Chapter 1

INTRODUCTION

1.1 Introduction

The gameplay resembles a driving game, featuring a car similar in appearance to a Ferrari Testarossa, referred to in the game as an expensive "sports car". The screen shows a first person perspective from inside the car, through the windshield. To separate it from other driving titles of that era, stunt loops and other road hazards were added. The game generally consists of 1 or 2 laps around the stunt track. In certain modes, if the player scored in the top 10, the player races against the computer controlled car, Phantom Photon. In this race, it was possible to race the wrong way around the course and beat the Phantom Photon across the start-finish line. The game challenges the players in a daredevil fashion and broke away from traditional racing games like Pole Position. It was also one of the first games to allow for more than three initials on the high-score board; which enterprising drivers could use to their advantage to construct sentences during the course of game-play.

It also features a realistic manual transmission mode (including a clutch pedal and the possibility of stalling the car should one mis-shift) and force feedback steering wheel, in which the driver would have to properly operate the car as they would in real life.

A notable feature of the game is the "instant replay" display that is presented after a crash, which sets Ultimate Racer apart from most driving games of its time, which after a crash would just put the player back on the road, stopped, and let them accelerate again. Before resuming play after a crash, Ultimate Racer would run an approximately ten second animation, captioned "Instant Replay", which showed a wide aerial view of the movements of the player's car and surrounding vehicles leading up to the crash, with the player's car always centered on the screen. During the replay, the player could not change the action on screen, but the replay could be aborted to immediately get back to active gameplay. The replay would continue for about two or three seconds after the crash, showing a polygon-rendered fireball and the movement of the car, including any spinning, flipping, or bouncing off the struck obstacle. The replays add to the appeal of the game and actually add a motivation to crash in spectacular ways in order to see them played out from the aerial view.

Besides collisions, a non-survivable landing after going airborne (even if the car landed right-side up), or even going too far off-road, could cause a crash which would be replayed like any other crash, with the car even exploding into the same orange fireball. The game tracks the player's progress around the track by invisible waypoints (denoted by flags on the course map showing the player's progress when the game ends due to time running out), and after a crash, the car is placed back on the track at the last waypoint passed; this sometimes is a significant distance back from the point of collision. (One of the waypoints on each track was the marked checkpoint about halfway around, which when passed granted the player extra time.)

Working with these requirements, we decided to use Unity 3D as our platform to develop our 3D game with. This decision was made with regard to that the platform had many in-built tools and provided a good framework for us to get started with the development as fast as possible. The fact that Unity 3D also used javascript as development language was also in consideration, since we wanted to learn this newly developed javascript language. Arcade-style racing games put fun and a fast-paced experience above all else, as cars usually compete in unique ways. A key feature of arcade-style racers that specifically distinguishes them from simulation racers is their far more liberal physics. Whereas in real racing (and subsequently, the simulation equivalents) the driver must reduce their speed significantly to take most turns, arcade-style racing games generally encourage the player to "power slide" the car to allow the player to keep up their speed by drifting through a turn. Collisions with other racers, track obstacles, or traffic vehicles are usually much more exaggerated than simulation racers as well. For the most part, arcade-style racers simply remove the precision and rigor required from the simulation experience and focus strictly on the racing element itself. They often license real cars and leagues, but are equally open to more exotic settings and vehicles. Races take place on highways, windy roads, or in cities; they can be multiple-lap circuits or point-to-point, with one or multiple paths (sometimes with checkpoints), or other types of competition, like demolition derby, jumping, or testing driving skills. Popular arcade-style racers include the Need for Speed series, the Ridge Racer series, the Daytona USA series, the Sega Rally series, the Rush series, the Cruis'n series, the Midnight Club series, the Burnout series, the Out Run and MotorStorm series.

During the mid-late 2000s there was a trend of new street racing; imitating the import scene, one can tune sport compacts and sports cars and race them on the streets. The most

widely known ones are the Midnight Club 3: DUB Edition and the Midnight Club series, certain entries in the Need for Speed series, Initial *D* series, and the Juiced series.

