



DISSERTATION

CO3202 Entrepreneurial Project 2022-23 Y

Abstract

This document is the dissertation for CO3202. It will explore the process of the project from start to finish.

Cooper, Tom C.
[Email address]

Contents

1 Introduction	3
1.2 Aims.....	3
1.3 Objectives.....	3
1.4 Challenges	4
1.5 Planning and Timescale.....	4
2 Survey of Literature/Information Sources.....	5
2.1 Existing Games	5
2.2 Background Research.....	6
3 Requirements.....	7
3.1 Essential	7
3.2 Preferable.....	7
3.3 Optional.....	8
4 Business Development.....	8
4.1 Target Audience and Customers	8
4.2 Lean Canvas.....	9
4.3 Unique Value Proposition (UVP)	10
4.4 The Market.....	10
4.4.1 Market Size.....	10
4.4.2 Ways to Market	11
4.4.2 Competition	12
4.5 Revenue Model	13
4.5.1 Monetisation	13
4.5.2 Revenues and Expenses	13
5 System Design	15
5.1 Software Architecture	15
5.2 Technologies.....	16
5.3 System Overview.....	17
6 System Implementation	18
6.1 Modular Programming.....	18
6.2 Game Initialization and Setup	19
6.3 Main	20
6.3.1 Main Menu.....	20
6.3.3 Game Loop	20
6.3.4 Screen Rendering and Game Clock	21
7 System Testing.....	21

7.1 Testing Table.....	21
8 Critical Appraisal	24
8.1 Objectives.....	24
8.2 Social, Commercial and Economic Context	25
8.3 Future Work	25
8.4 Personal Development	25
9 Conclusion.....	26
References.....	27
10 Appendix	30
10.1	30
10.2	30
10.3	31
10.4	33
10.5	33
10.6	35

1 Introduction

Simulation games (sim games) are rising in popularity with both new and existing gamers in the PC space. This can be seen with the global market estimated at \$4.86 billion, expected to rise to \$20.76 billion by 2030 [1]. As the market is already large for sim games, it is a safe assumption that this project will not necessarily be creating anything novel. However, creating a sim game offers a significant and interesting challenge. A tech blog outlines some interesting demographics concerning games in general. They explain that most gamers are between 18 and 24 and generally male [2].

The project's development from start to finish will be investigated during this dissertation.

1.2 Aims

The aim of this project is to create a sim game that can be accessed for any user, irrelevant of skill level. It will offer a basic gaming experience that should be enjoyable for the user. It is primarily accessible and compatible with Windows PC's. The game will be focused on Formula 1 motorsport, although with simple changes it can be applied to different motorsports both fictional and non-fictional. Features of the game include selecting a desired driver changing values associated with the driver and car to enjoy experimenting with how they affect lap time. The game is written in Python and uses the PyGame and PyGameMenu frameworks/libraries to facilitate the Graphical User Interface (GUI) and the event logging/handling. The game has no online capabilities and is run from the Windows operating system (OS).

1.3 Objectives

When undertaking this project, it is important to evaluate what the specific objectives of the game and project are. This will aid in an accurate understanding of progress of the project and concluding how well the project has met what was set out in the beginning.

Objective 1 – Implementing a GUI.

To ensure that the game is enjoyable at the most basic level, it is important to implement a GUI that is easy to navigate and is clear for the user. GUI's and the User Experience are a critical component of an enjoyable experience in gaming. No matter how good the mechanics are in a game, if they are presented in a way that prevents the user fully realising them, they cannot be enjoyed.

Objective 2 – Packaging the game.

Games that exist on the market today are mostly easy to access and rarely require any additional software or requirements. This can be seen in game management tools and stores such as Steam. Therefore, an important objective is that the game should be able to function independently of any other software.

Objective 3 – Allow customisation.

An important part of a sim game is to remove constraints and allow customisation in the simulated environment. This is because enjoyment of a sim game can be experiencing scenarios that would not be possible in the real world. The game Totally Accurate Battle Simulator is the epitome of this concept- the user can create their own "battle units" or customise existing units. As well as this, the constraints of the battle being simulated can be greatly changed for the intrigue of the user. Evidence supporting the claim that users want this customisation can be seen in the review statistics of the store page for Totally Accurate Battle Simulator where almost 100,000 users rate this game as "Overwhelmingly Positive" [3].

Objective 4 – Realism

Perhaps one of the hardest objectives to achieve. For this objective, users should have a gaming experience that is close to real life as possible. Although, this may be out the scope of the project. Specifically, the physics simulation poses the highest challenge.

Objective 5 – Licensing

Like realism, the objective of licensing is where real life teams, drivers, and car are used. The sim game Microsoft Flight Simulator shows how successful a well licensed game can be. The aviator enthusiast site “Aviator Insider” rates Microsoft Flight Simulator as one of the best offerings, due, in part to the number of real-life aircraft incorporated into the game [4]. However, is it important to realise that again, this may be out of the scope of the project as licensing can cost vast amounts of funds.

1.4 Challenges

1. The main technical challenge for the project is to ensure that the game is as realistic as possible. This is a large challenge as there are many aspects of simulation that can be incredibly difficult to implement. One of these is the physics. This can be overcome by using game engines, such as Unreal Engine which has a built-in physics simulation. However, the learning curve for using these frameworks is high and is out of the scope of the project. The solution that was used to overcome this challenge can be seen in section 8.1.
2. Leading on from physics realism, the realism to match the teams and drivers is also a large challenge to overcome. If this project were to be commercialised, the cost would be significant. The solution to this challenge can be seen in 8.1.
3. Despite what may be seen as a reasonably trivial challenge, the GUI in a game can be a hard to implement. This is due to the game loop and how the interface provided to the user is updated at a high rate of frequency. This differs, for example, from a database interface where changes to the GUI are much less frequent. The solution that was used can be seen in 8.1.
4. Whilst using Python can have advantages, such as personal knowledge, the language itself can be a limitation in writing the software requiring a high time resolution, such as a game. This is because Python is an interpreted language meaning that the code is compiled into bytecode, not source code. A result of this is that, for example, Python is natively unable to exploit all the CPU power available (although thread locking can be applied).

1.5 Planning and Timescale

To ensure that the project is completed on time, different milestones, mainly derived from the requirements were established. These can be seen in appendix 10.1 External factors have greatly impacted the progression of the project.

2 Survey of Literature/Information Sources

2.1 Existing Games

As discussed in 1.1, there is a thriving market and demand for sim games. Because of this, sub genres exist, specifically racing. This grants an opportunity to examine these games and discover their weaknesses and strengths, enabling a potential influence on the project.

The first, and arguably, most relevant game to be examined is “F1 Manager 22” by Frontier Developments. It is relevant as it is a prime example of the game that most resembles the project, at least in the fundamentals. It differs slightly as it is aimed around providing a macro management simulation, rather than one from the sole perspective of the driver. One of the major positives of this game is how much development time by professionals have been applied. This results in a game where there is high fidelity graphics. This can be seen in *figure 2.1.1* below.

Figure 2.1.1 [5]



Additionally, with the presumably high budget of the game, Frontier Developments were able to acquire the licensing from F1 themselves to effectively bring the real world into the sim. However, despite looking great, many reviews cite the poor artificial intelligence and shallow mechanics as a negative for the game [6]. The somewhat dampened reception of the game can also be seen in the low review count (and therefore presumably a low purchase count), at 6,866 reviews [6].

Additionally, the reviews are categorised overall as “Mostly Positive” [6], a stark difference from the “Overwhelmingly Positive” seen Total Accurate Battle Simulator [3].

Whilst the response from “F1 Manager 22” may indicate a low desire from users to play a motorsport sim game, there are other games in the genre that garner a higher level of praise. One of these is “Motorsport Manager”. Developed by a smaller developer “Playsport Games”, this sim game has managed a higher review count of 9,943 and a higher categorisation of “Very Positive” [7]. It can be presumed that this game had a lower budget than “F1 Manager 22” as the development company is smaller. Also, they have not acquired any licensing related to any motorsport, choosing fictional race series and drivers. Despite the smaller budget, it can be argued that the graphical fidelity is almost as good as “F1 Manager 22”, with *Figure 2.1.2* below.

Figure 2.1.2 [8]



Having experienced both “F1 Manager 22” and “Motorsport Manager”, personal validity can be given to the categorisation of the reviews. Also, through the experiences of playing the games and my knowledge of software systems, a high appreciation for the complexity of the code and the time it must have taken to develop is gained. Overall gameplay mechanics, such as changing driver abilities and car dynamics were one of the most enjoyable aspects in my opinion. These have attempted to be recreated in the project.

2.2 Background Research

Previous, much smaller scale attempts were made at creating games. However, a game like the scope of this project had not been attempted before. Because of this, research had to be undertaken to understand how it could be achieved.

Firstly, the language used had to be determined. Python was chosen due to the familiarity of the language. Through research, it was also understood that PyGame would be a useful framework to implement in the project. As well as this, the popularity of PyGame meant that there would be a high level of documentation to refer to.

