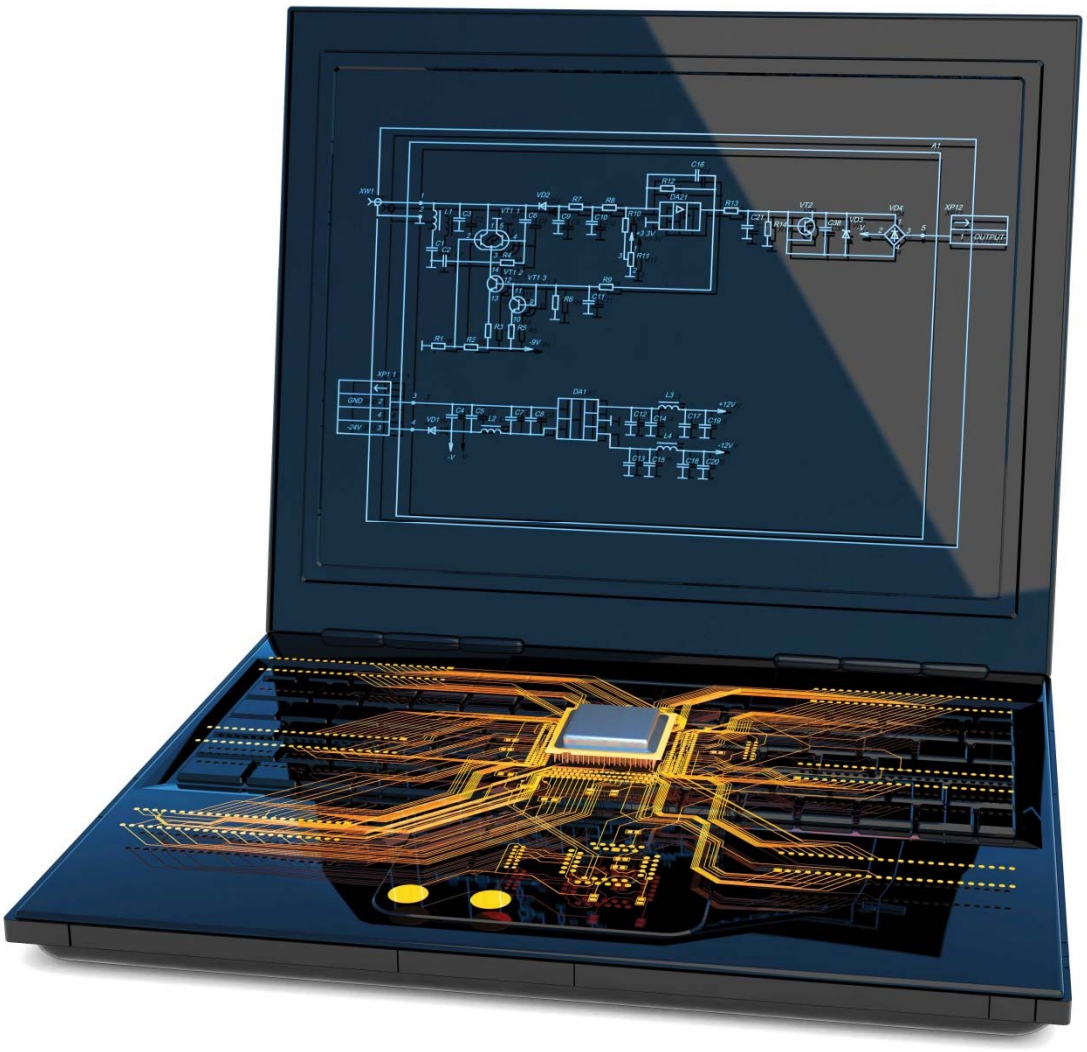
A-level
COMPUTER SCIENCE

Feedback on the 2019 NEA

Student response booklet

Published: Autumn 2019



Contents

Analysis

Level	Criteria	Mark
3	Fully or nearly fully scoped analysis of a real problem, presented in a way that a third party can understand. Requirements fully documented in a set of measurable and appropriate specific objectives, covering all required functionality of the solution or areas of investigation. Requirements arrived at by considering, through dialogue, the needs of the intended users of the system, or recipients of the outcomes for investigative projects. Problem sufficiently well modelled to be of use in subsequent stages.	7–9
2	Well scoped analysis (but with some omissions that are not serious enough to undermine later design) of a real problem. Most, but not all, requirements documented in a set of, in the main, measurable and appropriate specific objectives that cover most of the required functionality of a solution or areas of investigation. Requirements arrived at, in the main, by considering, through dialogue, the needs of the intended users of the system, or recipients of the outcomes for investigative projects. Problem sufficiently well modelled to be of use in subsequent stages.	4–6
1	Partly scoped analysis of a problem. Requirements partly documented in a set of specific objectives, not all of which are measurable or appropriate for developing a solution. The required functionality or areas of investigation are only partly addressed. Some attempt to consider, through dialogue, the needs of the intended users of the system, or recipients of the outcomes for investigative projects. Problem partly modelled and of some use in subsequent stages.	1–3
	No evidence presented	0

Student A

1. Analysis

1. Problem

1.2 Outline

Education is the basic, most important factor in our childhood, we learn to become smarter, some learn to chase dreams. It's shown that children don't enjoy sitting in a classroom copying notes and having to repeat the cycle every day until they enter high school. My main focus is around children within the ages of 8-10, these children will have to learn times tables. Due to modern times there is a push with electronic devices and by the age of 5 a child can use an electronic device without trouble to play games. It most likely is a shock to them when they learn that they have use paper and pen to write notes and then remember the times table off by heart.

1.3 Research

1.3.1 Prospective User:

The software I am creating could be used to make it easier for these children to learn their times tables. It will make the user think fast due to there being a timer and will make the user think as the moving cars will have different numbers on top of them. This could also be used as a revision tool for children to play in their spare time as learning the times tables from a piece of paper is never the best way to approach it. This software could also be used by teachers as a quiz, to identify who has learned the times tables in the shortest amount of time therefore finishing the game in the fastest time.

1.3.2 Problem Scope:

This is a problem all children within the primary school experience will face. Especially ones who are struggling with their memory or find it hard to memorise certain things, in this case it's the times tables.

1.3.3 Limitations of Current Situation

The current limitations of the existing times tables are that the amount of questions parents ask on the times tables. Parents are always mixed or unsure how to start teaching the times tables to their children or how to make sure what they learn at school is staying with them. A common question asked by the parents is: 'How can I help my child learn times tables?', there are always answers that may or may not help for example: 'Hang up a times table sheet' or 'Listen to some fun songs'. These solutions only work if the child learns the topic they are supposed to learn.

1.3.4 Existing systems/Investigation

To begin my investigation, I looked at a simple game of frogger to have a deeper insight of what my work could look like and what I should avoid to not make it too dull/boring.

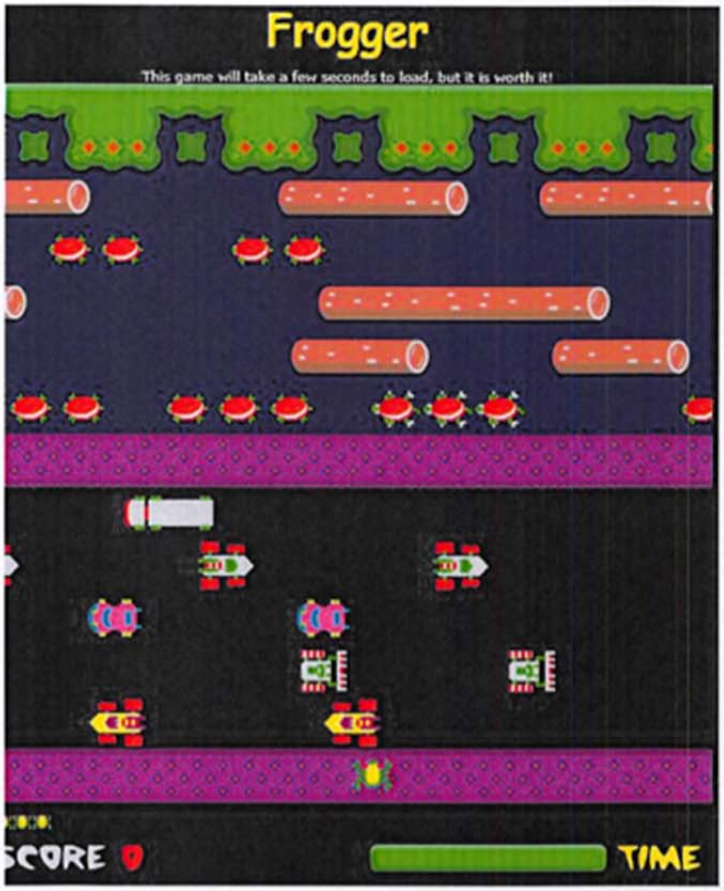
Since this is a simple game of frogger there is a score included, this indicates how far the player has gotten without dying or losing his life.

Similar to the Score the 'HI SCORE' indicates the most amount of point a players has ever gotten.

Frogs indicates how many lives the players have left, this means that if the players dies more than 3 times then the game will be over.

The time concept is there to remind the player that there is a restricted amount of time left for the player to reach the end. If the time runs out the game should reset, and the players loses 1 life.

This game has been designed in the style of an old/classic Retro game. This means that its all cramped, there are many things happening on the screen and the art style is really old. The 'TIME' is hard to spot without looking carefully and the top section with score and lives is cramped as well without clear and neat indication of both digits.



Similar to the first example I have used this game also seems to use the Retro style art scheme, both games look almost identical with slightly different colour and textures. The goals in both are the same, there are cars that the frog must avoid in order to reach to the other side of the road. After there seems to be turtles and logs which lead to the end goal. Both games use the same style of game logic, the frog has to avoid the cars, then jump onto the moving objects to reach the end.

Both games have a **'TIME'** counter, in the first example the time counter is displayed through a series of bars that go down indicating how much the player has left. It goes from green to yellow to red, the simple colour scheme used in traffic lights. In example two, the timer count down goes down through an animated green bar which doesn't change colour.

Both games have the **'SCORE'** counter, the first example used a retro style scheme where it has 8x0's and every time the players score it goes up. Example two uses a much simpler looking style where the number just goes up.

The **'LIVES'** are also done differently in both games, the first example uses a number style where as the second uses small frogs that indicate how much lives the player has left.