Some arcade-style racing games increase the competition between racers by adding weapons that can be used against opponents to slow them down or otherwise impede their progress so they can be passed. This is a staple feature in kart racing games such as the Mario Kart series, but this kind of game mechanic also appears in standard, car-based racing games as well. Weapons can range from projectile attacks to traps as well as non-combative items like speed boosts. Weapon-based racing games include games such as Full Auto, Rumble Racing, and Blur.

Chapter 2

Review of Literature and Definition of Problem

2.1 Features of Unity

- Unity 3D comes loaded with a ton of professional tools for both programmers and artists.
- Unity provides a workspace that combines artist-friendly tools with a component-driven design that makes game development pretty darn intuitive.
- Both 2D and 3D development is possible in Unity, with 2D physics handled by the popular Box2D engine.
- Unity uses a component-based approach to game dev revolving around prefabs. With prefabs, game designers can build objects and environments more efficiently and scale faster.
- With powerful shaders, physics-based materials, post-processing, and high-resolution lighting systems, Unity can deliver impressive graphics across the board.
- Everyone from Ubisoft to NASA is utilizing Unity's VR technology too.
- For the platform itself: Unity was built in C++ and optimized over the years for performance. Premium users will have access to Unity's source code for even greater opportunities.
- Cross-platform deployment is a major draw for today's developers and Unity shines in this area. With support for every major console and operating system, games developed in Unity can be deployed to absolutely any platform.
- With Unity's editor tools you can simultaneously handle inputs for mice, keyboards, and game controllers.
- There's also some pretty strong support for cloud-based solutions for multiplayer games with server hosting and scalable matchmaking, making it an all-in-one solution to multiplier experiences.
- Team collaboration has been greatly improved in the newer versions of Unity. Built-in version control and cloud integration make working with others easier than ever before.

- And Unity has a customizable editor with full API support for building your own editor tools and scripts. Make almost any tool you want to have for Unity, with Unity.
- And it's definitely worth mentioning the asset store which contains thousands of models, scripts, scenes, materials, and everything else you could want. You can even sell your own assets on the Unity store.
- Truly one of the strongest communities you'll find in the 3D space.

2.2 Features of C#

C# is one of the most acceptable, organized and popular programming language in the world of programming. C# has been recognized as one of the most influential and powerful programming language. C# is one of the compatible languages. It completes the tasks easily and has a smooth running. In this article, I would take a look at the advantages of C# over other languages:

- **Object- Oriented Language**

  C# is pure object-oriented language, this allows you to create modular maintainable applications and reusable codes. This is one of the biggest advantages of C# over C++.
- **Automatic Garbage Collection**

  C# has got a very efficient system to erase and remove all the garbage present on the system. C# doesn't create a mess in the system and the systems do not get hanged during execution.
- **No Problem if Memory Leak**

  C# has a major advantage of a strong memory backup. There would be no problem of the memory leak and other such type of problems in the C# as it happens in the case of C++ language. In this case C# has a very clear edge on all other languages.
- **Easy-to-Development**

  The rich class libraries make many functions easy to be implemented. C# has influence on most of the programmers of the world and it has a history in the programming world.
- **Cross Platform**

  Your application will run well only if the machine has installed the NET framework. This is the most important requirement for the C#. also this could be an important opportunity for the young programmers to get them trained with .NET framework

- **Better Integration**

  Applications written in .NET will have better integration and interpretability with other NET Technologies. Actually C# runs on CLR, making it easy to integrate with components written in other languages

- **More Legible Coding**

  Formalized concept of get-set methods, so the codes becomes more legible. Also in C#, you don't need to worry about header files. Coding would be a worth to do in C#.

- **Backward Compatibility**

  .NET applications work on windows platforms only and Microsoft keeps retiring support for old windows platforms. So always there would be a need to upgrade your .NET framework if you are going on a new version of the windows. This could be an advantage or a disadvantage as well. A need to always improve gives you the motivation to work hard and excel in your field. I think this is a good thing.