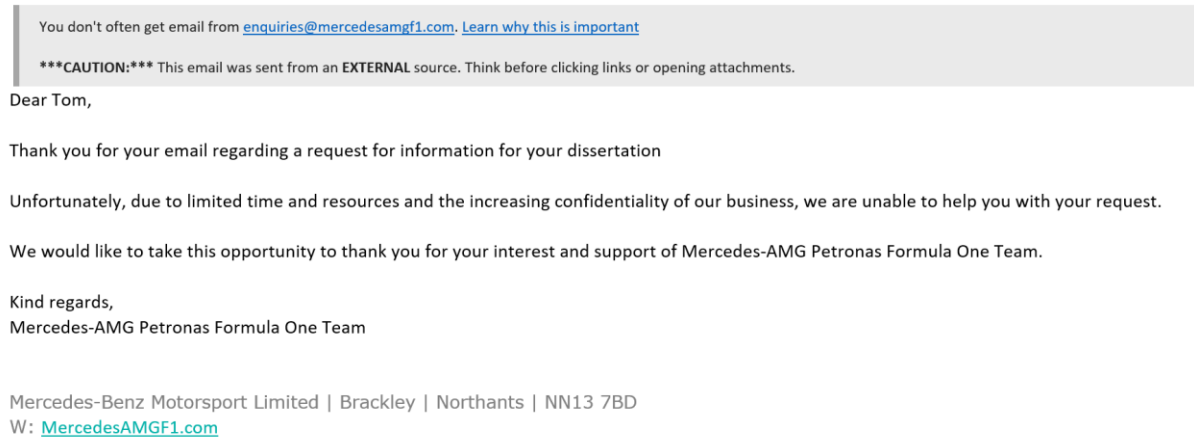
Also, through research, other alternatives were considered. One of these was to use Unreal Engine. This engine and framework would have benefited the project in many ways. One of these would have been the integrated physics simulation and the general ease of use implementing a professional appearing GUI. However, it was also discovered that the skill to use Unreal Engine and its capabilities to a productive level would have been out the time and scope of the project.

Instead, PyGameMenu was used. After careful evaluation, it was determined that leveraging this library would provide a more efficient solution than the development of bespoke GUI from the ground up, despite the limited documentation and relatively modest scale of PyGameMenu.

As well as learning these libraries, attempts were made to gain an insight into both the real-life simulators that F1 teams use, and the development of the sim games mentioned in 2.1. This was achieved through contacting all the major UK F1 teams and the development companies of the

respective games. Unfortunately, only one response was received, seen in *Figure 2.2.1*. This was disappointing as it could have presented an ideal opportunity to learn more about simulations and games that may have dramatically increased the quality of the finished product.

Figure 2.2.1



3 Requirements

It is important that clear requirements of the software should be outlined. This section aims to achieve that. The requirements are presented in three ways. The first is essential requirements. These are requirements that are “must-have” features. Without these requirements being met, the game will not function, even at the most basic level. The second is preferable requirements. Preferable requirements explain features that are not strictly necessary for absolute functionality but instead would greatly increase the software capabilities. Lastly are optional requirements. Requirements that fall under optional are those that would be good to have if an ideal situation could be obtained.

3.1 Essential

- The user must be able to select their driver for a run.
- The user must be able to select their car for a run.
- The default cars must be allocated to the default teams.
- The user must be able to change the parameters of the car.
- The user must be able to change the parameters of the driver.
- The user must be able to select which track they wish to play on.
- The user must be able to see how their time is progressing.
- The user must be able to change their driver.
- The user must be able to cancel the run.
- The user must be able to quit the game.
- The user must not need any additional software.
- Once the driver, car, and track has been selected, the user must be able to start the game.

3.2 Preferable

- The software must provide A.I competitors for a run.
- The user must be able to interact with the game and its selections through a GUI.
- The user must be able to run the game in a simulated and sped up manner in relation to real-time.
- The user must see the results of the run.

3.3 Optional

- The user must be able to store results and the parameters that were used.
- The user must be able to directly control aspects of car performance during a run.

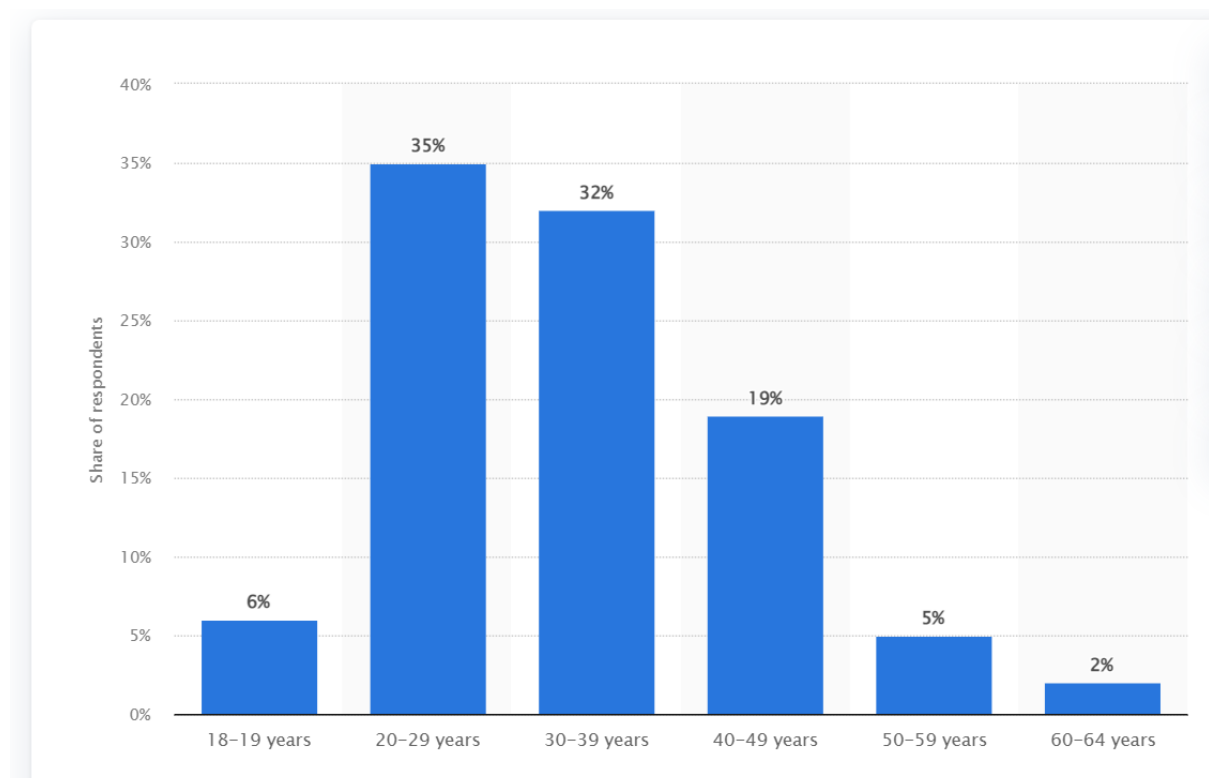
4 Business Development

4.1 Target Audience and Customers

As discussed in 1.1, the primary audience for this project will be “gamers”. To help further define what precisely this means, an investigation needs to occur. As the game is for Windows on PC, it is useful to examine what the largest platforms are where users purchase and manage their games. PCMag, one of the most popular websites and publications for PC users, concludes that Steam is the “best place to buy and rent PC games online” [9] and is the editor’s choice [9]. With this information in mind, it would therefore seem important to ascertain the demographics and audience that Steam entertains.

Firstly, the age demographics will be examined. From an online survey of 815 Steam users in the last quarter of 2022, a graph can be presented, seen in *Figure 4.1.1* [10].

Figure 4.1.1 [10]



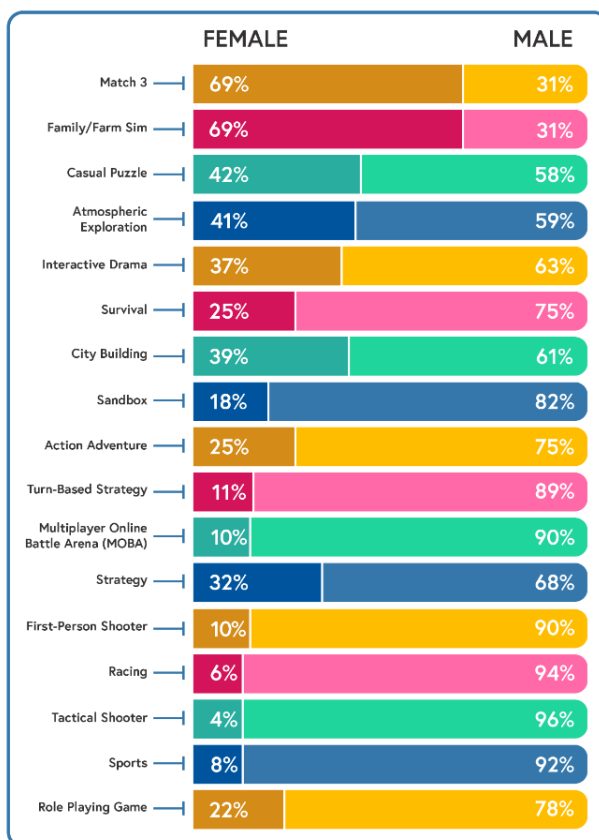
As seen in *Figure 4.1.1*, many users are between 20-29. This is a clear indication that the target audience will be in that age group. Inferences can therefore be made. One of these might be that the 20-29 age group have a reasonable amount of disposable income and therefore the funds to purchase. This is somewhat backed by evidence in the form of a dataset provided by the Office for National Statistics, where the age group 24-43 have the second highest amount of disposable income, and the age group 18-24 is the sixth highest [11].

Following age, investigating the gender for “gamers” may reveal more of the target audience. Earthweb, a research service reports that:

“As of December 2021, most of the gamers on Steam are male at 88.1% compared to 2.4% female and 9.5% unknown. Only 4% of Steam’s traffic comes from women, 46% comes from unknown/unidentified gender, and 50% men.” [12]

To gain more insight into gaming genres, and specifically racing or sports games, a blog from “Gamify” - a game design service was found [13]. In this blog, statistics are gathered and presented. Importantly a bar chart representing gender by genres is presented and is shown in *Figure 4.1.2* [13]:

Figure 4.1.2 [13]



As is apparent from this graph, most gamers who play both racing and sports games are male.

When concluding who the target audience may be for the result of this project, it can safely be assumed that they will be male, between the ages of 20-29.

As far as catering to the target audience, it is believed that the genre of simulation combined with racing and sports that is inherent in the project will suffice in meeting the needs.