Interviews with Users

To conclude my investigation, I have asked 2 ICT students a set of questions to get a feel for what my software should look like and what they think a bad educational frogger game would consist of.

Question 1: What do you think about educational games?

Answer1: I believe that it's a good thing, they help people understand a certain topic depending on the type of game it is and the way it is presented to the user.

Answer2: I agree, they are good, however it really depends on what they consist of and how they are made.

Question 2: What does a 'good' educational game consist of?

Answer1: Objectives, a sense of achievement, a game that helps the user learn while keeping them engaged.

Answer2: Same here, at long as it doesn't look really bad and it keeps the players interest.

Question 3: Who's game would you be more persuaded to play, a teachers or a college students?

Answer1: I think I'd prefer a teachers as it has more consistency and focuses on the subject that the teacher is trying to teach.

Answer2: I disagree I'd prefer a college student than the teacher as the student still has a sense of creativity therefore the game might be more enjoyable than the one made by a teacher while keeping focus on the topic.

Question 4: The final question is: how did you approach times tables as a child and if you did struggle would an educational game help.

Answer1: As a child I really struggled with times tables, I found it really boring and often I would pretend to learn them while doing something else in the meantime. I found it more like a punishment there I think having to complete levels in a game would be more enjoyable.

Answer2: I didn't struggle therefore I approached it normally, I just found it really boring, so yeah maybe a game would be more enjoyable and helpful.

List of objectives:

1. Inform the player of how to play the game in the first screen.
2. Have working solution.
   2.1: The score section works according to the player movements and that the further the player moves the more score he/she earns.
   2.2: If the player hits a car he/she loses a life and therefore the game resets keeping the score since he/she will have 3 lives.
   2.3: Have a working timer that makes the player lose a life if times goes down to 0, this encourages faster thinking and competitiveness amongst players.
   2.4: If the player loses all lives then a prompt should come up allowing them to press a button to continue, this button will reset the game.
3 Allow the player to move inside the game, using directions buttons on the keyboard indicating the direction. This will require a set speed and delay between each movement to stop the player from zooming across the entire map.
4 Have multiple levels which in this case indicate the level of difficulty of the times tables. This will test the players to see who can get to the hardest level the fastest and how many lives he/she will have left.
5 Have a high score table with names. (An extra)
6 Every X amount of SCORE the player would be asked a times tables question, this is how the game will have an educational factor.

Student B

Analysis

Primary school is located in and teaches around 225 pupils ageing from 3 to 11. The school currently has around 35 staff and volunteers who attend the school 5 days a week. The 137,163 ft² grounds of the school is managed by one site manager therefore it is very difficult to control and restrict access to the school to comply with government regulations regarding lockdown procedures and personal safety.

Current System

Summary

Currently the security measures in place to restrict access to the school (when the grounds are open) are an entrance door followed by a second locked door containing an electric strike lock which is set to a Fail-Safe state meaning when power is lost it remains unlocked (this cannot be changed due to fire safety regulations). This door can be opened two ways; either the access control keypad (4-digit code) or via a button if the reception staff authorise access to a recognised individual. Staff and visitors must sign in via pen and paper once in the premises however this data is never digitalised making it extremely volatile.

Operation

The current system collects 4 digit user input, compares the user input with the pre-set password inside program and if the user input and stored password matches, access will be granted (by opening the door with the help of relay for a few seconds and closing it automatically after stipulated time) . If there is a mismatch between user-input and stored-password, access will be denied (by not opening the closed door – that is by keeping the relay in OFF position)

Security Risks

The current system uses a PIN to electronically open the door but is not changed often meaning the PIN could be accessed by an intruder and used to gain entry to the building. There is no record or transaction log of when the building was accessed meaning the door could be opened at any time without any staff knowing. This poses a new risk of burglary as the door could be opened at any time if the passcode is known.

Overall

The present system contains various security loopholes which could result in the pupils' and staffs' safety being put at risk. Therefore, a new and updated system is required to maximise the school's safety. A new and up to date access control system is needed to maintain the school's reputation and create a safer environment for the pupils and staff.

Updated System

Overview

The new system should use RFID tags embedded in the teacher's ID cards which they wear around their necks. The system will allow a staff member possessing a relevant ID card to 'tap in' instantly gaining access to the building. As the user 'taps in' their RFID tag data is read, and a staff ID number is cross-referenced to an Access Control List validating their access rights. This can then be sent to a database can then be used to create a digital staff register meaning there would be no further need for the paper register currently used. Any visitors can have a guest account created for them with a visitor's access pass that expires every 6 hours to prevent unauthorised re-entry.

Operation

When a credential is presented to the reader, the reader sends the credential's information, usually a number, to a control panel, a highly reliable processor. The control panel compares the credential's number to an access control list, grants or denies the presented request, and sends a transaction log to a database. When access is denied based on the access control list, the door remains locked. If there is a match between the credential and the access control list, the control panel operates a relay that in turn unlocks the door. The control panel also ignores a door open signal to prevent an alarm. The reader will provide feedback, such as a flashing red LED for an access denied and a flashing green LED for an access granted.

As an extra security level, the new system will include a camera which takes a snapshot every time the system is used therefore in the event of a security breach the intruder can quickly and easily be found and dealt with. If there is no one on reception, there will be a button that a user can press which takes a photo and emails it to senior staff who can then remotely authorise entry.

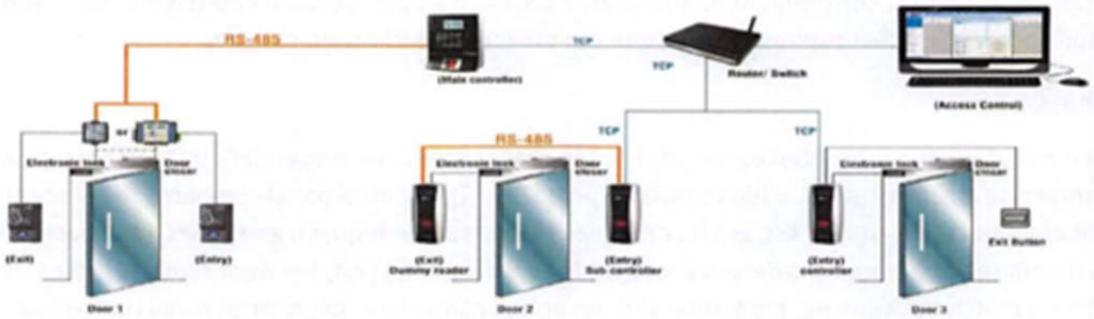
An unauthorised person could still gain access to an ID card and would therefore have full access to the building despite their snapshot being taken. To prevent this, two-factor authentication can be used. In a two-factor transaction, the presented credential (RFID) and a second factor are needed for access to be granted; another factor can be a PIN or a password. This would make the system more secure but would increase entry time and would be redundant unless an ID card got lost or stolen. This therefore could be used occasionally when a staff member reports an ID card missing.

The new system will also allow the whole school to be protected by access levels to restrict access to certain rooms or times which may pose a danger to pupils or staff. The school office for example maintains a large volume of personal files. Lots of sensitive information is kept in this room therefore can be restricted so only certain staff can access pupil records and school finances.

Existing Systems

The systems that currently exist and are being used tend to be much less specialised and less specific to a certain environment. Usually they consist of a keypad, a mag lock controller, a press to exit button and a mag lock which when connected allow two-way restricted access to a building however without paying thousands for a specifically designed system such as the ones used in airports and banks.

Such as:



Even this extremely expensive system does not allow remote access over the internet, so any remote authorisation is still local to the system so requires an access manager to be present on site to authenticate access.

Communication with the end user

I spoke to one of the school governors about the current system in place for access control. I wanted to know the specific protocols for staff and visitors and what I could do to improve the current system. He got back to me with the following email which explains fully the current system in use and what I can do to improve it. It also outlines the regulations regarding staff and pupil safety and the problems faced when designing a system like this in the public sector.

I am writing in my capacity of Chair of the Academy Council at . I am writing to outline the current entry and exit procedures that are in place within the school and some of the challenges with the current system. I will also outline some of the improvements that could occur as a result of a change in system.

The currents system is one that has been in place for several years. Essentially, access to the building is controlled through a single doorway that is situated alongside the reception area. Access from the reception entrance area to the main building is via a locked door. The door is released either manually by staff within the reception office or alternatively via a keypad.

The keypad entry system has been in place for several years and is relatively basic. The door is released using a 4-digit code. The system does not allow for individually assigned codes and as such there is one single code that is provided to all relevant staff. There are real challenges to maintaining the appropriate controls around who has access to this code and ensuring for instance that staff do not share the code inappropriately and making decisions around when to change the code.

The keypad system that is currently in place does not create a log of entry both in terms of 'who' and 'when'.

For visitors and those that do not have access to 'the code' entry into the building can only occur when a member of the administration team opens the door with the manual release button within their office space. This would occur when the administration team member is assured that the person is rightfully entering the building and provided them with a generic visitor's identification badge.

In addition to this physical check, visitors are required to complete an entry in the visitor's book. This is the only record of visitors within the building and acts to support the security requirements of the building and safety requirements in relation to things such as fire safety protocols. This process is a manual process that requires vigilance on behalf of the school staff to ensure that it is completed accurately and consistently. The system as it stands is labour intensive and always relies on the availability of administrative staff. This can have an impact upon staff members being able to undertake other duties. The paper record is not searchable in any systematic way would be archived in a simple format. This may create challenges if for instance, an incident occurs, and the records need to be searched.

The current exit and entry point do not currently record images of those that have passed through. As such there is no clear fool proof way to identify with an image, individuals who may have entered the

building and when access took place. it could be argued that the current system has weaknesses in relation to safety and security of the building and protection from people who may have nefarious intent.

A move to a more automated and sophisticated system could have significant advantages and, if designed appropriately, mitigate some of the challenges that I have alluded to above. An appropriate system would enhance the security of the building (both as a deterrent and in detecting and investigating incidents) improve aspects in relation to safety of visitors and finally be efficient in relation to staff member time.