Chapter 3

Materials and Methods

3.1 C#

C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language.
In January 1999, Dutch software engineer Anders Hejlsberg formed a team to develop C# as a complement to Microsoft's NET framework. Initially, C# was developed as C-Like Object Oriented Language (Cool). The actual name was changed to avert potential trademark issues. In January 2000, NET was released as C#. Its NET framework promotes multiple Web technologies.
The term is sometimes spelled as C Sharp or C-Sharp.
The term's # character derives its name from the musical sharp key, which denotes a one semitone pitch increase. C# is pronounced "see sharp."
C# improved and updated many C and C++ features, including the following:

- C# has a strict Boolean data variable type, such as bool, whereas C++ bool variable types may be returned as integers or pointers to avoid common programming errors.
- C# automatically manages inaccessible object memory using a garbage collector, which eliminates developer concerns and memory leaks.
- C# type is safer than C++ and has safe default conversions only (for example, integer widening), which are implemented during compile or runtime.

No implicit conversions between Booleans, enumeration members and integers (other than 0) may be converted to an enumerated type. User-defined conversions must be specified as explicit or implicit, versus the C++ default implicit conversion operators and copy constructors.

3.2 Unity

Unity is, simply put, the world's most popular game engine. It packs a ton of features together and is flexible enough to make almost any game you can imagine. With unrivaled

cross-platform features, Unity is popular with both hobby developers and AAA studios. It's been used to create games like Pokemon Go, Heathstone, Rimworld, Cuphead, and plenty more.

While 3D is in the name, Unity 3D also packs tools for 2D game development. Programmers love it because of the C# scripting API and built-in Visual Studio integration. Unity also offers JavaScript as a scripting language and MonoDevelop as an IDE to those who want an alternative to Visual Studio. But artists love it as well since it comes with powerful animation tools that make it simple to create your own 3D cutscenes or build 2D animations from scratch. Nearly anything can be animated in Unity.

Also Unity 3D offers a free version so developers can release games made with Unity Personal without paying for the software, so long as they make less than $100,000 from games made with Unity. For those willing to pay, Unity offers some extra features and a flexible licensing plan under a tiered subscription model. Premium users will have access to Unity's source code and developer support as well. Because Unity has been around since 2005 it has developed a massive following of users and an amazing library of resources. Not only does Unity have fantastic documentation, but the sheer wealth of videos & tutorials online is staggering.

3.3 Mono-Develop

MonoDevelop is an IDE primarily designed for C# and other .NET languages. MonoDevelop enables developers to quickly write desktop and ASP.NET Web applications on Linux, Windows and Mac OSX. MonoDevelop makes it easy for developers to port .NET applications created with Visual Studio to Linux and Mac OSX and to maintain a single code base for all platforms.

Features of Mono-Develop: -

- Multi-platform IDE and user projects (Linux, Windows and macOS)
- Multi-language (C#, F#, Visual Basic .NET, C/C++, Vala, JavaScript, Typescript)
- Project templates for C#, Visual Basic, Boo, Java (IKVM) and C/C++
- Code completion support for C#, code templates, code folding
- Customizable window layouts, user defined key bindings, external tools
- Integrated Debugger for debugging Mono and native applications
- Integrated Compiler (supports up to C# 6.0)

Chapter 4

Proposed Work

Game development is the job of very experienced and skillful programmers. It can cost hundreds of millions of dollars. It is a very creative work which also demands technical proficiency. They need programming languages with specific needs to work with.

- **C++**

  It should not come as surprising seeing C++ as the best programming language for games in almost every guide you read. The pioneer of modern game programming languages, C++ adds the concept of Object Oriented programming(OOP) onto its predecessor C. The ability to control very low-level system components has contributed a very fast running time for C++ programs, a much-needed element in game programming. Most high-end games that you play today depends on C++ codes in one way or the other. Popular gaming consoles such as the Xbox and PlayStation both utilize this game programming language heavily. C++ is the must-know language if you anticipate yourself developing futuristic games.