It is not believed, however, that the project could reasonably cater to gender and age specifically and it might therefore be more useful to ensure the game is accessible to anyone who may wish to play it.

4.2 Lean Canvas

In lieu of a fully fledged business plan, a lean canvas can be created to present a business model in a succinct manner [14]. Specifically, they are used in start-up companies who need to “stay lean & avoid waste” [14]. It is believed that if this project were to be commercially available, a lean

approach and a start up mentality would almost certainly need to be implemented. This is because investment and capital are unlikely to be available and building a commercial version of this project would need to start from the ground. A lean canvas has been created and can be seen in *Figure 4.2.1*.

Figure 4.2.1

Problem	Solution	Unique Value Propostion	Unfair Advantage	Customer Segments	
Exisitng solutions to gaming are expensive	Ensure the cost of development and the product is low	Allow complete freedom of customisation with little or not restraints on the creativity and experimental nature of the user	There currently is no racing sim that allows full tweaking by the user. There is always constraints.	Males between the age of 20-29	
	Allow the user to change whatever parameters that want			Early Access discount	
There are restraints in current racing sims	Key Metrics		Channels		
	Games purchased		Game store platforms		
	Postive reviews		YouTube/Streaming for advertisement		
Cost Structure			Revenue Streams		
Further market research			Posting on the many digital game selling platforms		
Salary of developer					
Licensing costs					

4.3 Unique Value Proposition (UVP)

The UVP presented by this product is the customisation of any parameters the user wishes. Many sim games do, in some way, have constraints to what can be achieved within the game. There can be satisfaction in this, learning the limitations and performing well in the game. However, it is believed that enabling the user to do what they wish has a higher benefit. This is because they can create their own constraints and therefore their own objectives to what they wish to achieve in the sim game.

The UVP arises from this project as there are no known racing or sports sim game that allows the user this freedom. There are, however, other sub genres within sim games that have similar philosophy.

The first, as mentioned in section 1.3, is “Totally Accurate Battle Simulator”. The game focuses on allowing the user to have as much freedom in setting the parameters of both the in-game units and scenarios. This sparks imagination and creativity from the user to “be the leader of wobblers from ancient lands, spooky places, and fantasy worlds” [3].

The second notable mention is “Universe Sandbox”. In this game, a user can [15]:

“Create & destroy on an unimaginable scale with a realistic physics-based space simulator. Explore the beauty of our universe and the fragility of our planet. Use science to bend the laws of gravity, collide planets, boil away oceans, fire epic space lasers, and customize your universe.”

This approach to sim games is clearly popular and profitable with “overwhelmingly positive” reviews for both these games [3][15].

4.4 The Market

4.4.1 Market Size

Using “annual financial reports of the market-leading companies and industry associations, third-party studies and reports, survey results from our primary research” [16], Statista expects the

revenue from the games industry to reach \$384.90bn in 2023 [16]. The top grossing gaming companies can also expect to gain a revenue of at least \$2.5bn to \$28.2bn in 2023 [17].

With the figures in mind, it is therefore reasonable to assume that the gaming industry can be very profitable for an entrepreneur. Furthermore, despite an estimation as much as 80% of technology focused startups will fail, the percentage is only 50% for gaming startups [18].

However, even with these opportunities outlined, it would be unreasonable to expect this project and its game to get even close to the revenues previously discussed. Despite this, even a moderately successful game might expect to bring in revenue.

4.4.2 Ways to Market

Through experiencing the gaming market first hand as an avid consumer, a trend towards early access games has been observed. Before the rise of digital retailers and widespread use of the Internet, once a game had been released, the state of the development for the game would remain the same. This is due to the inability to patch and update the game with the restrictions of physical media. Game ownership used to entail owning a type of physical media (such as compact disks or tape-based media). With this restriction in mind, a game would have to be fully developed to the best of the ability of the development team prior to release as there would be little to no opportunity to provide the users with updated versions.

Recently, however, the phenomenon of releasing games in early access has become prevalent. An early access game is a product that is released (with both knowledge and consent from the consumer and developer), in the early stages of development. This can be achieved through digital ownership and the ability to provide updates through the Internet.

Early access, therefore, allows many benefits to both parties. The developers gain free testing and bug reporting of their products, and the users gain early play time and enjoyment out of product that may otherwise take years to fully realise. Minecraft is a standout example of this way to market. It was perhaps one of the first early access games to gain massive popularity and the creator subsequently sold the game for \$2.5bn [19].

The best way to market for this project's game, would therefore most likely be to release in early access first. Revenue and play testing would become apparent almost immediately, enabling further development and refinement of the game.

Other ways to market, disregarding early access, would be to launch the game on the many digital retailers on the market. These can include, but are not limited to [9]:

- Steam
- GOG
- Humble Bundle
- Itch.io
- GameFly
- Xbox
- Epic Games store

These retailers take a percentage of the game's revenue but allow the game to gain exposure and accessibility to the market that otherwise may be hard to establish.

Finally, the rise of gaming streamers and YouTubers are an important factor to consider when marketing and advertising a game and therefore a route to market. These streamers and YouTubers

are individuals or collections of people who play games to create video content. Many indie developers and larger companies utilise this method to advertise their games. This can potentially be risky, however, because if a game is not in a state where both the content creator or their viewers enjoy or appreciate the game, a public disavowing may occur.

4.4.2 Competition

As discussed in sections 1.3 and 4.3, there are some notable examples of sim games that would either be directly or indirectly in competition with this project's game.

They, and others, can be categorised as the following:

Direct

- F1 Manager 22
- Motorsport Manager
- Motorsport Manager Mobile

Indirect

- Totally Accurate Battle Simulator
- Universe Sandbox

To more accurately gauge how the competition may fare against this project's game, their strengths and weaknesses can be expressed in *Figure 4.4.2.1*:

Figure 4.4.2.1

	Strengths	Weaknesses
F1 Manager 22	Exceptional graphics, licensing from official sources	Mechanics and gameplay seen to be lacking from reviews, lack of true customisation
Motorsport Manager	High praise for game mechanics and AI, customisation achieved through mods	Lack of licensed content, somewhat lacking in graphical quality
Motorsport Manager Mobile	Ability to be played on mobile platforms such as iOS and Android	Performance bottlenecks-likely through hardware limitations
Totally Accurate Battle Simulator	High level of customisation for both units and world	Not much meta strategic thinking
Universe Sandbox	Excellent representation of the universe/simulated environment for users to explore	Limited in somewhat of the scope of the game, no set objectives to achieve

If the project's game were to be released in the current state of development, it would be unlikely that many would sell many due to how early the development process is. It would take many more hours to achieve a product that could truly compete. However, exploring the competition is still useful as it can guide the development to avoid the weaknesses that the competition expresses whilst also capitalising on their strengths.

4.5 Revenue Model

4.5.1 Monetisation

It is believed that there is only one true method for monetisation for this project. This method would be selling a digital copy of the game to the users. This would be achieved through listing the game on the many digital retailers discussed in section 4.4.2. The revenue and success of this game is directly related to how many could be sold.

A second, much less reliable, method would be to continue development for the purpose of selling the product as one piece to a development company or publisher. This method would be less reliable as the product would need to be at a much higher standard to entice any sort of sale. It would also mean that any intermediate revenue would not likely exist. However, it could be the case that once a more developed prototype is in place, a publisher could potentially purchase the rights to the game once finished.

4.5.2 Revenues and Expenses

Expenses

The primary expense, if this project would be a commercial venture, would be the salary of the developer or developers on the project. Based on salaries obtained through “talent.com”, the average per hour cost for a software engineer is £26.62 [20]. Averaging 40 hours per week, the monthly cost can be estimated at £4,259.2.

A second expense would be the Integrated Development Environment (IDE). For the first year, the monthly cost for using PyCharm would be £19.8 [21]. All other libraries are open source.

A third expense would be the listing of the game. Steam has been used in this example and would cost £80.34 and a commission of 30% on every purchase. Every month, for the first year would be £6.69.

Finally, in order to market the product (as discussed in section 4.4.2) hiring content creators to play and advertise the game would be an additional expense. According to “mysocial”, an individual can expect to pay \$1000 to \$3000 per video for a content creator with 100,000 to 500,000 subscribers [22]. Using the middle of this estimate it would therefore cost £1,205.16. Averaging this cost across the year would be £100.14 per month.

Revenue

The main revenue stream would be selling the game itself. Due to many complex factors, it would be hard to estimate how much it would sell. However, using “game-stats”, a website that collates and presents sales data from Steam, a median revenue for indie games can be established. This site concludes that the median revenue (total) for indie games is \$750 or £602.58 [23]. However, the average is as high as \$210,000 [23]. These statistics do not provide a timeframe for these revenues. To establish what a potential revenue could be, it is more useful to look at what 74% of indie game on Steam earn- \$0-5000 [23]. If the middle of this figure is used, it can be assumed that the project’s game will earn \$2500 over its lifetime. If we assume that the development and sales period is across three years, this will equate to £55.79 per month. The commission from steam would also have to be considered at 30% leading to a final figure of £39.06.

Table

To better understand these figures in a visual manner, *Figure 4.5.2.1* has been created.

Figure 4.5.2.1

Type	Expenses per month (£)	Type	Revenue per month (£)
Salary	4259.2	Sales	39.06
IDE	19.8		
Steam Listing (First Year Only)	6.69		
Marketing	100.14		
Total Monthly (First Year)	4385.83		39.06
Total Monthly (First Year, No Salary)	126.63		39.06
Total Monthly (Other Years)	4379.14		39.06
Total Monthly (Other Years, No Salary)	119.94		39.06

Furthermore, appendix 10.2, shows the projected figures for the first three years, assuming a starting balance of 0. It shows that the game would have lost £156,431. Appendix 10.3 are the same figures; however, it is assuming that a salary will not be taken. In that case, the loss would be £7359.16. Appendices 10.4 and 10.5 show the same figures with an increase of revenue to £100 per month and the loss would be £154,129 with a salary and a loss of £798.12 without.