Kind regards

Objectives

Primary Objectives:

1. Staff can quickly and easily enter the building using RFID tags.
2. Visitors can be given a visitor pass which expires after a set number of hours which is a temporary authorised tag.
3. A transaction log should be taken to monitor all entry activity.
4. A secure access control list should be created to cross-reference with the system to identify authorised staff.
5. A database should be created to register which staff have entered and exited the building.
6. There should be an automatic emergency unlocking protocol for fires/evacuations.
7. Staff should be able to access the building 24/7 to catch up on work or use school facilities.
8. All snapshots should be uploaded to the database to cross-reference the transaction log.
9. There should be an online interface in which senior staff can allow remote access to the door after authorising from a snapshot taken of the client.
10. There should be access levels corresponding to the authority of each staff member as to who can access which restricted parts of the building.
11. There should be an alert call to the site manager if the building is accessed outside of school open hours.

Remote Authorisation

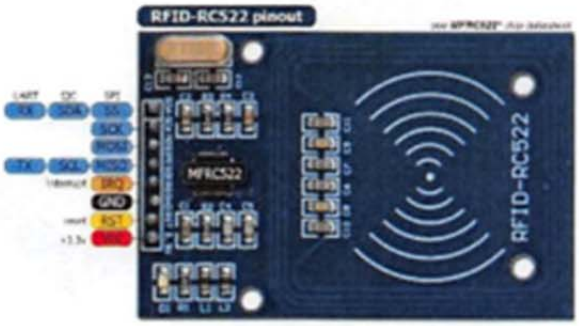
The online functionality of the program will allow remote authorisation of people trying to gain access to the building, this means when a user presses a button on the access controller it will send an email to a set staff list of a snapshot taken of them so they can be verified as an authorised user. The email will tell the user to open a viewing program containing a live stream of the user trying to access the door and two buttons either to authorise access or to deny access. The video streaming will use TCP/IP to send frames to a computer on the network. The TCP/IP program requires two part to work; a server program and a client program. The TCP/IP Transmitter sends packets to the default port and allows a specific IP address to read it. The TCP/IP listener will connect to a specified port and waits till a packet is received from the IP address of the camera. When it is detected it will decode the packet and read the data, in this case a bmp image. The image will then be processed and displayed on screen.

Controller

I am going to use a raspberry pi as a main controller as it contains a reliable processor and has Ethernet, USB and GPIO pins already soldered onto the board so no connector shields would be needed. There is also a camera module which makes it highly attractive for a project involving photo/video.

RFID Receiver/Transmitter

For the RFID functionality I am going to use a RFID RC522 board which can read and write tags so the pi can read a tag and cross-reference this with a database to either authorise or deny access.



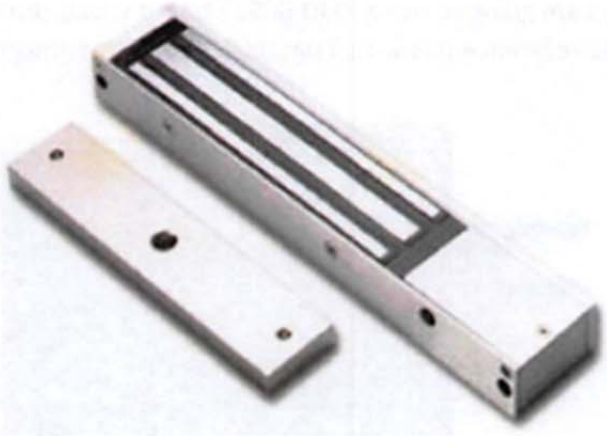
Relay

A relay is used as a switch to connect (non-electronically) two circuits together and trigger a maglo to unlock or lock the door.

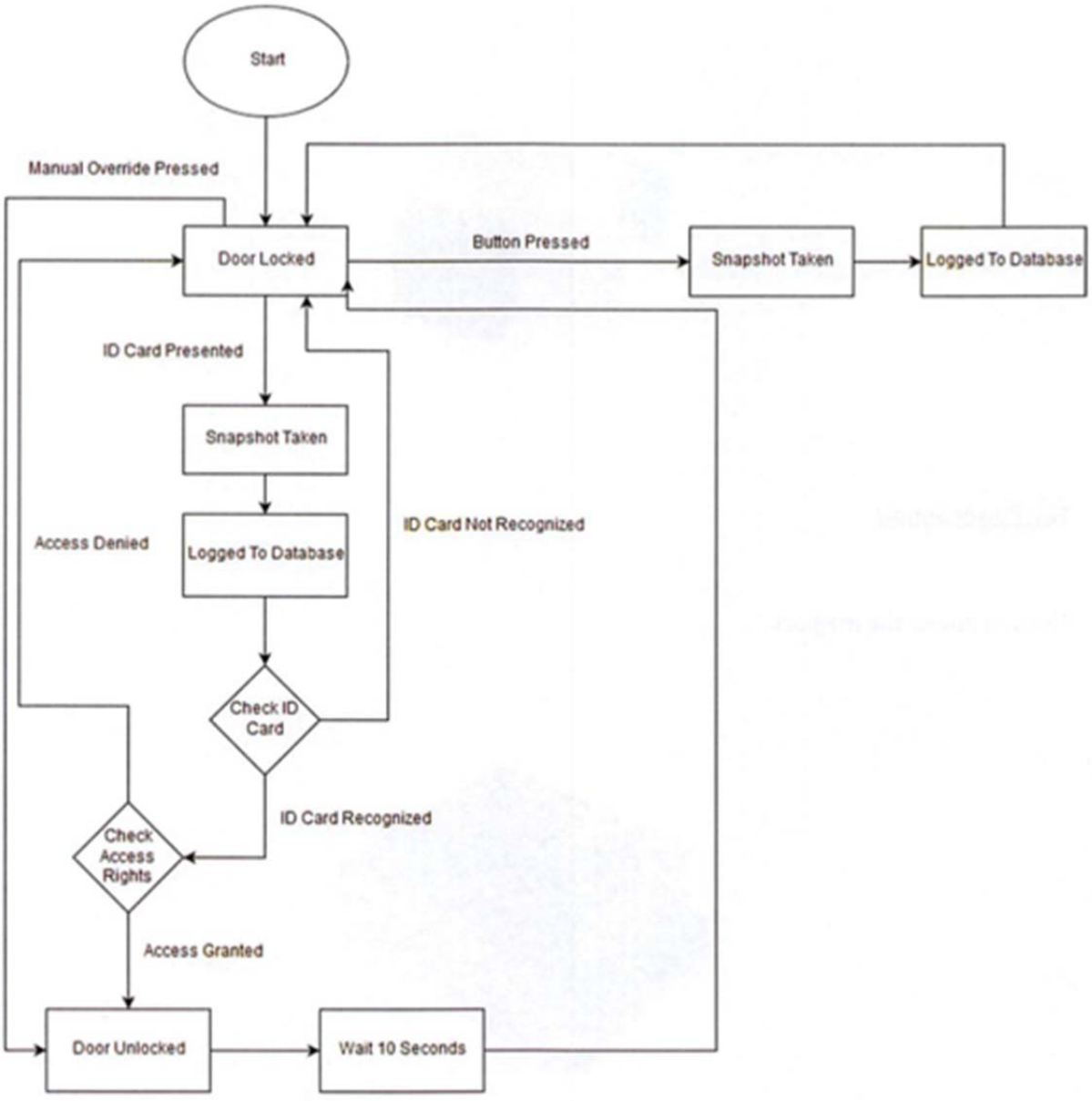
I chose this relay because it has an input voltage of 5v which is the output voltage of the GPIO pins on the raspberry pi.

Maglock

This is basically an electromagnet that gets connected to a switch, when it is on the door is locked (magnetised). When it is off the door is unlocked (demagnetised).



Flow Process Diagram



Student C

Analysis

Preface

The problem I will be solving with my project will be the lack in the market of relaxing simulation games, while helping to promote Formula 1 (F1) the sport. F1 has seen diminishing numbers as more and more people stop watching, so currently the FiA has begun a campaign to market the sport to the younger audience. This project will take a similar approach and try to encourage new people into the sport. Aiming at people between 16 and 24.

Introduction

The game itself focuses the player on becoming part of the side-line crew of an F1 team, giving the user new and different challenges and views on the sport as much of the work of the side crew isn't made clear on TV. When to pit, what tires to change to and weather management are just some of the few problems the player will have to try and overcome to win races and climb up the leagues.

At the start of the game you will need to pick attributes of the car such as acceleration, top speed and cornering on a slider system. And through the season you will be given the opportunity to increase your stats as your team researches new ideas and engines to improve the car. The player will spend the points they earn from races to afford the upgrades.

A big decision that the user will have to make is whether the driver should "push" the car to its limit or try to "save" the tyres duration for the long haul. Another important choice the player must make is when to come into the pits which in the F1 sport is also called "Boxing". The user will have to consider the weather, tire wear and if there is any other damage to the car. Trying to reach the perfect balance of time spent in the pit is a race winning must. In real F1 the call to box will be made over the radio to the driver. Next when the driver reaches the part of the track to pull off, he will slowly drive into the pits as there is a safety speed limit in that area. Then when in position, a team of 20 people will work as fast as they can to finish their work on the car.

Around 3 people will work on each tire, where one holds a bolt gun to remove the nut that holds the wheel. One will then grab the old wheel to remove it and the final man who is holding the new tire will put it into position. Finally the bolt is screwed back on. Other workers will prop the car up or do fuel.

While in the pits the player will need to work out which tires to equip onto the car between soft, medium, hard and wet. Each tire has its benefits and drawbacks as while soft is for maximum penetration, they don't last very long. And hard tires are designed to last long. On top of this the player will need to manage weather changes and any possible damages to the externals of the car. All while keeping an eye on what the competition does.

Other keys parts of the game will be random collisions between cars and random weather. Information like this will be displayed to the user in a text box. These mechanics aim to keep the game fresh and dynamic.

In summary the objective of the game is to work up the leagues and making it to the top, F1. Then win the constructors championship. The player will do this by winning many races on a range of tracks which will all require different strategies, and while winning these races they will gather points to improve the car to further get themselves up the ranks.

The target audience would be mature teenagers as they are the next generation of people to watch F1. They would be old enough to quickly pick up the finer parts of the game, but still be young enough to capture their imagination in the sport. As said before the objective of the game is to bring life and youth to the fan base of F1 and draw new people to the sport. I will be able to market towards this target audience by keeping the game mature, avoiding cartoon characters and unrealistic colours. Keeping the user interface clean and discrete will be able to keep the user entertained while focusing their attention on the game. Furthermore, customization is a vital part at attracting younger mature players because they are draw by the opportunity to make it their own and have an impact, like how they are growing and developing as a person, they love the ability to choose the strengths of their car and watch its power grow in their own hands. On top of this as the aim of the program is to have a lot of new people to sport, the program will need a comprehensive guide to how everything in the game itself works but also about key aspect of the sport. A mixture of tooltips, guides and possible videos can aid the user along a discovery of how it all works without the learning process becoming boring or stale.