- **C#**

  One of the best video game programming languages, a thorough knowledge of C# is elemental to every game programmer out there. It's often the first preference to many developers to learn C# over other game programming languages due to the high-level of convenience it offers. The Microsoft-borne language supports the infamous Unity3D, one of the best game engines currently used in industry. C# gives developers the ability to build games of any type, for any architecture without any extra hassles. The language is also far easier to learn than C++. So, C# is one of the best programming languages for games you'd want to learn.

- **Java**

  Java takes a prominent role in the industry and is one of the best programming languages for games, and for good reasons. It uses the same OOP principle utilized by C++ but offers a broader range of systems to play for. Java codes typically run on the Java Virtual Machine(JVM) and translate into generic bytecodes, that can be executed on any system. So, Java is one of those few game programming languages that gives developers the ability to develop games for any given system. It's also

one of the primary language used for developing Android games, thanks to an increasing number of open source third-party modules like LWJGL.

- **Python**

  One of the most straightforward yet most versatile programming languages you can get your hands on today is Python. However, do not misinterpret its seemingly easy syntax and semantics inappropriate as a game programming language. Python offers full-fledged OOP (Object Oriented Programming) techniques to developers, just like C++ and Java. The Pygame framework, based on this popular language is increasing in momentum every day, thanks to its ability to let developers prototype their games insanely faster. Python is gaining its share of glory as one of the best video game programming languages for games, and it certainly wouldn't hurt you to learn it right now.

- **Objective C**

  This is the best programming language for games on the iOS platform. Objective C blends the syntax of Smalltalk and C together and offers convenient solutions for building iOS games. It also comes with the ability to program in an Object Oriented approach, much required for faster rendering time. It powers game engines like Core3D, Cocos2D and enables C developers to develop their games in an OOP style. Although not suitable for the most futuristic games, Objective C is a pretty viable choice if you want to get your hands dirty with an easy-to-grasp game design programming language.

- **JavaScript**

  Although it was not meant for developing large-scale games, JavaScript is turning the convention as days passes. It's one of the most-used languages of the web and integrates pretty easily with any web applications. As we continue to progress more and more towards a web-based industry, online games are becoming much familiar every day. JavaScript is definitely the best video game programming language for games when it comes to building interactive online games. The ability to integrate JavaScript codes easily with conventional web technologies like HTML and CSS are also contributing rapidly to an increasing number of cross-platform mobile games.

- **HTML5**

  Despite your struggle to believe, HTML5 has become one of the most common game programming languages for the web. A vast majority of mobile games that you play today utilizes this markup language. You can easily create a highly sophisticated web-based game blending HTML5 with JavaScript for interactivity and WebGL for graphical components.

Developing software applications is a time-consuming process, and with time-consuming processes come high costs. During the last years, several software development methodologies, often known as *agile software development*, have become widely used by software developers to address this issue. Many different development methodologies can be more or less good, depending of the task and application type.

One of the software development methodologies is the *evolutionary software method*, which, as the name hints, takes on an evolutionary approach to the problem, and allows the project to evolve through different stages of the project. Our case study will show how well this evolutionary approach worked on our project where we choose to develop a 3D graphic computer game. Some requirements for the computer game were given from the beginning, such as:

- **3D graphics** – The game must contain 3D models, and render these in the game. 3D environments were never a requirement, and platform games with 2D environment could still open up for 3D objects.
- **Impressive result –** The game result must impress whoever plays the game. It should last long, and make the players come back and play it over and over again.
- **Graphical effects** – To achieve an impressive result, we would need to add modern graphical effects, such as real-time rendered soft shadows, motion blur, and ambient occlusion.