Judging by these projections, the game may not ever turn a profit. This is even more apparent when including the salary of a single developer. Therefore, additional funds would need to be obtained. It is also important to note that any taxes owed or due have not been considered in these calculations.

Raising funds

When considering raising funds so that the starting balance would not be zero, a loan is one option. This, however, would not increase the worth of the company as all the revenue would effectively be going to paying the loan (and with interest).

A better approach may be to crowdfund the project. With this approach you would “collect money from a large number of people via online platforms” [24]. The general agreement with this approach is that whoever was to back your project, they would receive the game first or gain other benefits such as pre-release versions or behind-the-scenes access. Some popular crowdfunding websites include [25]:

- Kickstarter
- Indiegogo
- Patreon
- Crowdfunder
- GoFundMe
- Fundable
- Crowdcube
- Mightcause
- SeedInvest
- StartEngine

Of course, in order to gain backers in this way, a well-presented prototype and vision needs to be apparent.

5 System Design

5.1 Software Architecture

The main concept for the architectural design of the project is an Event-Driven approach. In this design there are three distinct sections. There are the “event producers, event routers, and event consumers” [26] which are described as asynchronous. This is important in game development as the game can continue to run without any direct input from the user, allowing that game to continue evaluating events and potentially prepare other interactions for the user. Likewise, the user will not have to wait (in most cases) for the game to react to input.

Whilst Event Drive Architecture (EDA) is usually used to describe a large system, like in business processes [27], it can be applied to a software system of any size. The main feature of an EDA is that the software is focused on capturing events and processing them in some manner [28]. This is different from the traditional architecture of responding to requests [28].

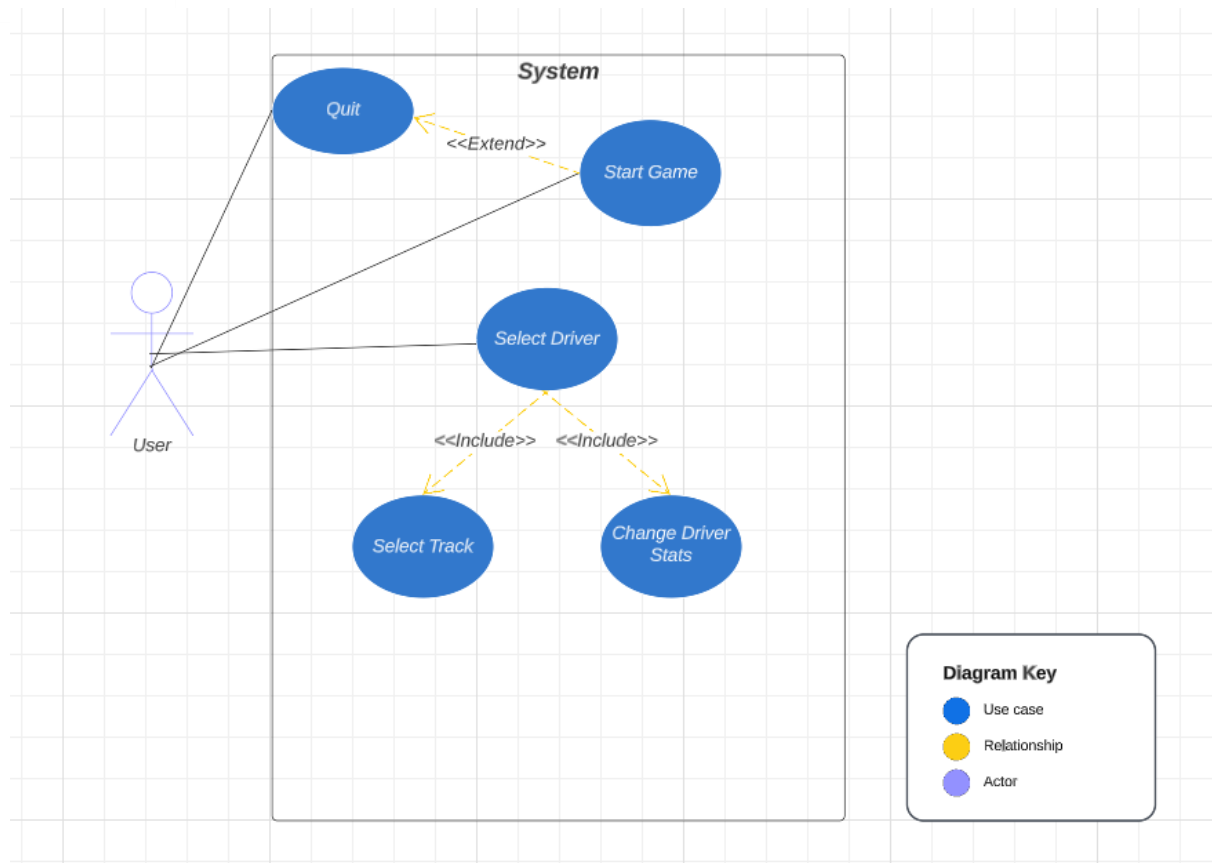
The first step in understanding an EDA is investigating the event producers. These are entities that trigger the event system. From the perspective of this project, the event producers are the inputs from the user. These are mostly keyboard and mouse inputs to the event handler and GUI.

Secondly, the event router or broker is used to handle the triggering of events. They can act in an independent manner, accepting and processing multiple events at once [30]. The event broker in the case of this project is the *pygame.event* module within PyGame and “handles all its event messaging through an event queue” [31]. Events are their own objects and can have different attributes and are mostly user input from peripherals [31]. In more recent versions of PyGame, however, the ability to log events relating to touch controls, voice controls, and screen/window events [31].

Lastly, the event consumer is at the end of the flow. These are parts of the system where the results of the event are realised, and logic can be applied [30]. Normally, these would be systems “such as a billing system, or microservice” [30]. In my project the main event consumer is the PyGameMenu library, which whilst utilizing *pygame.event*, acts on the events- mostly relating to GUI and menu actions.

A use case diagram can be used to visualize the functions of this system and can be seen in *Figure 5.1.1*

Figure 5.1.1



5.2 Technologies

The main technologies implemented in the project are the libraries and frameworks.

PyGame

This module is the most prevalent in the project. Whilst it is technically only a library, it has been used in the project to function as a framework due to the structure and tools provided by the library. I have used the library in such a way that it forms the basic structure and coding practices for the project [33]. This has meant that I have avoided having to build an event handling system myself and PyGame has provided many useful functions to aid in the development, such as ability to receive and understand inputs from system peripherals. PyGame is built upon Simple Directmedia Layer (SDL) library [32]. SDL allows “low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D” [34].

PyGame was chosen because of previous experience with the library. Due to its popularity, there is complete and extensive documentation. This enables effective troubleshooting and removing bugs. PyGame also has a wide community, this means that there are examples that can be investigated to either aid in troubleshooting or present potential ideas.

The main aim of using PyGame is for the event handling capabilities as well as the robust, if not simple, GUI that it can provide.

PyGameMenu

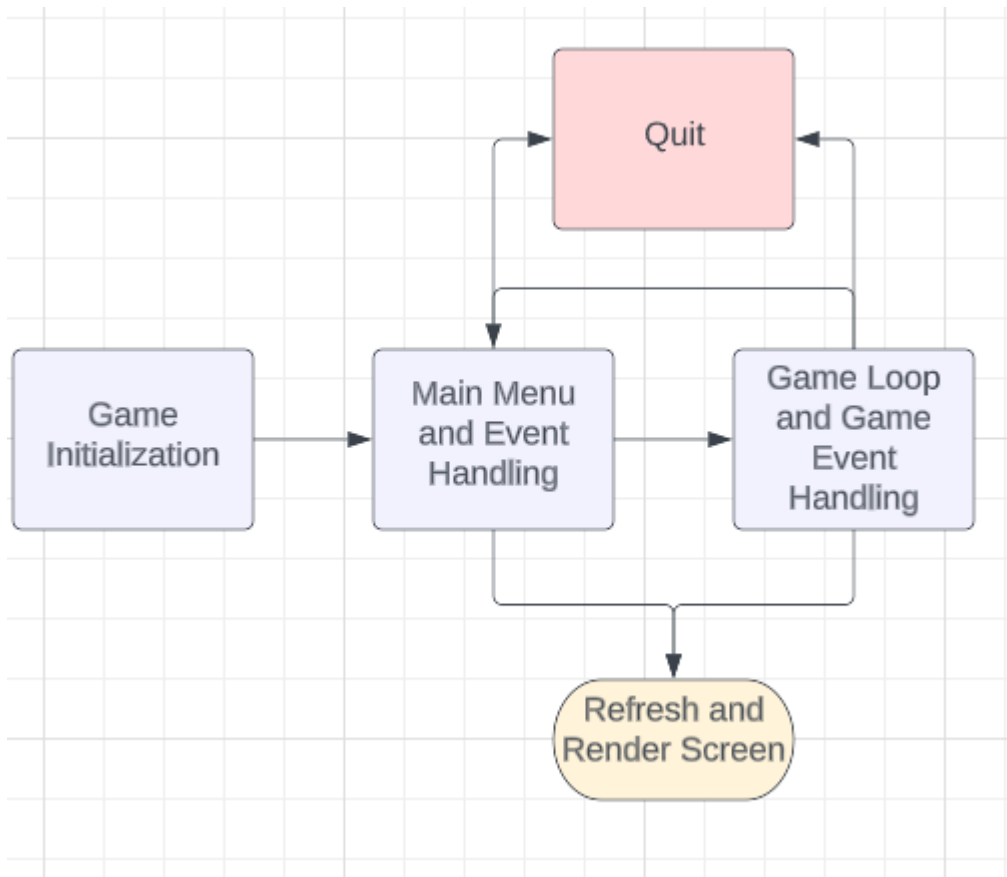
When considering how to implement a GUI to the project, two options were available. One was to use the GUI functions within PyGame. Whilst this may not have been too difficult, a second option was chosen. This was to use PyGameMenu.