Research

Games which inspired this idea are F1 2017 the official game which takes the point of view of the driver, allowing the player to control the cars every move during the race. This game is a big reason why I didn't create a driving/ racing game to market f1 because such a game already existed. But, from this game I did take the idea of personalising the stats of the car as within F1 2017 there is a skill tree where you can upgrade certain parts of the car, while this is very diverse and good for game play because my game is more based on simulating, a complex tree isn't necessary. Instead I opted for a slider system which can be periodically changed by the player as it is more effective for what I want to accomplish.

Here is the front cover of the game which uses pictures of the drivers and the cars to portray the game as a realistic simulator of the sport in the eyes of the driver.

Here is an example of the development tree in F1 2017 where the player can use options to pick which will improve the ability of the car.

While I have found phone games which are similar to what I want to achieve such as motorsport manager. I have used them as an inspiration to drive me to create this game. I wish to create something similar while adding to the design and mechanics such as a simulation mode so the player can fast forward time. Moreover to make the game more realistic to appeal to my mature audience I will add other parts of F1such as the safety car and car damage. Although this will make the game more complex to the player, it will also make it more interesting in each race. Also, I will add the idea of upgrading the car to the game

Here are some of the parts of this game I would like to recreate and improve on.

Having a driver with a strategy so whether to push hard or reserve for tire wear. Also this box shows how much longer the tires. I would like to add to this section the drivers energy as well as condition the car is in, whether it is damaged or not.

Random weather patterns to mix up the game play

The game will need algorithms for:

- The speed of the car, where the variables are the cars itself, weather, tires and strategy.

  I will need to set constants for the range of variables depending on the weight of importance I want to put on each variable. For example if the weather was raining the weather coefficient in the speed algorithm would be around 0.9 as the speed would be around lower in the rain. I worked this out by looking at qualifying times for the race tracks in the rain and in the dry, so for example Lewis Hamilton did a 1:42.553 in Belgium 2017 which was dry, while in 2018 in a wet Belgium he got a 1:58.179. "https://www.jamesallenonf1.com/2016/03/f1-performance-analysis-snapshot-of-mercedes-vs-ferrari-at-barcelona/"

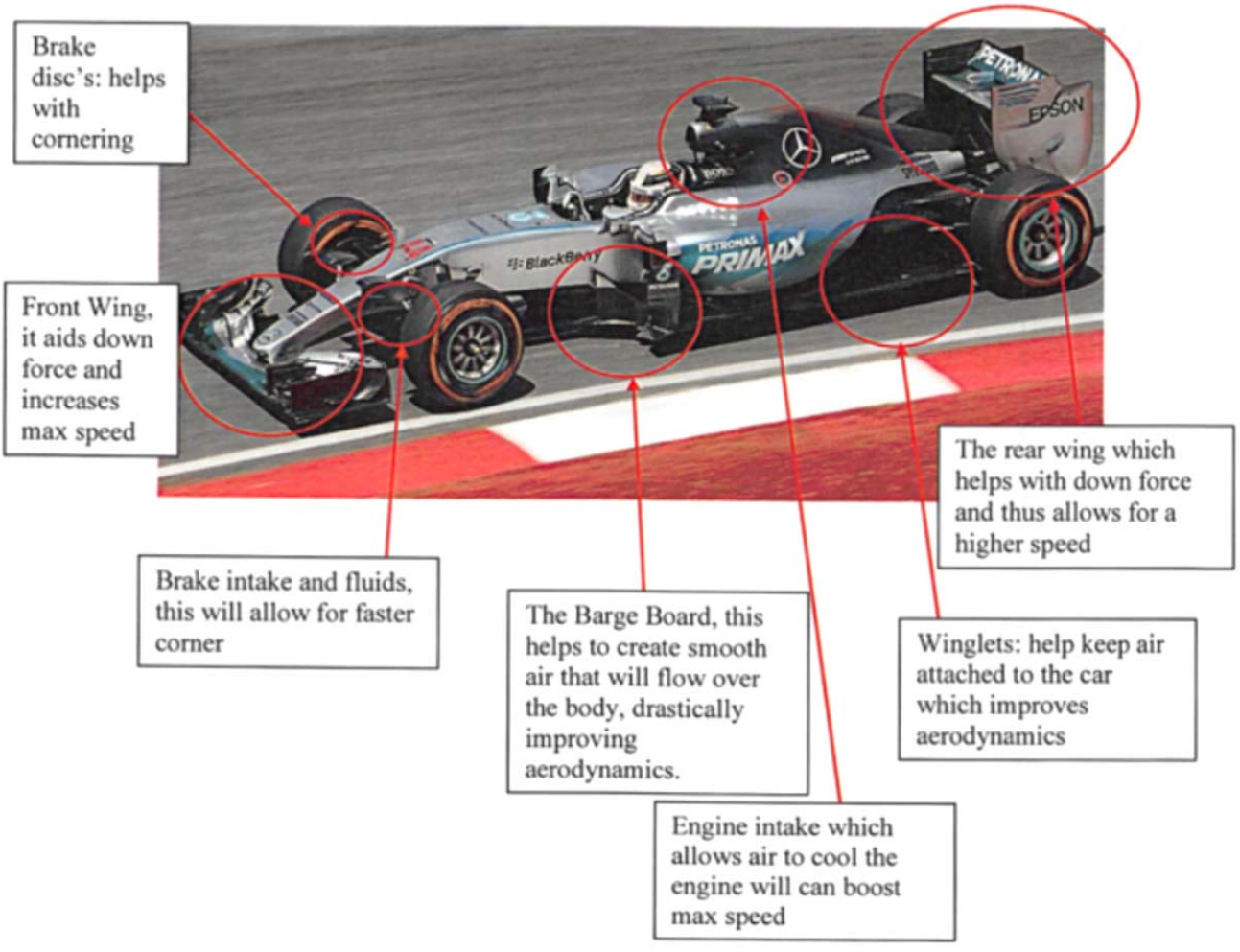
  When it comes to tires, Pirelli (the makers of F1 tires) states that on a single lap of a test circuit that the difference between each consecutive tire compound is around 1 second. This means that the difference between ultra soft and super soft is 1 second, and the difference between ultra soft and medium is around 3 seconds. To make this a more generic coefficient I will do it as a percentage and as 1 second roughly on most tracks results as 1%, the coefficient becomes (0.99 – (0.01 * n)) where n is the number of steps below ultra soft tires. If I have the extra time, my research shows that because the more soft tires have better grip they perform better when turning so if a certain circuit contains a large number of bends then the gain in having softer tires should be more than 1%, meaning I could create an algorithm that takes into consideration the number of bends to alter the statistical weight of having softer tires. Here is the data "https://en.wikipedia.org/wiki/Formula_One_tyres"

  The wear of tires: although softer tires may initially run faster, they degrade faster too. The ultra soft tires can last around 15 laps while remaining useful, according to data on https://www.quora.com/How-long-can-a-soft-compound-tyre-on-an-F1-lap-before-it-will-get-blown-off. But there are a lot of factors at play like fuel load and temperature of the track, such if the temperature of the track was much hotter the ultra soft tire would only last 12 laps. I think the best way to form an algorithm to give tires an accurate realistic wear would be to give each tire a base durability each scales based on the tire compound (so a ultra soft will have a base durability of 20, soft's would be 30 and so on with an increase of 10 each compound up) then this base durability can form the maximum laps the tire could last. Then in the algorithm the maximum durability would be reduced by factors such as weather and fuel load. Once the algorithm starts to outputs low enough numbers the program will simulate the tires losing grip which will slow the car significantly. The reason I wouldn't want to do tire durability in

kilometres instead of laps is because in more specific measurements, such as kilometres, the numbers aren't very accurate due to the effect of cornering on tire wear.
When it comes to fuel load, the more fuel the car has, the slower it will go. The player my decided to start the car with less fuel to be faster at the start but obviously when the car comes to pit, it will take longer as the car must fill up on fuel. Therefore the gain of the start becomes pointless unless the pit stop can be done during a safety car period but that makes it a risky strategy as it is impossible to predict.

Finally the car itself, the difference between the best performing car and worse one in current F1 is around 3%. This means when it comes to improving the car through research, each research should increase the cars performance by around 0.5%. Although it doesn't sound like much 0.5% each lap adds up over a whole race.

As part of my research and development within the game I will go through all the different engineered parts of the car and which improve aerodynamics, max speed and cornering.



Other algorithms I will require include a random variable for each car performance to make the game interesting, as if your car was always 3% faster than all over you would always win and it would be boring; therefore an element of random variables should spice the game up a bit. Along the same vein I will need an algorithms for random crash and DNF (do not finish) events.

Now that I have covered the upgrade able parts of the car, I can use these as options to pick from to improve the performance of the vehicle in the different regards. This gives the player the choice to spend their points on barge board research or engine performance. Knowing the names and effects of each part of the car will be useful for the designer to get the terminology accurate to create additional realism while also acting as a teaching tool to the target audience about the nitty-gritty of the sport.

Background

In my background I will investigate the more specific depths of the real sport of F1. Learning and gathering real data about a range of features within the sport to help me translate the ideas accurately to the game. By the end of the background I hope to have a strong idea of all the factors that must be considered on race day by the team; ranging from strategy, weather and tyres.

Engineer interview

I had the opportunity to speak to one of the lead engineers at McLaren. I'm very lucky to have contact within the F1 team due to frequent trips within their facilities. These trips were the inspiration for the project.

With the engineer I had the chance to discuss about the types of simulation software they use within their labs as well as the types of maths used.