Working with these requirements, we decided to use Unity 3D as our platform to develop our 3D game with. This decision was made with regard to that the platform had many in-built tools and provided a good framework for us to get started with the development as fast as possible. The fact that Unity 3D also used javascript as development language was also in consideration, since we wanted to learn this newly developed javascript language. Arcade-style racing games put fun and a fast-paced experience above all else, as cars usually compete in unique ways. A key feature of arcade-style racers that specifically distinguishes them from simulation racers is their far more liberal physics. Whereas in real

racing (and subsequently, the simulation equivalents) the driver must reduce their speed significantly to take most turns, arcade-style racing games generally encourage the player to "power slide" the car to allow the player to keep up their speed by drifting through a turn. Collisions with other racers, track obstacles, or traffic vehicles are usually much more exaggerated than simulation racers as well. For the most part, arcade-style racers simply remove the precision and rigor required from the simulation experience and focus strictly on the racing element itself. They often license real cars and leagues, but are equally open to more exotic settings and vehicles. Races take place on highways, windy roads, or in cities; they can be multiple-lap circuits or point-to-point, with one or multiple paths (sometimes with checkpoints), or other types of competition, like demolition derby, jumping, or testing driving skills. Popular arcade-style racers include the Need for Speed series, the Ridge Racer series, the Daytona USA series, the Sega Rally series, the Rush series, the Cruis'n series, the Midnight Club series, the Burnout series, the Out Run and MotorStorm series.

Figure 4.1: Midnight Club

The game generally consists of 1 or 2 laps around the stunt track. In certain modes, if the player scored in the top 10, the player races against the computer controlled car, Phantom Photon. In this race, it was possible to race the wrong way around the course and beat the Phantom Photon across the start-finish line. The game challenges the players in a daredevil fashion and broke away from traditional racing games like Pole Position. It was also one of the first games to allow for more than three initials on the high-score board; which enterprising drivers could use to their advantage to construct sentences during the course of game-play.

It also features a realistic manual transmission mode (including a clutch pedal and the possibility of stalling the car should one mis-shift) and force feedback steering wheel, in which the driver would have to properly operate the car as they would in real life.

A notable feature of the game is the "instant replay" display that is presented after a crash, which sets Ultimate Racer apart from most driving games of its time, which after a crash would just put the player back on the road, stopped, and let them accelerate again. Before resuming play after a crash, Ultimate Racer would run an approximately ten second animation, captioned "Instant Replay", which showed a wide aerial view of the movements of the player's car and surrounding vehicles leading up to the crash, with the player's car always centered on the screen. During the replay, the player could not change the action on screen, but the replay could be aborted to immediately get back to active gameplay. The replay would continue for about two or three seconds after the crash, showing a polygon-rendered fireball and the movement of the car, including any spinning, flipping, or bouncing off the struck obstacle. The replays add to the appeal of the game and actually add a motivation to crash in spectacular ways in order to see them played out from the aerial view.

Besides collisions, a non-survivable landing after going airborne (even if the car landed right-side up), or even going too far off-road, could cause a crash which would be replayed like any other crash, with the car even exploding into the same orange fireball. The game tracks the player's progress around the track by invisible waypoints (denoted by flags on the course map showing the player's progress when the game ends due to time running out), and after a crash, the car is placed back on the track at the last waypoint passed; this sometimes is a significant distance back from the point of collision. (One of the waypoints on each track was the marked checkpoint about halfway around, which when passed granted the player extra time.)

Unity is, simply put, the world’s most popular game engine. It packs a ton of features together and is flexible enough to make almost any game you can imagine. With unrivaled cross-platform features, Unity is popular with both hobby developers and AAA studios. It’s been used to create games like Pokemon Go, Heathstone, Rimworld, Cuphead, and plenty more.
While 3D is in the name, Unity 3D also packs tools for 2D game development. Programmers love it because of the C# scripting API and built-in Visual Studio integration.

Unity also offers JavaScript as a scripting language and MonoDevelop as an IDE to those who want an alternative to Visual Studio. But artists love it as well since it comes with powerful animation tools that make it simple to create your own 3D cutscenes or build 2D animations from scratch. Nearly anything can be animated in Unity.
Also Unity 3D offers a free version so developers can release games made with Unity Personal without paying for the software, so long as they make less than $100,000 from games made with Unity. For those willing to pay, Unity offers some extra features and a flexible licensing plan under a tiered subscription model. Premium users will have access to Unity's source code and developer support as well. Because Unity has been around since 2005 it has developed a massive following of users and an amazing library of resources. Not only does Unity have fantastic documentation, but the sheer wealth of videos & tutorials online is staggering.