This library offers GUI widgets that are compatible with PyGame [35]. The main widgets that were used were the buttons and the menu system itself. PyGameMenu does offer customisation of these widgets for a more bespoke aesthetic, however, for the most part I used default appearances and themes.

5.3 System Overview

Figure 5.3.1 visualises the basic overview of the system and the general flow of processes.

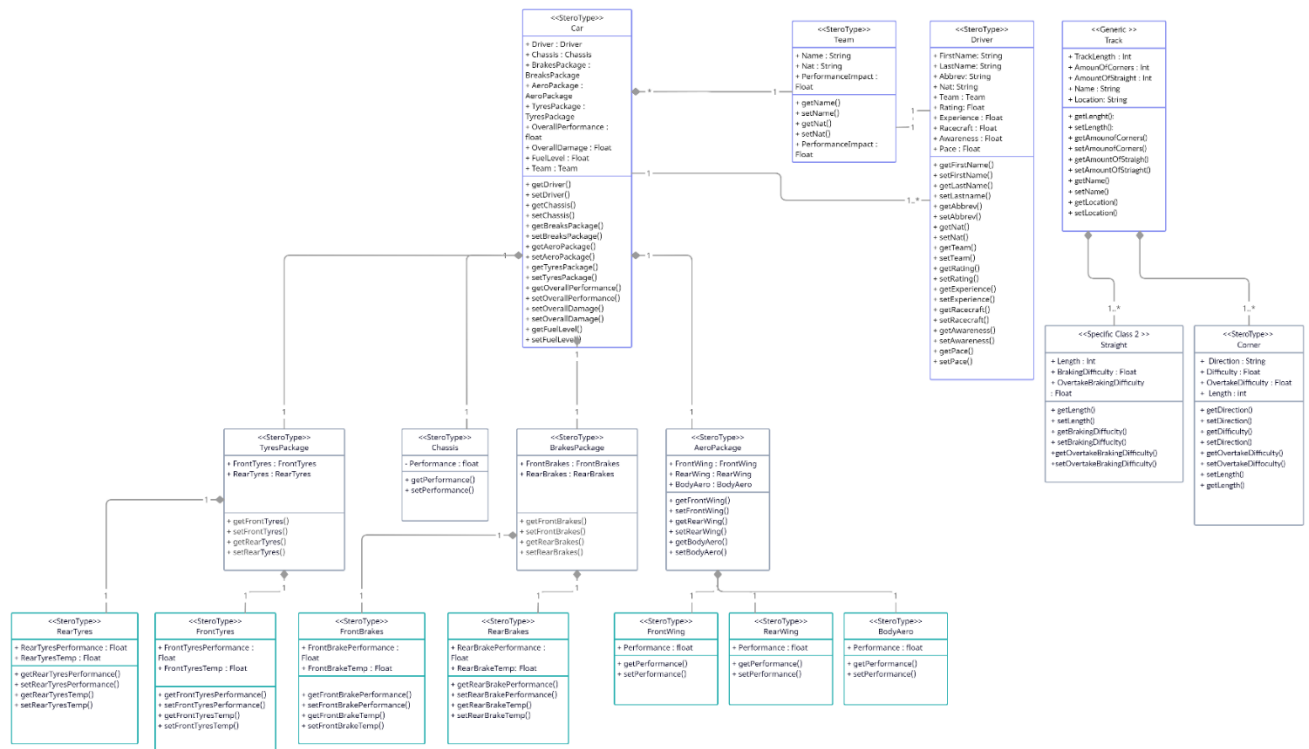
Figure 5.3.1



Whilst there are few abstract sections of the system, each one is somewhat complex and will be discussed further in section 6.

To provide a visualisation of the different entities and their relationship an Entity Relationship Diagram has been provided in Figure 5.3.2

Figure 5.3.2



6 System Implementation

The aim of this section is to go into detail regarding the deeper structure of how the code was written and the reasons behind the methods implemented.

6.1 Modular Programming

Overall, the main design choice was modular programming, meaning to separate the main processes and sections (as seen in *Figure 5.3.1*) into different files. Modular programming is generally used in large projects, however, the advantages in this project can be seen.

Advantages

Instead of having multiple classes, functions, and methods in one file, separating them has enabled the code to be easy to read and understand. If an error occurs in any part of the code, it is easy to navigate and understand where the error is. For example, if there was an error in the Driver class, the program would not get past the setup phase therefore narrowing the scope of the search required.

As well as readability, modular programming enables easy reuse of code and functionality. For example, to change an attribute of the game clock, a function was created in the game clock file that was used in multiple places in the program. Also, if a change was needed to how the game clock functions overall, this could be achieved by changing the code in the game clock file only, avoiding having to find each example of where the function is being used or the function itself.

Scalability is a main advantage of modular programming. Whilst, at the present state, each file is relatively small, forethought for the future of the program was considered. As the design stands, each file can be expanded greatly without necessarily having to change other code segments to

ensure compatibility. For example, if a rework of how the Driver class and associated code is required, only the one file would need to be changed.

Disadvantages

As this project is small, the benefits of modular programming cannot be fully realised. Also, it does increase the code size and complexity. For example, I had to ensure each file and module was correctly imported into the corresponding relevant file.

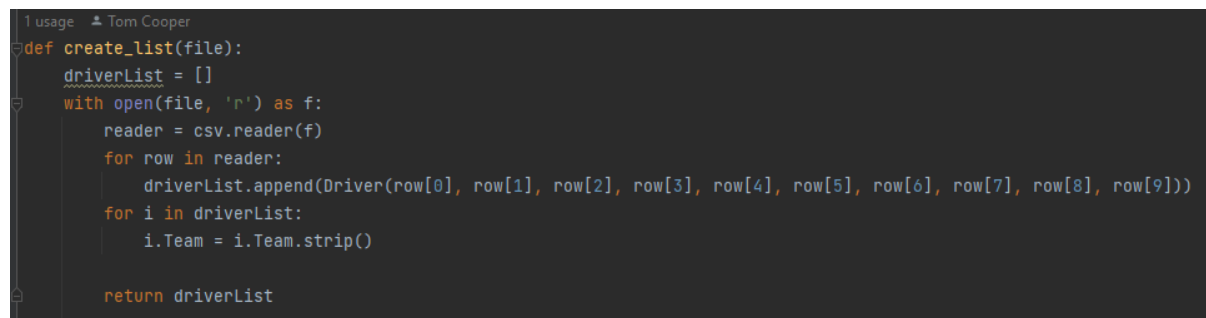
6.2 Game Initialization and Setup

The first step in creating the initialization was to define the classes to be used. This was achieved through the files:

- Car_init_.py
- Driver_init_.py
- Track_init_.py
- Team_init_.py

In these files, the classes and their attributes are defined, as well as any associated functions and methods relating to that class. For example, in the Driver file, *Figure 6.2.1* shows a function in that file that is used in the setup.

Figure 6.2.1



```

1 usage  Tom Cooper
def create_list(file):
    driverList = []
    with open(file, 'r') as f:
        reader = csv.reader(f)
        for row in reader:
            driverList.append(Driver(row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7], row[8], row[9]))
        for i in driverList:
            i.Team = i.Team.strip()
    return driverList

```

Lists were used to store the track, driver, car, and team objects. In the setup process in setup.py CSV files are used to define the default values for these objects. *Figure 6.2.1* shows how the CSV files are used to set the values. The reader takes each row from the CSV file and creates a new driver to be appended to the driver list based on the values from each row in the CSV file. An example CSV file can be seen in appendix 10.6

Using this approach for initialising the objects is useful as any changes to desired default values can be changed by the user, without too much knowledge required. They need only to change the values in the CSV file which is separate from the source code. Also, it allows different settings to be applied to the game. For example, if the game was to use a different racing series, a separate CSV for that series could be created and used in the initialization instead. This could allow updates for the user to be provided with relative ease as the code itself is not changing, only what is inputted into the reader.

The downside to this approach, however, is that the CSV need to be formatted in the correct way. There is currently no error handling for a CSV file that, for example, has too many rows.

Setup.py is the main file used to start the game. It imports all the relevant modules and with this, Python executes the code of those modules automatically. The lists of the objects are then initialized using the CSV files located in Resources.

Next, a function is used to assign each driver to a team object. They will only be assigned if the team's name for the driver and the team's name for the team match. In future versions, error handling will have to be implemented here if there is no match.

Drivers are then assigned to a car object whilst simultaneously the car for each driver to be assigned is created.

Finally, the track is created using a provided CSV file and messages showing progress and completion are printed.

6.3 Main

There are three major elements in the main.py file. These encapsulate the main menu, the game loop, and any rendering and screen display.

6.3.1 Main Menu

First, the PyGame module is initialized with all the variable declarations to follow. The main menu loop is then started. In this loop PyGameMenu creates the buttons for the menu choices and deals with an event if any option is chosen. PyGameMenu has its own loop designed for menus and this can be seen in *Figure 6.3.3.1*. Contained in this loop is built-in event handling. Although event handling for quitting the game is coded separately using PyGame functionality.

Figure 6.3.3.1

```
if main_menu.is_enabled():
    main_menu.mainloop(screen, main_background)

pygame.display.flip()
```

This loop only occurs when the main menu has been enabled. It takes in the screen parameters and the background parameters and then the display is updated and rendered.

In the background function, the colour can be changed. It was decided that as the background is constantly being updated in this loop, it would be an appropriate place to check if a driver has been selected or not. With this check, background text would be displayed to show the user who they have selected, or if they have selected anyone at all.