They told me a lot of their maths and simulations are based of experimental data they collect by testing their car, they sadly couldn't give me a sample of the data to use, but I was able to get a better picture of average car statistics (which is surprisingly hard to find online). These statistics included average straight-line speed, corning speed based on radius, height of the car from the ground as well as coefficient of drag and friction for the car in different scenarios

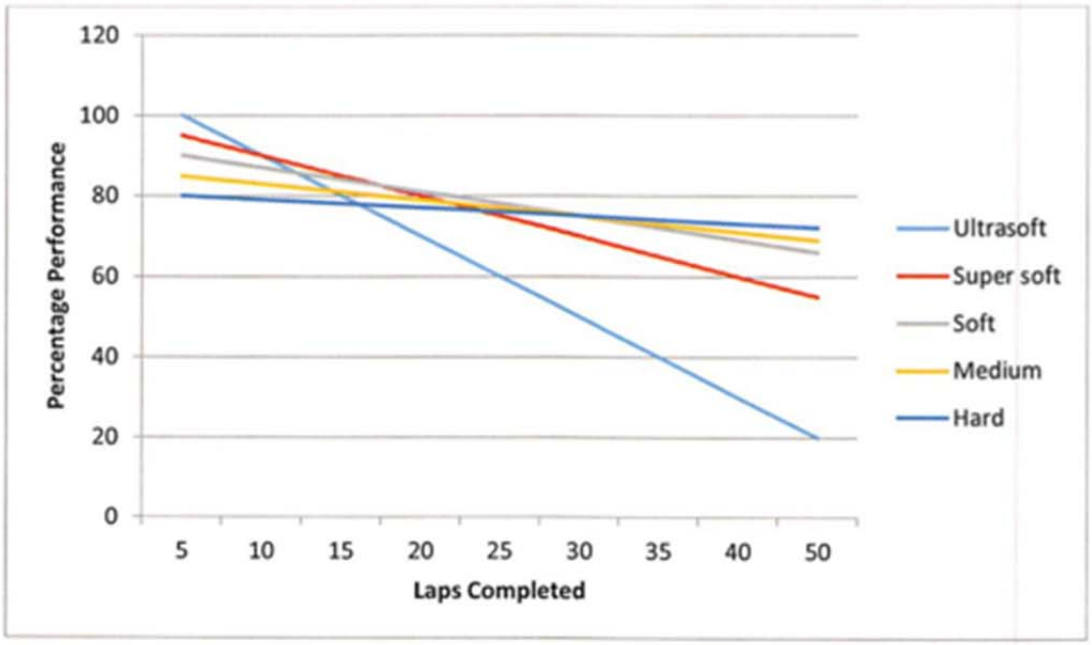
One thing I learnt from the meeting, was that their cars aren't the same from race to race. They have a range of aerodynamic packages which they pick from and fit to the car depending on the race track. For example, a track with more corners that straights will require a more downforce heavy package allowing for faster cornering. On top of this, race tracks such as Mexico-city, which are very high in altitude, require a heavy downforce package too. But this is because the air is much thinner at higher altitude, so downforce is hard to maintain.

From this meeting, what am I going to take to add to the program? First of all, I got a rough idea of how the complex side of the math works as well as a better idea of how the big three coefficients effect the car in positive and negative ways. These coefficients are: drag, aerodynamics and downforce. Next, I got an understand of what the values are of different parts of the car, this will help me create more realistic models of cars within the program by nailing the mathematical dimensions of it.

Finally, it gave me an idea to add to the program the capability to alter the values of the different coefficients before each track.

Tyre Performance

Here is a graph which shows tires and their performance based on distance. So, the ultra-soft start at the top of performance but quickly degrade to the bottom.



The performance axis is based on irrelative percentage numbers and the X axis is based on lap numbers. These figures are done for the average ideal weather conditions; temperature of 24 degrees Celsius and dry conditions.
Now to formulate the algorithms for each tire will need a linear regression model in the form $Y = mx + C$ where y is the value of performance and x is the lap number.

Hyper-Soft	$Y = -1.778X + 100$
Super-Soft	$Y = -0.889X + 95$
Soft	$Y = -0.533X + 90$
Medium	$Y = -0.355X + 85$
Hard	$Y = -0.178X + 80$

As show by the equations of the lines for each tire, as you move down into the harder tire compounds the performance decreases less quickly with an X coefficient which is closer to 0. I calculated these equations by first finding the gradient of each line on the graph using the formula of $(y1 - y2) / (x1 - x2) = M$. Then to find C you just take the original value of Y.

The reason why a tyres performance decreases after extended use is because the rubber gets ripped off and torn as can seen on the picture.

As the surface becomes unsmooth, it loses a lot of contact surface to the road meaning that its coefficient of friction to the track becomes slower so the car has less traction. Resulting in a lower top speed and reduced ability to slow down.

Each compound of tyre whether it is hard or super soft will have the same coefficient of friction at any given degradation. We can use a linear regression model to find values of the friction coefficient. At the start of the life of a tyre the coefficient of friction would be 0.9 so to find the coefficient of friction at a different point in the life span of the tyre we can use the equation:

$$Cf = 0.9 \times (\frac{Performace\ of\ the\ tyre}{90})$$

For example if we had a soft tyre after 20 laps, the performance as given by the table above would be (-0.533*20) + 90 which is about 80. So the Cf equals 0.9 * (80/90) which is 0.8

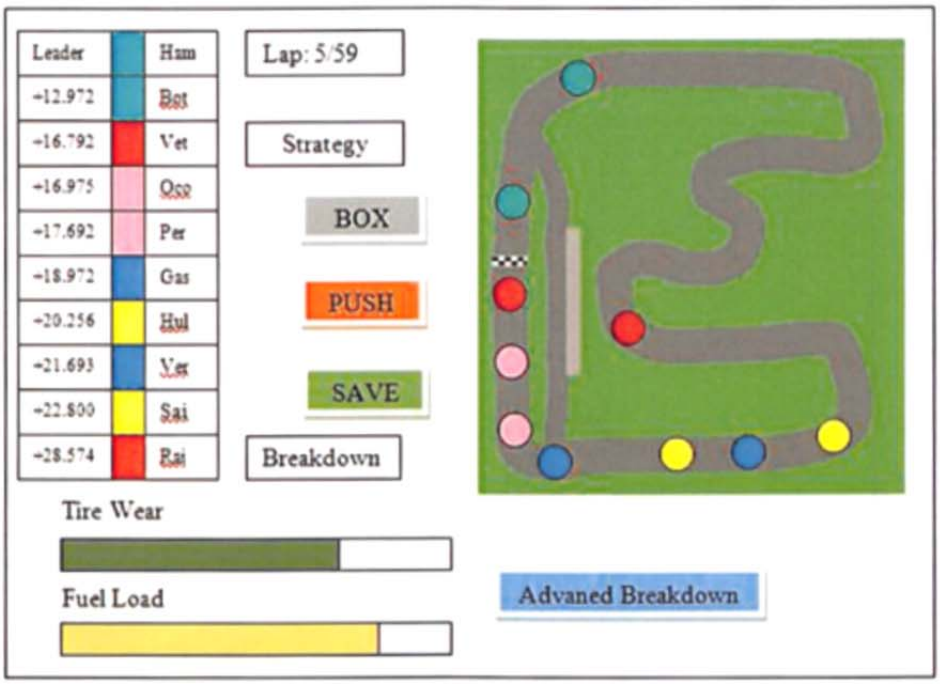
Wet Tyres

Now time for wet tires, of which there are two options; wet-soft and wet-hard, their performance to durability ration will be similar to the corresponding dry tires while in the wet. Obviously if a wet tire was used in the dry it would have a big lost in performance. This is because of how the wet tires tread work

On the left hand side is the wet tire treads but on the right hand side is the dry tires. As you can see, the dry ones don't have any treads but instead have a flat surface. This is because a flat surface will result in a higher surface area touching the track so treads means left surface area touching track which results in less grip, meaning a lower coefficient of friction so less speed.

If the player was to be on a set of wet tyres while it wasn't wet, meaning there was no water to keep the wet tyres cool and give the treads additional grip, the coefficient of friction would be 0.1.

Further pages of research and relevant formulas



Here is my graphical prototype for the user interface during a race. It shows the grid of racers which would update as the race unfolds; also, it shows the circuit with circles representing each driver as they move around the track. The buttons the users can use to play the game ("BOX", "PUSH" and "SAVE") are colourful to be easier differentiated and their corresponding colours connotes to their use. For example, "SAVE" is in green and it represents going easy to save the wear of the tires so the colour green shows recycling and saving.

I will give my prototyping to my end users to help generate more accurate feedback for the project as well as this, it will help them understand better.

End User

While it was useful to ask an engineer from a perspective of high-end simulations and the complexities of it. At the end of the day, this program is designed to be a game based for teenagers to enjoy. These people, that will play the game, are my end users and I needed to grasp feedback from them to greater target my game.

I approached a range of friends from school and online that I know play games. These are people that I know and play games with regularly as well as meet with. After pitching the idea, I gave them the opportunity to give some verbal feedback as we talked about the project.

First, the standard pitch: "I am building a game revolving around being part of the side line team of an F1 race. Instead of driving the car, your job is to manage a complex list of components and the race strategy. Asking yourself questions such as, should we pit this lap? Will the weather change? What tyre compound will help us

Objectives

My game simulates the job of an engineer in an f1 team to give a different perspective of the sport rather than just seeing the sport through the eyes of the driver like most f1 games. This includes managing tyre wear, strategy, pitting, damages and car stats instead of what most would associate with f1; driving. Within the game, data is given to the user using a range of graphs, data tables as well as visual aids such as an animation of the track and the user must decide the cars actions. While the program is a game for the user to play, it also doubles up as a simulation experience because there are portions of the game the user just must sit and watch the car race using the actions given to it. During this the user can just enjoy the simulation aspect.

While researching other similar programs they all contain a certain range of features which makes the game much more enjoyable and improves ability to play. Here is an objective list for in game features:

Saving Game Data
First, the game should have the ability to save the user progress. This save file should be linked to the user account so it can re-opened and the user can continue at a any time. Parameters which should be saved is data about the users car statistics, rank data such as how many points each driver and team has as well as the in-game date.

Pause Feature within the game
The next objective which I feel is very vital for the type of game I'm going for, is a pause feature. Portions of the game are very intense as a lot of different critical information is given to the user all at once, and the user needs to make a choice quick. The ability to pause is important to help give the user a little more time to properly plan out their strategy and enjoy the game without getting overly stressed on not making choices fast enough. Let me give an example, your car is coming up to the pit lane, but you just got a fresh set of tyres, so your current plan is to keep pushing and not pit. But in that instant, your car makes slight contact with another driver and you lose 30% of your front wing. You now have under 5 seconds to work out if you want to pit and replace the wing or wait until the tyres are lower to replace tyres and wing together later in the race, forcing you to drive with a damaged wing for a portion of the race. This is where a pause function comes in to help give the user to create a thought-out strategy with more time.

