/2361426/behind-the-success-of-temple-run.html
[Accessed 2015 October 10].
Microsoft, 2008. *Download Microsoft XNA Framework Redistributable 4.0 from Official Microsoft Download Center.* [Online]
Available at: https://www.microsoft.com/en-gb/download/details.aspx?id=20914
[Accessed 19 01 2016].
Rivello, S. A., 2011. *Understanding game development with Flash technologies.* [Online]
Available at: http://www.adobe.com/devnet/games/articles/getting-started-flash-games.html
[Accessed 10 October 2015].
Rouse, M., 2008. *What is XNA Game Studio?.* [Online]
Available at: http://whatis.techtarget.com/definition/XNA-Game-Studio
[Accessed 7 October 2015].
Team, M., 2014. *About | MonoGame.* [Online]
Available at: http://www.monogame.net/about/
[Accessed 1 February 2016].