Also contained in the main menu loop is the logic for populating the driver, team, and track list for the sub menus. Once the selections by the user for the game have been registered and the start game button has been pressed, the game loop starts.

6.3.3 Game Loop

First, a try/except condition is set to test if a car/driver object has been selected. If not, the game loop will return to the main menu with a message displayed to the user. Next, if a driver has been selected, the main menu is disabled and a count to start the game is shown as well as the car/driver object being set to 0 on the track progression. The driving code for the progression of the loop is the position of the car on the track. The rate at which the car/driver object makes progress in its position

is based on the attribute values of the car, track, team, and driver as well as which segment the car/object is on. For example, in a corner, progression will be slower.

The value representing progression equates to 1m in real life. This is accurate because detailed satellite imagery was used to produce the CSV in the track creation. In this process Google Earth was used and manual waypoints were selected and measured to gain the geometry of the track. This was a manual process, after some automatic processes were attempted and failed. In the current state, only one track has been mapped in this way. However, it does prove the concept works and therefore only time is the limiting factor in adding additional tracks.

The progression is evaluated in every loop. It is compared against the total length of the run. This is determined by the track length multiplied by how many laps have been selected. Once the total length has been exhausted the game loop finishes. The other way for the loop to terminate is by running out of fuel. This is determined by the fuel attribute in the car class and triggers a termination, or game over, when fuel is at 0.

The user is returned to the main menu once the game loop has finished. Also, an option (and a message to explain) to quit can be actioned by the user by pressing “ESC”. This is handled by the PyGame event handler.

6.3.4 Screen Rendering and Game Clock

The main method for determining time in this project is by using the PyGame clock. This functionality allows time keeping and frame rate stability [36]. The clock is updated at the start of each loop; therefore a “tick” can be described as each update of the clock. The clock is used to display the time elapsed for the user. As well as this, it has been set to limit the frames per second to 30. This is to ensure that any display changes are smooth.

The screen is declared and initialized at the beginning of this file. However, `pygame.display.flip()`, updates and renders the screen at the end of each loop (both menu and game). This is to ensure any changes or events that have happened in any loop is represented in the display, if required. For example, the display timer is updated in every single tick.

7 System Testing

No automated methods for testing have been implemented in this project. Instead, a test list was created, and each test was conducted manually. The test list was based off the requirements and contains many types of testing. It is important to note, however, that exploratory testing [36] occurred on many occasions during the development process. With this testing, once a feature had been coded, a test would occur immediately. As well as this, testing the overall program was undertaken after every milestone. This is because certain modules or features were disabled during the development process and therefore a test needed to occur to ensure functionality when the disabled features were re-enabled.

7.1 Testing Table

There are three types of tests that occur in this table. They are:

- Unit
- Integrated
- Functional

Unit testing is a low-level type. It serves to test specific stand-alone methods and functions. These would have been the best to automate [38]. Integrated testing serves to test how well or functional

different modules interact with each other [38]. When testing for the business type requirements, functional testing has been applied [38].

Test #	Test Purpose /Requirement	Expected Result	Test Type	Test Result	Comments
1	The user must be able to select their driver for a run.	In the menu system, a list of drivers is displayed and once the user has selected one, that object will be stored in memory	Unit	Pass	
2	The user must be able to select their car for a run.	In the menu system, once a driver has been selected, the user can select which car to use.	Unit	Pass	
3	The default cars must be allocated to the default teams.	In the setup processes, all cars are associated with correct team.	Integrated	Pass	
4	The user must be able to change the parameters of the car.	In the menu system, options to change to parameters of the car should be displayed and any changes to the object must be applied	Unit	Pass	
5	The user must be able to change the parameters of the driver.	In the menu system, options to change to parameters of the driver should be displayed and any changes to the object must be applied	Unit	Pass	
6	The user must be able to see how their time is progressing.	In the game, there is a timer displaying elapsed time	Functional	Pass	
7	The user must be able to change their driver.	In any part of the program, the user should be able to select a difference driver	Unit	Fail	This test is specific to any part of the program. The user can in fact change their driver but only in the selection process in the menu.
8	The user must be able to cancel the run.	In the game, the user should be able to quit back to the main menu	Unit	Pass	
9	The user must be able to quit the game.	In any part of the program, the user should be able to quit the program	Unit	Pass	

10	The user must not need any additional software.	The program and game will run without any other requirements or downloads.	Functional	Pass	
11	Once the driver, car, and track has been selected, the user must be able to start the game.	Once all selections have been stored in memory, the user will be able to start the game	Integrated	Pass	
12	The software must provide A.I competitors for a run.	AI competitors will populate the game	Integrated	Fail	No attempt was made to implement AI drivers
13	The user must be able to interact with the game and its selections through a GUI.	The GUI provides an easy to use and intuitive system for navigating and playing the game	Functional	Pass	
14	The user must be able to run the game in a simulated and sped up manner in relation to real-time.	The game can be run in simulated time, rather than real time	Integrated	Fail	Attempts were made to change the time resolution- with no success
15	The user must see the results of the run.	At the end of the game, statistics of the run are shown	Functional	Pass	
16	The user must be able to store results and the parameters that were used.	Save game files are present with the settings that were used for the saved runs	Integrated	Fail	No attempt was made to implement a save game file system
17	The user must be able to directly control aspects of car performance during a run.	The user can change aspects such as if the driver is pushing or not	Unit	Fail	The only aspect a user can change is pushing or not, whilst this may satisfy this test condition, it is believed the result

				should express more than just being to "push" or not.
--	--	--	--	---

8 Critical Appraisal

Critical appraisal will focus on how well the final product of the project has met the requirements, aims, and objectives set out in section 1.

8.1 Objectives

Objective 1 – Implementing a GUI.

Whilst only a very basic GUI has been implemented, I believe this objective has been satisfied. The GUI implemented is easy to use and requires little explanation as to what interactions the user can do. The GUI makes use of inputs through both a mouse and keyboard and therefore gives options for the user leading to a better experience.

However, there can be improvements made to the GUI. These relate to the overall “look and feel” of the GUI. These improvements are:

- Professional appearance of the GUI.
- Choosing colours that better suit any users that may have disabilities.

Objective 2 - Packaging the game.

This objective has been achieved. PyInstaller was used to achieve this objective. This library “bundles a Python application and all its dependencies into a single package” [39]. However, I was unsuccessful in getting the one file function working. Instead, a folder is created with all the relevant dependencies and an .exe for the user. This still fulfils the objective as the user strictly does not have to download and make use of any other software.

The only negative point that is associated with this objective is that exceptions need to be made in Windows Defender to allow the .exe to run. However, pop-ups from the Windows OS normally function to give the user a choice as to allow the .exe or not.

Objective 3 – Allow customisation.

With the possibility to change the CSV file values, and the also to change attribute values within the game, I believe this objective has been fully satisfied. Improvements need to be made in ensuring any changes that users make to values, especially concerning the CSV files, do not prevent the game from functioning.

Objective 4 – Realism

There has only been a superficial satisfaction of this objective achieved through using real names, locations and teams etc. However, no physics realism has been implemented nor is there any implementation to try and achieve a “feel” of realism. Improvements to meet this objective could be achieved through using a purpose-built game engine such as Unreal Engine.

Objective 5 – Licensing

Whilst real names and teams have been used, this is only applicable to this project due to the private study exemption [40]. If commercialising were to occur, the objective would fail as there would be no available funds to purchase licenses. As discussed in section 4.5.2, the project would fail to generate a profit, even without factoring in licensing.

8.2 Social, Commercial and Economic Context

Social

As this project is producing a game, and therefore an entertainment product, social considerations may not be a high concern. However, ensuring the game is accessible by as many demographics as possible is important. This is because inclusion needs to be offered to all, especially with those who may have disabilities that limit their ability to enjoy games.

The game is benign concerning any social issues and would be appropriate for any age range or culture.

Commercial

This project would not be “game-breaking” in the industry. Even when fully developed, there are similar products on the market. There is the potential for this game to generate a profit, but it would take significant time to achieve, and a much-refined version of the game would need to be produced.

Legal issues would arise if released commercially due to the licensing concerns.

Economic

The only contextual application the economy has to the project would be if it was viable to release as it is. As discussed in section 4.5.2, it would not be. However, development could continue without any expectation on return or pay. This path would need to be supported by other means income. For example, another job.

8.3 Future Work

All but one of the essential requirements have been met. The failed requirement, however, could arguably been seen as being met (as discussed in the comment section of 7.1 test 7). Only half of the preferable requirements have been met and only one of the optional requirements met.

Whilst some requirements have not been met, the issue relating to this may be due to an ambitious estimation of skill and scope. In future software projects, it will be important to temper expectations and reach a relatively basic level of achievement first, before moving on to harder challenges.

Developing the project in the future would require additional tests by outside sources and “play-testers”. Currently, the only tests have been concluded by me.

From analysing section 8.1, it is clear there are two stand out points for improvement. One would be making the GUI look and feel more professional. A dedicated UI designer would be essential for this as my personal skills are somewhat lacking.

Secondly, the realism is not high in this project. This is even more apparent when compared to the competition. This, I believe, has a big impact on how the game would be currently viewed.

8.4 Personal Development

Through this project, my experience using Python has grown. With previous experience using Python, the learning curve was not too high. However, using PyGame proved to be a harder

challenge. Through that challenge, though, I have gained more experience and knowledge regarding game development.

The skill of managing a game regarding clock speed, event handling, and rendering has been enjoyable as these skills have not directly appeared in this form during my studies.

Due to external factors, however, time was a limiting factor. I believe a higher quality product could quite easily be achieved through starting the development earlier.

Because of the external factors, organisational skills have increased as I have had to manage achieving what could be possible in a smaller time frame. However, more improvements could be made to document the progression of the project as it happened. This would lead to a better analysis of effort and where further improvements could be made.