Professional Interface
As for the interface of the game, it should be comparable to a professional interface. This means it should be clean in how it presents data and information without given the user a sense of information overload. This can be solved by order the data through a tree structure and spreading it out through a range of forms. For example, instead of putting the settings, the race track and car data all on one form, create a central hub form which can then branch off into a separate form for which type of information to make it more manageable. Another way a professional interface sorts data effectively is with a concise colour scheme, for example naturally the colour red represents urgent pieces of information. Furthermore, the range of colours should be kept narrow to avoid confusing the user and make the interface seem messy. The aim should be a 3-colour scheme. On top of this all, the interface should cater towards our target audience of 16 to 22-year olds. Ensuring colours chosen are bright such as yellows

and blues. Also making sure the design is centred about modernistic choices for example integrated buttons.

User guidance built into the interface
The game should be graphically customizable, allowing the user to change the size of objects in the program to help them read or see it as well as the colour of certain objects in case the user is colour blind. On top of this the interface should be optimized to help new players understand the interface. This means the interface should have tooltips labelling objects and buttons as well as a page with information describing how the game itself works.

Cars should move around the track at variant speeds
The program should actually work. This should do this by simulating the movement of F1 cars. Achieving this requires the cars to move around the track through the use of manipulating objects position. The car objects should remain within the track lines as they race around the track. Furthermore, the speed/ how fast the game runs should be allowed to be altered. The user should have the ability to slow or speed up time if they wish to either enjoy the simulation experience or skip parts of the race they aren't needed.

Game should store and generate postrace data
Then at the end of the race, an objective for the program is to have a podium screen which can accurate show the end of the race positions as well as the amount of points which driver has gained by their race position

The Game should be efficient and run smoothly
In terms of the game play of the program itself, it should be able to run smoothly and effectively even on low spec machines. No one enjoy playing a game which is stuttering and laggy. This can be done through ensuring the code is clean and efficient, meaning that there is no unneeded loops or functions that can be better simplied down. This parametric can be proven through testing on a range of computer specs to make sure all users should be problem free. The test will be 2 hours of gameplay, testing with tooltips on, running through all the races. If during the whole gameplay session frames of the game stay above 45 then it will be a success. It is important that the gameplay sessions are quite long (2 hours) is as you continue to run through the game, the program will accumulate data and information, so the longer the user goes, the more data there is to manage which can put a lot of stress on the machine; which needs to be tested.

New maps should be easy to add
Also, on the topic of scalability, the program should have methods that make it quick and easy to add new track maps to the game. The tracks when loaded into the game should be saved on the file with the image of the map as well as coordinates and statistical data. Then the track should be playable in the game when the user next opens it.

Maps added to the game should be realistic
The race track map within the game should be of a real track to help it become an even stronger simulation of the real sport. Caters towards real fans as they can relate and recognize the track

The program should be able to run on low computer specs
This program is designed to run on even very low computer specs as we want the program to be widely accessible to anyone including those who can't get their hands-on high-end computers. The specs we would like to aim for is windows 10 64bit with at least 4 Gb of RAM, a dual core processor at a minimum of 1 Ghz clock rate. No graphics cord/ GPU is required.

The numerical values should be realistic
At the end of the day, a core objective for the game is that it should be semi realistic, while in real life engineers cant pause the race to think longer, the numerical values outputted by the game should be realistic as the program isn't just a game but also a simulation of the sport. While the maths for the program isn't the depth of a real f1 simulator because the specs of lower pcs wouldn't be able to handle it, the numbers of values such as speed and lap times should be within a 10% tolerance to real life values. Other small nuances such as pit time should be like the real sport. For example, in real f1 pits are between 2.5 and 3.5 second.

Account management
Regarding the management of the program. An admin should be able to change account passwords or usernames as well as create new accounts for others.

Data should remain integral and secure
Data should be kept safe and secure within the program with limited data breaks. The program should have a range of features to mitigate the possibility of an issue. The program should only allow one data leak in every year maximum.

Platform Discussion

There is range of different platforms I could design the project on, each with a range of advantages and disadvantages. Within this segment, we will go through these deviations between platforms and work out which one best suit our project.

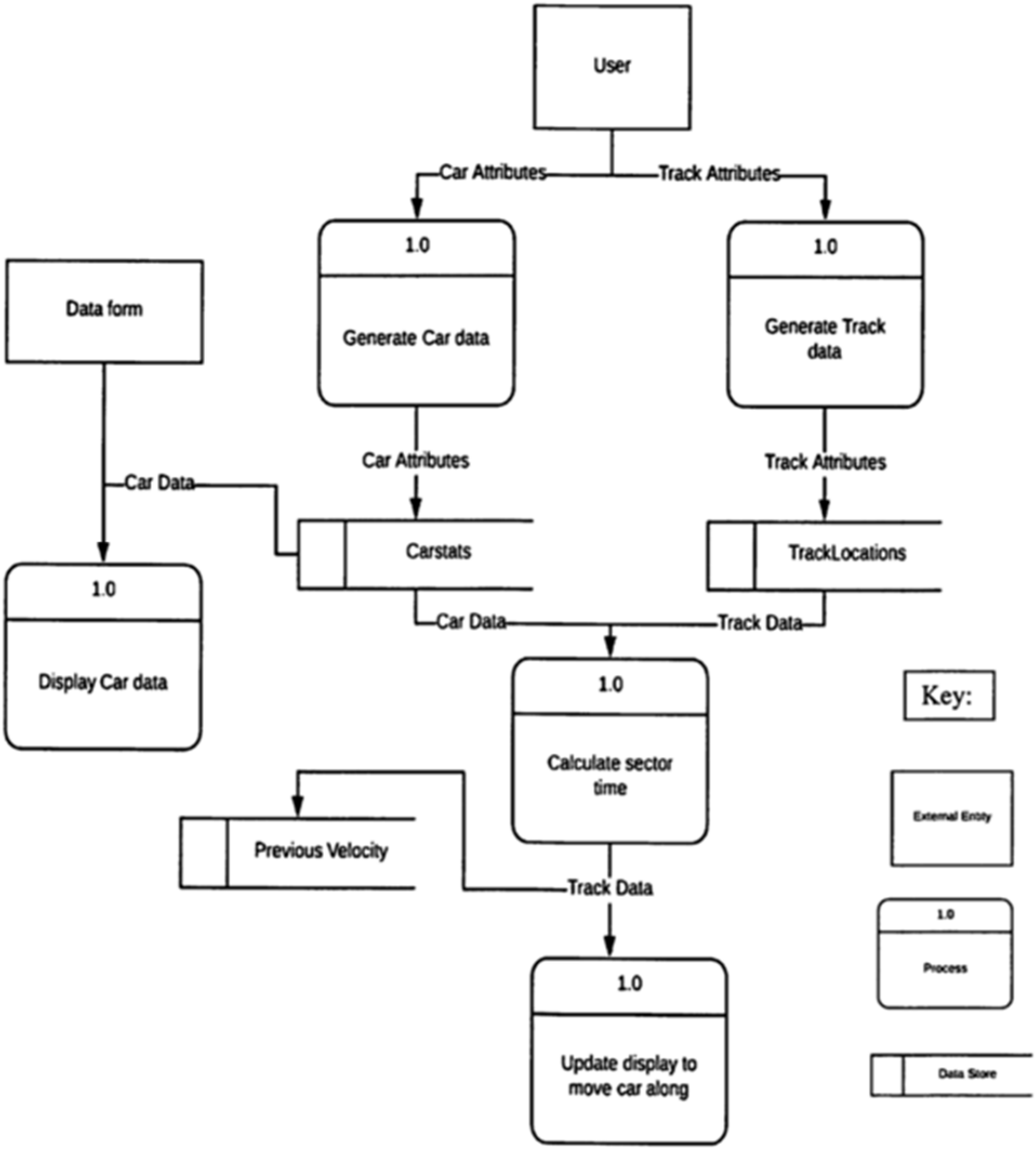
Visual Basics Forms

- This platform has an integrated graphical output display which will be useful to bring the game to life. Furthermore, this medium allows for a large range of different additional features and objects such as timers and textbox which will speed up the length of development.
- On the other hand, visual basic forms lack the ability to convert the game to an online presence. On top of this, there is no physics engine to any degree which will make the motion of objects difficult.

Unity

- At core, this is a platform which is made for creating games. It has a real time physics engine allowing a more complex array of force mathematics, as well as a large collection of online resources.

Level 1 – DFD Identification of Key Processes



Documented design

Level	Criteria	Mark
4	Fully or nearly fully articulated design for a real problem, that describes how all or almost all of the key aspects of the solution/investigation are to be structured/are structured.	10–12
3	Adequately articulated design for a real problem that describes how most of the key aspects of the solution/investigation are to be structured/are structured.	7–9
2	Partially articulated design for a real problem that describes how some aspects of the solution/investigation are to be structured/are structured.	4–6
1	Inadequate articulation of the design of the solution so that it is difficult to obtain a picture of how the solution/investigation is to be structured/is structured without resorting to looking directly at the programmed solution.	1–3
	No evidence presented	0

Student A

2. Project Design:

General Explanation

The system will run in XNA therefore the file that the user will be using is an .exe also known as an executable. This is a standard type like any other you may use one example is Microsoft Word. Once the user runes the executable the game will load a start screen, this start screen will consist of instructions and a simple introduction to the game. The start screen will ask the user to press a specific button which will trigger the actual game. The start screen will also have a screen that will trigger the high score screen that shows the highest scores of past runes of the game.