Working with these requirements, we decided to use Unity 3D as our platform to develop our 3D game with. This decision was made with regard to that the platform had many in-built tools and provided a good framework for us to get started with the development as fast as possible. The fact that Unity 3D also used javascript as development language was also in consideration, since we wanted to learn this newly developed javascript language. Arcade-style racing games put fun and a fast-paced experience above all else, as cars usually compete in unique ways. A key feature of arcade-style racers that specifically distinguishes them from simulation racers is their far more liberal physics. Whereas in real racing (and subsequently, the simulation equivalents) the driver must reduce their speed significantly to take most turns, arcade-style racing games generally encourage the player to "power slide" the car to allow the player to keep up their speed by drifting through a turn. Collisions with other racers, track obstacles, or traffic vehicles are usually much more exaggerated than simulation racers as well. For the most part, arcade-style racers simply remove the precision and rigor required from the simulation experience and focus strictly on the racing element itself. They often license real cars and leagues, but are equally open to more exotic settings and vehicles. Races take place on highways, windy roads, or in cities; they can be multiple-lap circuits or point-to-point, with one or multiple paths (sometimes with checkpoints), or other types of competition, like demolition derby, jumping, or testing driving skills. Popular arcade-style racers include the Need for Speed series, the Ridge Racer series, the Daytona USA series, the Sega Rally series, the Rush

series, the Cruis'n series, the Midnight Club series, the Burnout series, the Out Run and MotorStorm series.

During the mid-late 2000s there was a trend of new street racing; imitating the import scene, one can tune sport compacts and sports cars and race them on the streets. The most widely known ones are the Midnight Club 3: DUB Edition and the Midnight Club series, certain entries in the Need for Speed series, Initial *D* series, and the Juiced series.

Some arcade-style racing games increase the competition between racers by adding weapons that can be used against opponents to slow them down or otherwise impede their progress so they can be passed. This is a staple feature in kart racing games such as the Mario Kart series, but this kind of game mechanic also appears in standard, car-based racing games as well. Weapons can range from projectile attacks to traps as well as non-combative items like speed boosts. Weapon-based racing games include games such as Full Auto, Rumble Racing, and Blur.

C# is one of the most acceptable, organized and popular programming language in the world of programming. C# has been recognized as one of the most influential and powerful programming language. C# is one of the compatible languages. It completes the tasks easily and has a smooth running. In this article, I would take a look at the advantages of C# over other languages:

C# is pure object-oriented language, this allows you to create modular maintainable applications and reusable codes. This is one of the biggest advantages of C# over C++. C# has got a very efficient system to erase and remove all the garbage present on the system. C# doesn't create a mess in the system and the systems do not get hanged during execution. C# has a major advantage of a strong memory backup. There would be no problem of the memory leak and other such type of problems in the C# as it happens in the case of C++ language. In this case C# has a very clear edge on all other languages. The rich class libraries make many functions easy to be implemented. C# has influence on most of the programmers of the world and it has a history in the programming world. Your application will run well only if the machine has installed the NET framework. This is the most important requirement for the C#. also this could be an important opportunity for the young programmers to get them trained with .NET framework.

During the mid-late 2000s there was a trend of new street racing; imitating the import scene, one can tune sport compacts and sports cars and race them on the streets. The most

widely known ones are the Midnight Club 3: DUB Edition and the Midnight Club series, certain entries in the Need for Speed series, Initial *D* series, and the Juiced series.

Some arcade-style racing games increase the competition between racers by adding weapons that can be used against opponents to slow them down or otherwise impede their progress so they can be passed. This is a staple feature in kart racing games such as the Mario Kart series, but this kind of game mechanic also appears in standard, car-based racing games as well. Weapons can range from projectile attacks to traps as well as non-combative items like speed boosts. Weapon-based racing games include games such as Full Auto, Rumble Racing, and Blur.

Figure 4.2: Overdrive