9 Conclusion

A prototype level game has been produced as a result of this project. A higher level of professional appearance regarding the game could easily be achieved as the underlying code is robust. Most aims and objectives have been met in the progression of this project. However, more time would be required to fully realise the initial imagined scope of the project.

The biggest technical feature faced would most likely be how to manage the time resolution of Python and implementing a robust main and game loop where the display is rendering in the intended fashion as well as the game correctly handling events. Much of the development time was focused on achieving this robustness with many trial and error attempts having to be made. However, a solid foundation has been produced.

Overall, I am for the most part satisfied with the outcome of the project but believe there is room for improvement.

References

- [1] P. B, S. K, and V. K, "Gaming simulation market statistics: Industry analysis - 2030," Allied Market Research, <https://www.alliedmarketresearch.com/gaming-simulators-market-A06821> (accessed May 15, 2023).
- [2] V. Yanev, "Video game demographics - who plays games in 2023?," Techjury, <https://techjury.net/blog/video-game-demographics/#gref> (accessed May 15, 2023).
- [3] "Totally accurate battle simulator on steam," Totally Accurate Battle Simulator on Steam, https://store.steampowered.com/app/508440/Totally_Accurate_Battle_Simulator/ (accessed May 15, 2023).
- [4] J. Tugayeva and Jamie Tugayeva Jamie comes from a long line of military and civilian pilots! His brother, "Best flight simulator software guide," Aviator Insider, <https://aviatorinsider.com/avionics/best-flight-simulator-software/> (accessed May 15, 2023).
- [5] Screenshot of F1 Manager 22.
- [6] "Save 80% on F1® manager 2022 on Steam," Save 80% on F1® Manager 2022 on Steam, https://store.steampowered.com/app/1708520/F1_Manager_2022/ (accessed May 16, 2023).
- [7] "Motorsport manager on steam," Motorsport Manager on Steam, https://store.steampowered.com/app/415200/Motorsport_Manager/ (accessed May 16, 2023).
- [8] Screenshot of Motorsport Manager.
- [9] I. 2013, "The best places to buy and rent PC games online in 2023," PCMag UK, <https://uk.pcmag.com/games/82383/the-best-places-to-buy-and-rent-pc-games-online> (accessed May 16, 2023).
- [10] A. Kunst, "Steam users by age in the U.S. 2022," Statista, <https://www.statista.com/forecasts/1242344/steam-us-user-share-by-age#:~:text=The%20statistic%20about%20the%20distribution,20%20to%2029%20years%20old.> (accessed May 19, 2023).
- [11] Office for National Statistics and E. Paula, Office for National Statistics, 2023.
- [12] "How many people use steam in 2023? (monthly active users)," EarthWeb, <https://earthweb.com/how-many-people-use-steam/> (accessed May 19, 2023).
- [13] M. Denton, "Game Genre & Statistics: Not all games are created equal (part 1)," Make a Video Game in Minutes Not Months, <https://www.gamify.com/gamification-blog/not-all-games-are-created-equal-pt1> (accessed May 20, 2023).
- [14] S. Mullen, "An introduction to lean canvas," Medium, https://medium.com/@steve_mullen/an-introduction-to-lean-canvas-5c17c469d3e0 (accessed May 20, 2023).
- [15] "Universe Sandbox on steam," Universe Sandbox on Steam, https://store.steampowered.com/app/230290/Universe_Sandbox/ (accessed May 20, 2023).
- [16] "Video games - worldwide: Statista market forecast," Statista, <https://www.statista.com/outlook/dmo/digital-media/video-games/worldwide#methodology> (accessed May 20, 2023).

- [17] "Top 10 biggest video game companies in the world," All Top Everything, <https://www.alltopeverything.com/top-10-biggest-video-game-companies/> (accessed May 20, 2023).
- [18] Luisa Zhou, "%%title%%," Luisa Zhou, <https://www.luisazhou.com/blog/startup-failure-statistics/#:~:text=Within%20the%20Gaming%20industry%2C%20startups,around%20a%2050%25%20failure%20rate.> (accessed May 20, 2023).
- [19] C. Matyszczyk, "Billionaire who sold Minecraft to Microsoft is sad and lonely," CNET, <https://www.cnet.com/culture/billionaire-who-sold-minecraft-to-microsoft-is-sad-and-lonely/#:~:text=The%2036%2Dyear%2Dold%20founded,outbidding%20Jay%20Z%20and%20Beyonce> . (accessed May 20, 2023).
- [20] "Software+Engineer salary in United Kingdom - average salary," Talent.com, <https://uk.talent.com/salary?job=software%2Bengineer> (accessed May 20, 2023).
- [21] "Buy pycharm professional: Pricing and licensing, Discounts - JetBrains Toolbox Subscription," JetBrains, <https://www.jetbrains.com/pycharm/buy/#commercial> (accessed May 20, 2023).
- [22] Mysocial, "How to determine the price for a YouTube influencer," Mysocial, <https://www.mysocial.io/blog/how-to-determine-the-price-for-a-youtube-influencer> (accessed May 20, 2023).
- [23] "Steam tags stats," Games, <https://games-stats.com/steam/tags/> (accessed May 20, 2023).
- [24] "Crowdfunding explained," Internal Market, Industry, Entrepreneurship and SMEs, https://single-market-economy.ec.europa.eu/access-finance/guide-crowdfunding/what-crowdfunding/crowdfunding-explained_en#:~:text=Crowdfunding%20is%20a%20way%20of,way%20of%20accessing%20alternative%20funds. (accessed May 20, 2023).
- [25] "10 top crowdfunding sites: Fund your vision in 2023," Shopify, <https://www.shopify.com/uk/blog/crowdfunding-sites> (accessed May 20, 2023).
- [26] H. Taylor, A. Yochem, L. Phillips, and F. Martinez, "Event-driven architecture: How SOA enables the real-time enterprise =Mian Xiang Soa de Shi Jian Qu Dong Jia Gou she Ji Yu Shi Xian," Amazon, <https://aws.amazon.com/event-driven-architecture/> (accessed May 21, 2023).
- [27] "What is event-driven architecture?," TIBCO Software, <https://www.tibco.com/reference-center/what-is-event-driven-architecture> (accessed May 21, 2023).
- [28] [1] "What is event-driven architecture?," Red Hat - We make open source technologies for the enterprise, <https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture> (accessed May 21, 2023).
- [30] Person, "Event-driven architecture: What it is and how to use it effectively: Contino: Global transformation consultancy," Contino, <https://www.contino.io/insights/event-driven-architecture#:~:text=In%20an%20EDA%2Dbased%20system,the%20other%20consumers%20will%20do.> (accessed May 21, 2023).
- [31] Pygame.event - pygame v2.4.0 documentation, <https://www.pygame.org/docs/ref/event.html> (accessed May 21, 2023).
- [32] "About - wiki," About - pygame wiki, <https://www.pygame.org/wiki/about> (accessed May 21, 2023).

[33] R. Ranjan, "What is a framework in Programming & Why You Should Use one," Insights - Web and Mobile Development Services and Solutions, <https://www.netsolutions.com/insights/what-is-a-framework-in-programming/> (accessed May 21, 2023).

[34] "About SDL," Simple DirectMedia Layer - Homepage, <https://www.libsdl.org/> (accessed May 21, 2023).

[35] "Pygame-menu," PyPI, <https://pypi.org/project/pygame-menu/#:~:text=Introduction,with%20multiple%20options%20to%20customize.> (accessed May 21, 2023).

[36] Pygame.time - pygame v2.4.0 documentation, <https://www.pygame.org/docs/ref/time.html> (accessed May 21, 2023).

[37] Atlassian, "Exploratory testing," Atlassian, <https://www.atlassian.com/continuous-delivery/software-testing/exploratory-testing#:~:text=Exploratory%20testing%20is%20an%20approach,the%20scope%20of%20other%20tests.> (accessed May 21, 2023).

[38] Atlassian, "The different types of testing in software," Atlassian, <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing> (accessed May 21, 2023).

[39] "Pyinstaller manual" PyInstaller Manual - PyInstaller 5.11.0 documentation, <https://pyinstaller.org/en/stable/> (accessed May 21, 2023).

[40] "Exceptions to copyright," GOV.UK, <https://www.gov.uk/guidance/exceptions-to-copyright> (accessed May 21, 2023).

10 Appendix

10.1

Task	Date to Complete	Achieved	On time
Create Team Class	08/02/2023	Yes	Yes
Create Driver Class	08/02/2023	Yes	Yes
Create Car Class	08/02/2023	Yes	Yes
Add driver from csv and convert to dataframe object	08/02/2023	Yes	Yes
Add teams from csv and convert to dataframe object	08/02/2023	Yes	Yes
Create Track Class	15/02/2023	Yes	Yes
Add tracks from csv and convert to dataframe object	15/02/2023	Yes	Yes
Create set up script to assign default cars to drivers and teams	15/02/2023	Yes	Yes
Create logic algorithms for running a car on a track	20/02/2023	Yes	No
Have one drive/car/team combination run 1 lap on a track	20/02/2023	Yes	No
Have the lap be generally representative of a real lap time	22/02/2023	Yes	No
Add choice for car/team/track rather than hardcoded	01/03/2023	Partly	No
Add options for user to change parameters of car/team/driver	01/03/2023	Pending	Pending
Add CLI feedback on progress of simulation	08/03/2023	Yes	Yes
Add a second car to simulation with logic for interaction	15/03/2023	Pending	Pending
Add Starting Grid Class	22/03/2023	No	No
Have all 20 cars running on the simulation with logic for interaction	22/03/2023	No	No
Allow the user to "play" as a car/team/driver and make decisions themselves	01/04/2023	No	No