Data Dictionaries:

Game1 Class:

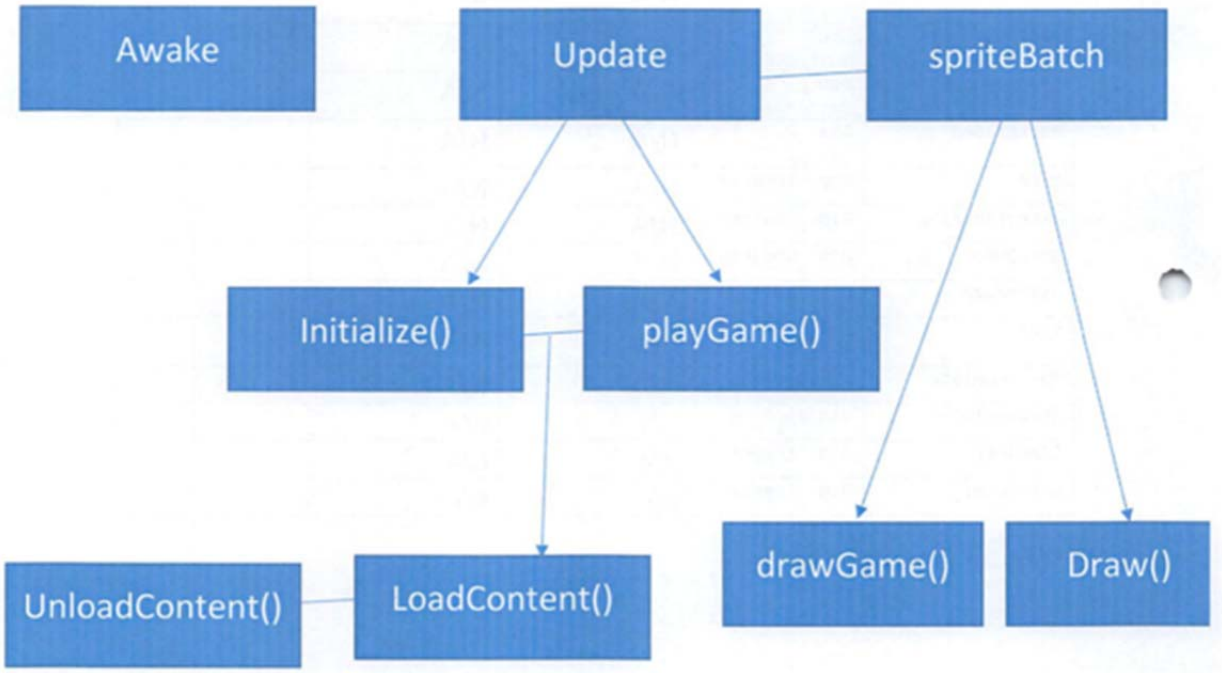
Data Item	Data Type	Validation	Sample Data	Purpose
SCREEN_WIDTH	Const Integer	N/A	800	
SCREEN_HEIGHT	Const Integer	N/A	480	
Car_Width	Const Integer	N/A	70	
Car_Height	Const Integer	N/A	(Car_Width \ 4) * 3	
Frog_WIDTH	Const Integer	N/A	Car_Width	
Frog_HEIGHT	Const Integer	N/A	Frog_WIDTH \ 2	
Frog_SPEED	Const Integer	N/A	10	
START_SPEED	Const Integer	N/A	60	
SPEED_STEP	Const Integer	N/A	5	
Lives	Const Integer	N/A	3	
LivesLeft	Dim Integer	N/A	N/A	
gameState	TgameState	N/A	N/A	
SCORE	Dim Integer	N/A	N/A	
Car1_Speed	Dim Integer	N/A	N/A	
clock	Dim Integer	N/A	N/A	
speed	Dim Integer	N/A	N/A	
splashScreen	Dim Rectangle	N/A	N/A	
tsplashScreen	Dim Texture2D	N/A	N/A	
tCar1	Dim Texture2D	N/A	N/A	
Car1	Dim Rectangle	N/A	N/A	
tCar2	Dim Texture2D	N/A	N/A	
Car2	Dim Rectangle	N/A	N/A	
tCar3	Dim Texture2D	N/A	N/A	
Car3	Dim Rectangle	N/A	N/A	
tCar4	Dim Texture2D	N/A	N/A	

Car4	Dim Rectangle	N/A	N/A	
TFrog	Dim Texture2D	N/A	N/A	
Frog	Dim Rectangle	N/A	N/A	
myKeys	Dim KeyboardState	N/A	N/A	
oldKeys	Dim KeyboardState	N/A	N/A	
tBackground	Dim Rectangle	N/A	N/A	
Background	Dim Texture2D	N/A	N/A	
time	Dim Integer	N/A	N/A	
DirectionFlag	Dim Boolean	N/A	N/A	
gameOver	Dim Boolean	N/A	N/A	
alternate	Dim Boolean	N/A	N/A	
font	Dim SpriteFont	N/A	N/A	
gameComplete	Dim Boolean	N/A	N/A	
Checkpoint	Dim Boolean	n/a	n/a	
timeleft	Dim Integer	n/a	n/a	
moveCount	Dim Integer	n/a	n/a	

OOP Design

Class Game1
Private procedure graphics
Private procedure spiteBatch
Public procedure new()
Protected procedure Initialize()
Protected procedure LoadContent()
Protected procedure UnloadContent()
Public procedure playGame()
Protected procedure Update
Public procedure drawGame()
Protected procedure Draw()

Hierarchy Diagram:

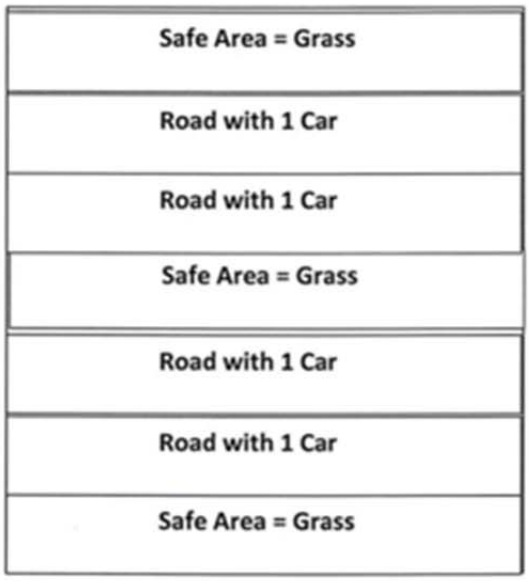


User interface design:

Stage 1 (Starting screen):

In the first stage of the executable the user will be greeted by a splash screen that has the following: Image of frogger, instructions, start button, and title frogger. This will be the first stage of the game. This stage will welcome the user and familiarize them with the keys and how the game works

Stage 2 (Running the game)



This is what the user will see once 's'/enter is pressed, the user will then go on and try to complete the game.

Commented code:

Please find the commented code in appendix 1.

Student B

3 Documented Design

Languages And Libraries:

The main program is going to be programmed in python, however with the control system running on c/c++ in the Arduino. Because of the complexity of certain aspects of the project - relating to computer vision specifically it would not be efficient to program entirely from scratch certain packages will be imported to use, these include:

- Open-cv (computer vision library)
  - The module includes pre-built functions to complete primitive image manipulation; this includes transformations, but also allows for more complex functions such as object recognition and tracking. The library allows for 2D arrays to be shown as images. In general the library can be considered to have similar functionality to image manipulation software but also has the possibility of more specific features for tasks.
    - For my project I will be using the transformation modules as well as the functions to convert between colour spaces and data type. i.e RGB 8bit int to BW 8bit int. In order to see the results of my operations I will also use the display features.
- Numpy (efficient mathematical operations)
  - Because Open-cv uses 2D arrays, it also is reliant on the Numpy module. Numpy 2D arrays allow for additional procedures to be performed and since opencv uses this type of format, if I also use it, it will mean that conversions are not required and as a result arrays can be processed faster.
- Tkinker (For developing the GUI)
  - A basic GUI creation module, the module allows for buttons and images to be drawn onto a canvas and so is relevant to my project for allowing options to be selected.
- urllib2 (to access the data streamed from the camera)
  - This allows the data from the website to be extracted in its raw form, meaning that individual bytes can be taken and the manipulated in the code.
- serial (to send data along the USB to the Arduino)

MAIN FUNCTIONS: (CRITICAL - THE FOUNDATION PROCEDURES)

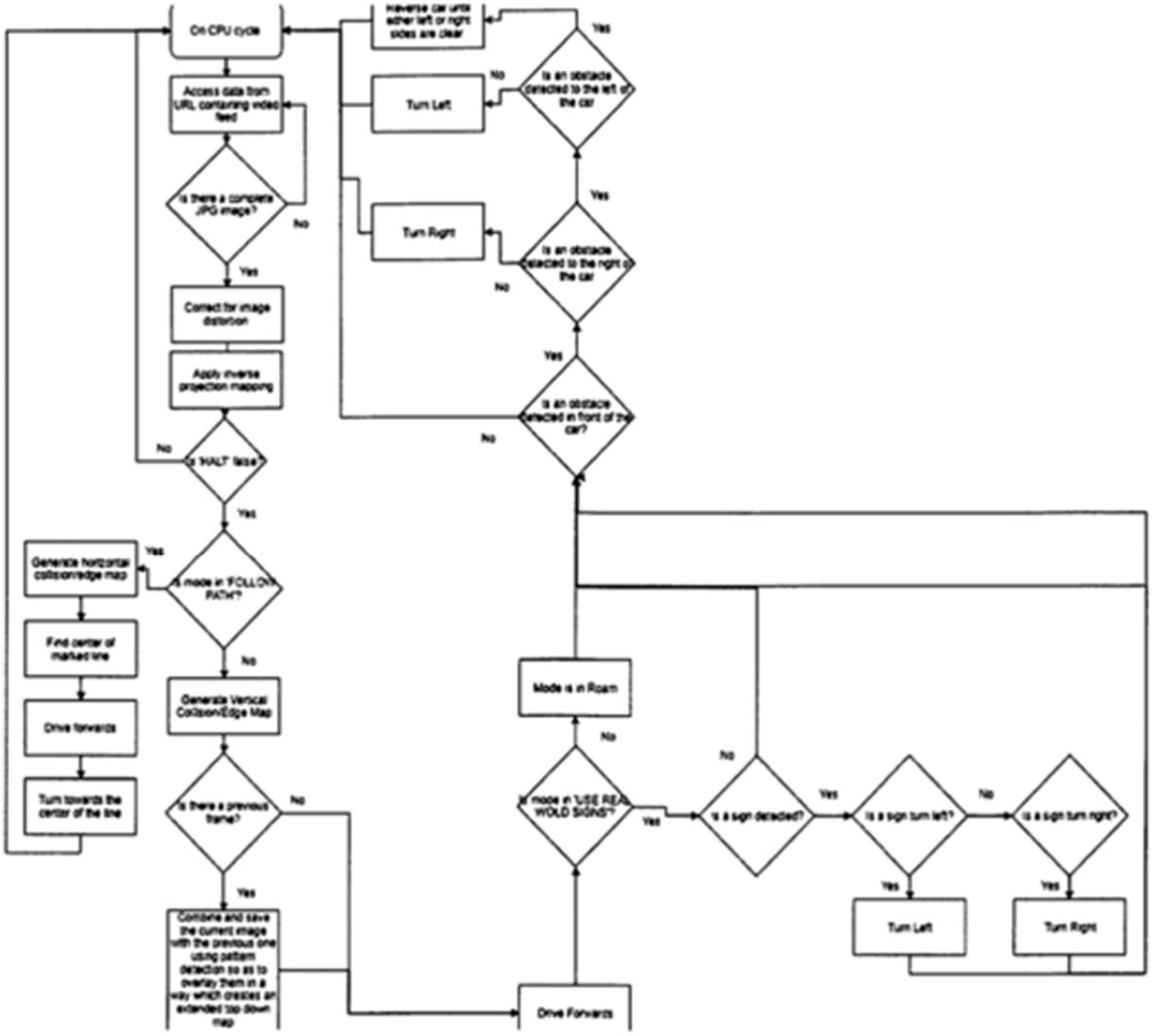
These are the functions which make up the main features of the program, since they are the ones that the rest of the functionality can be built upon. These are my own algorithms, some like the edge map creation have been slightly based on algorithms which already commonly used but have been modified for more specific purposes.