10.2

	Expense	Revenue	Loss/Gain	Balance
				-
Jan	4385.83	36.06	-4349.77	4349.77
				-
Feb	4385.83	36.06	-4349.77	8699.54
				-
Mar	4385.83	36.06	-4349.77	13049.3
				-
Apr	4385.83	36.06	-4349.77	17399.1
				-
May	4385.83	36.06	-4349.77	21748.9
				-
Jun	4385.83	36.06	-4349.77	26098.6
				-
Jul	4385.83	36.06	-4349.77	30448.4
				-
Aug	4385.83	36.06	-4349.77	34798.2
				-
Sep	4385.83	36.06	-4349.77	39147.9
				-
Oct	4385.83	36.06	-4349.77	43497.7

Year 2	Nov	4385.83	36.06	-4349.77	47847.5
	Dec	4385.83	36.06	-4349.77	52197.2
	Jan	4379.14	36.06	-4343.08	56540.3
	Feb	4379.14	36.06	-4343.08	60883.4
	Mar	4379.14	36.06	-4343.08	65226.5
	Apr	4379.14	36.06	-4343.08	69569.6
	May	4379.14	36.06	-4343.08	73912.6
	Jun	4379.14	36.06	-4343.08	78255.7
	Jul	4379.14	36.06	-4343.08	82598.8
	Aug	4379.14	36.06	-4343.08	86941.9
	Sep	4379.14	36.06	-4343.08	-91285
	Oct	4379.14	36.06	-4343.08	-95628
Year 3	Nov	4379.14	36.06	-4343.08	99971.1
	Dec	4379.14	36.06	-4343.08	-104314
	Jan	4379.14	36.06	-4343.08	-108657
	Feb	4379.14	36.06	-4343.08	-113000
	Mar	4379.14	36.06	-4343.08	-117343
	Apr	4379.14	36.06	-4343.08	-121687
	May	4379.14	36.06	-4343.08	-126030
	Jun	4379.14	36.06	-4343.08	-130373
	Jul	4379.14	36.06	-4343.08	-134716
	Aug	4379.14	36.06	-4343.08	-139059
	Sep	4379.14	36.06	-4343.08	-143402
	Oct	4379.14	36.06	-4343.08	-147745
	Nov	4379.14	36.06	-4343.08	-152088
	Dec	4379.14	36.06	-4343.08	-156431

10.3

Year 1		Expense	Revenue	Loss/Gain	Balance
	Jan	126.63	36.06	-90.57	4349.77
	Feb	126.63	36.06	-90.57	4440.34
	Mar	126.63	36.06	-90.57	4530.91
	Apr	126.63	36.06	-90.57	4621.48
	May	126.63	36.06	-90.57	4712.05

					-
	Jun	126.63	36.06	-90.57	4802.62
					-
	Jul	126.63	36.06	-90.57	4893.19
					-
	Aug	126.63	36.06	-90.57	4983.76
					-
	Sep	126.63	36.06	-90.57	5074.33
	Oct	126.63	36.06	-90.57	-5164.9
					-
	Nov	126.63	36.06	-90.57	5255.47
					-
	Dec	126.63	36.06	-90.57	5346.04
					-
	Jan	119.94	36.06	-83.88	5429.92
	Feb	119.94	36.06	-83.88	-5513.8
					-
	Mar	119.94	36.06	-83.88	5597.68
					-
	Apr	119.94	36.06	-83.88	5681.56
					-
	May	119.94	36.06	-83.88	5765.44
					-
Year 2	Jun	119.94	36.06	-83.88	5849.32
	Jul	119.94	36.06	-83.88	-5933.2
					-
	Aug	119.94	36.06	-83.88	6017.08
					-
	Sep	119.94	36.06	-83.88	6100.96
					-
	Oct	119.94	36.06	-83.88	6184.84
					-
	Nov	119.94	36.06	-83.88	6268.72
	Dec	119.94	36.06	-83.88	-6352.6
					-
	Jan	119.94	36.06	-83.88	6436.48
					-
	Feb	119.94	36.06	-83.88	6520.36
					-
	Mar	119.94	36.06	-83.88	6604.24
					-
Year 3	Apr	119.94	36.06	-83.88	6688.12
	May	119.94	36.06	-83.88	-6772
					-
	Jun	119.94	36.06	-83.88	6855.88
					-
	Jul	119.94	36.06	-83.88	6939.76
					-
	Aug	119.94	36.06	-83.88	7023.64
					-
	Sep	119.94	36.06	-83.88	7107.52

10.4

Year 1	Oct	119.94	36.06	-83.88	-7191.4
	Nov	119.94	36.06	-83.88	7275.28
	Dec	119.94	36.06	-83.88	7359.16
					-
					-
					-
					-
					-
					-
					-
					-
					-
Year 2	Jan	126.63	100	-26.63	-26.63
	Feb	126.63	100	-26.63	-53.26
	Mar	126.63	100	-26.63	-79.89
	Apr	126.63	100	-26.63	-106.52
	May	126.63	100	-26.63	-133.15
	Jun	126.63	100	-26.63	-159.78
	Jul	126.63	100	-26.63	-186.41
	Aug	126.63	100	-26.63	-213.04
	Sep	126.63	100	-26.63	-239.67
	Oct	126.63	100	-26.63	-266.3
	Nov	126.63	100	-26.63	-292.93
	Dec	126.63	100	-26.63	-319.56
Year 3	Jan	119.94	100	-19.94	-339.5
	Feb	119.94	100	-19.94	-359.44
	Mar	119.94	100	-19.94	-379.38
	Apr	119.94	100	-19.94	-399.32
	May	119.94	100	-19.94	-419.26
	Jun	119.94	100	-19.94	-439.2
	Jul	119.94	100	-19.94	-459.14
	Aug	119.94	100	-19.94	-479.08
	Sep	119.94	100	-19.94	-499.02
	Oct	119.94	100	-19.94	-518.96
	Nov	119.94	100	-19.94	-538.9
	Dec	119.94	100	-19.94	-558.84
Year 4	Jan	119.94	100	-19.94	-578.78
	Feb	119.94	100	-19.94	-598.72
	Mar	119.94	100	-19.94	-618.66
	Apr	119.94	100	-19.94	-638.6
	May	119.94	100	-19.94	-658.54
	Jun	119.94	100	-19.94	-678.48
	Jul	119.94	100	-19.94	-698.42
	Aug	119.94	100	-19.94	-718.36
	Sep	119.94	100	-19.94	-738.3
	Oct	119.94	100	-19.94	-758.24
	Nov	119.94	100	-19.94	-778.18
	Dec	119.94	100	-19.94	-798.12

10.5

Expense Revenue Loss/Gain Balance

Year 1				-	
	Jan	4385.83	100	-4285.83	4285.83
					-
	Feb	4385.83	100	-4285.83	8571.66
					-
	Mar	4385.83	100	-4285.83	12857.5
					-
	Apr	4385.83	100	-4285.83	17143.3
					-
	May	4385.83	100	-4285.83	21429.2
	Jun	4385.83	100	-4285.83	-25715
					-
Year 2	Jul	4385.83	100	-4285.83	30000.8
					-
	Aug	4385.83	100	-4285.83	34286.6
					-
	Sep	4385.83	100	-4285.83	38572.5
					-
	Oct	4385.83	100	-4285.83	42858.3
					-
	Nov	4385.83	100	-4285.83	47144.1
	Dec	4385.83	100	-4285.83	-51430
					-
	Jan	4379.14	100	-4279.14	55709.1
Year 3					-
	Feb	4379.14	100	-4279.14	59988.2
					-
	Mar	4379.14	100	-4279.14	64267.4
					-
	Apr	4379.14	100	-4279.14	68546.5
					-
	May	4379.14	100	-4279.14	72825.7
					-
	Jun	4379.14	100	-4279.14	77104.8
					-
	Jul	4379.14	100	-4279.14	81383.9
Year 3					-
	Aug	4379.14	100	-4279.14	85663.1
					-
	Sep	4379.14	100	-4279.14	89942.2
					-
	Oct	4379.14	100	-4279.14	94221.4
					-
	Nov	4379.14	100	-4279.14	98500.5
	Dec	4379.14	100	-4279.14	-102780
					-
	Jan	4379.14	100	-4279.14	-107059
	Feb	4379.14	100	-4279.14	-111338
	Mar	4379.14	100	-4279.14	-115617
	Apr	4379.14	100	-4279.14	-119896
	May	4379.14	100	-4279.14	-124175

Jun	4379.14	100	-4279.14	-128454
Jul	4379.14	100	-4279.14	-132734
Aug	4379.14	100	-4279.14	-137013
Sep	4379.14	100	-4279.14	-141292
Oct	4379.14	100	-4279.14	-145571
Nov	4379.14	100	-4279.14	-149850
Dec	4379.14	100	-4279.14	-154129

10.6

```

0, Straight, Straight, 0.25
1, Corner, Right, 0.1
2, Straight, Straight, 0.13
3, Corner, Left, 0.09
4, Straight, Straight, 0.19
5, Corner, Right, 0.08
6, Straight, Straight, 0.1
7, Corner, Left, 0.06
8, Straight, Straight, 0.13
9, Corner, Left, 0.05
10, Straight, Straight, 0.63
11, Corner, Left, 0.12
12, Straight, Straight, 0.07
13, Corner, Right, 0.19
14, Straight, Straight, 0.19
15, Straight, Straight, 0.55
16, Corner, Right, 0.14
17, Straight, Straight, 0.48
18, Corner, Right, 0.09
19, Straight, Straight, 0.08
20, Corner, Left, 0.11
21, Corner, Right, 0.13
22, Corner, Left, 0.11
23, Straight, Straight, 0.76
24, Corner, Right, 0.16
25, Straight, Straight, 0.28
26, Corner, Left, 0.1
27, Corner, Right, 0.07
28, Straight, Straight, 0.13
29, Corner, Right, 0.04
30, Straight, Straight, 0.21

```