- Obstacle detection
  - Includes the edge map creation which is used to decide the distance between any harsh edge (assumed to be generated from the border between the floor and an object) from the car.
- Roam area
  - The logic involved in how the car will navigate its surroundings and what to do when obstacles are detected.
- Drawing Map
  - Taking the inverse projection mapping which removes the perspective from an image and relays a 'top down' view and stitching multiple versions of this together as the car drives about.
    - So at frame one an IPM image is taken
    - At frame ten; another IPM image is taken
    - These two images are then mapped together so common areas can be seamless stitched together.
    - The cycle then repeats, applying the next frame onto the image with the already combined images. This will mean that every image added will be appended to the map.

MAIN FUNCTIONS WHICH ARE USING LIBRARIES EXCLUSIVELY TO COMPLETE THEIR FUNCTIONALITY:

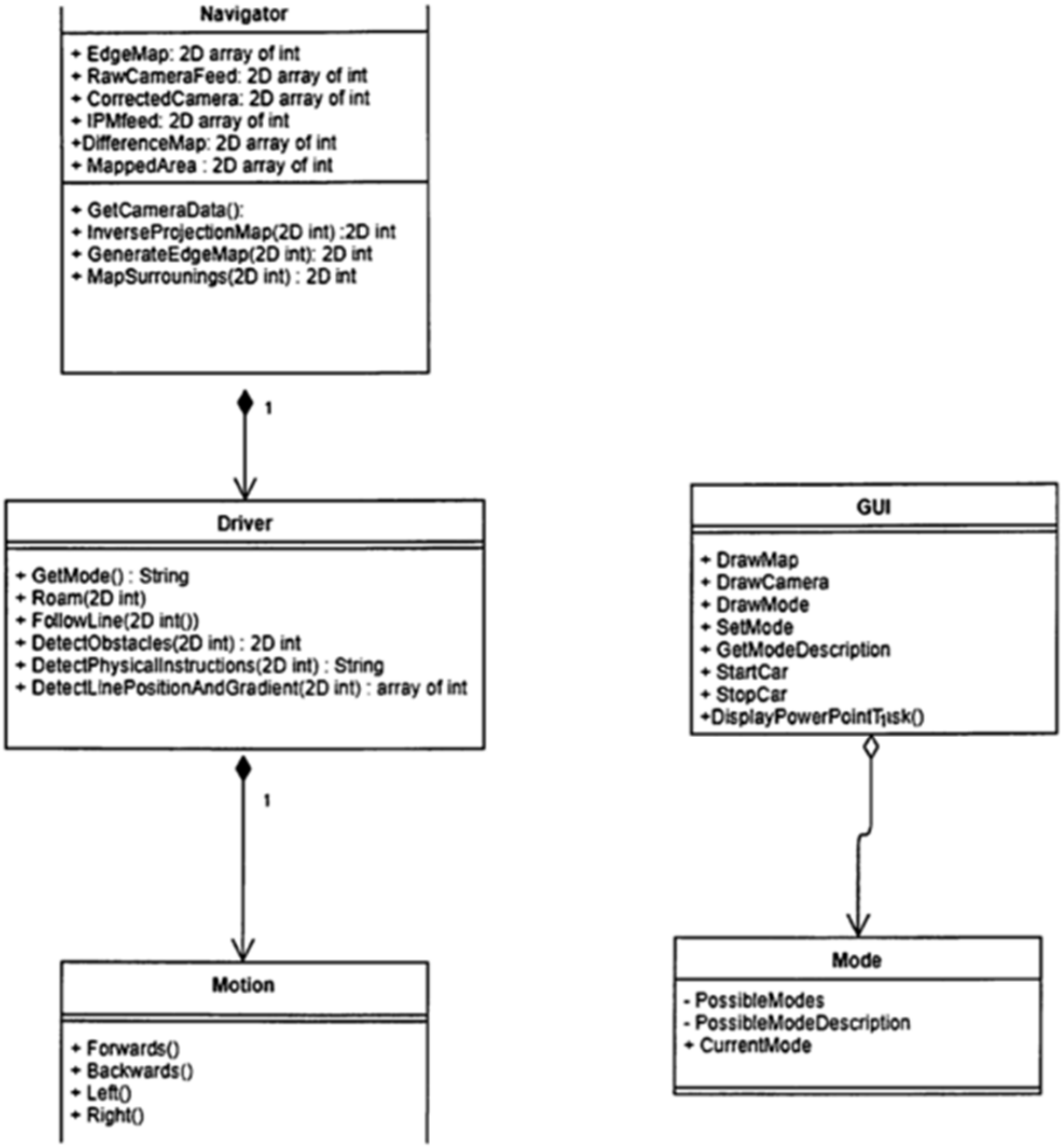
- Inverse Projection Mapping (uses OpenCV to complete the image transformation)
- Interpret physical markers (uses a library for the main portion of this function)
  - By looking at symbols of some sort on the floor to tell the car to move in a certain way.

SIMPLE FLOWCHART OF REQUIRED OBJECTIVES:



The modes, i.e 'FOLLOW PATH' or 'USE REAL WORLD SIGNS' are options selected by the user. When an option is selected then the logic will follow the above logic. The above is meant to be a logic diagram will enough simplification that it allows for a fairly fluid easy way of seeing how the code flows.

…



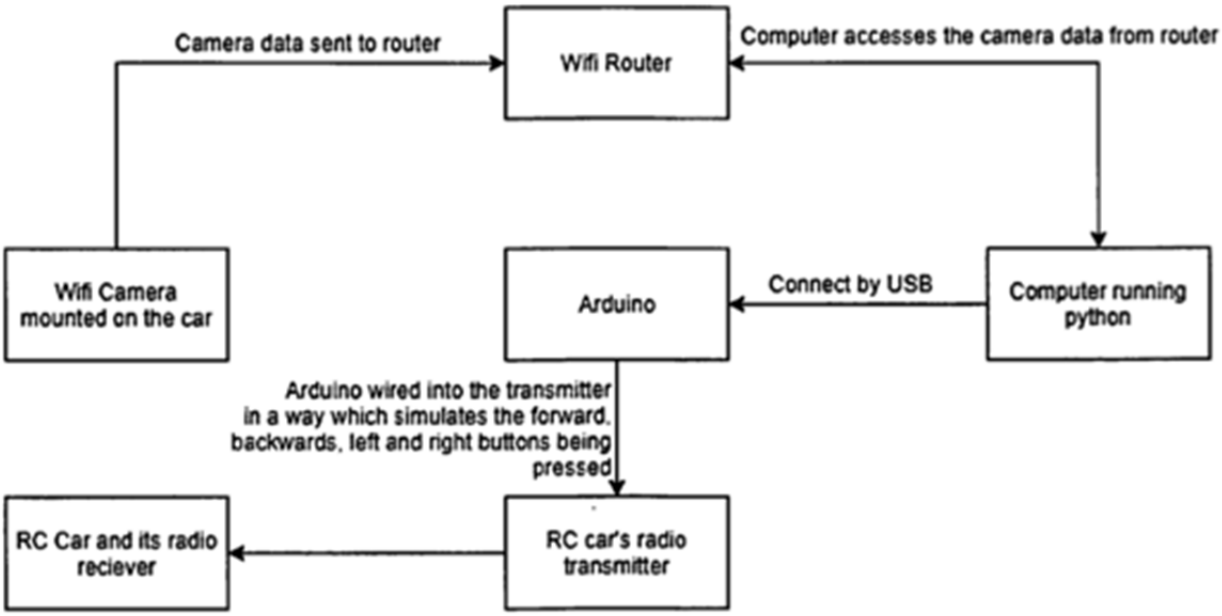
Using a class diagram structured as above will make the task easier to break down, add to and alter. Potentially the system could be written without using a class structure, however, by using it, the code will be more legible and also should me simpler for me to debug and make any modifications. As well as this, it leaves me the possibility of extending the system to encompass more procedures without having to restructure anything.

The navigator, driver and motion classes interact in a way similar to that of a ship navigation system; it all starts with the navigator which who interprets the surroundings, the driver then gets the information for where to go from the navigator, but in order to move the driver contacts the engine room or motion to get the car to move in the desired direction.

PRELIMINARY SETUP:

- RC car being fitted with a wifi camera
  - Wifi camera data sent to a server which can then be accessed as a HTML IP address
- Camera Calibration
  - Since the image is from a wide angle lens, there is going to be barrel distortion which needs to be corrected for in order for straight edges in the real world to be correctly displayed. In order to this, the camera needs to be calibrated. The topic of camera calibration is very complex and not relevant enough to include in the project, as a result, I am using pre-written code to calculate the intrinsic camera data which can then be used to correct the camera in my code.
    - https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-333b05afa0b0
  - The above link is a good guide I intend to follow in calibrating the camera.
- Radio remote being controlled by the Arduino

The Arduino acts only as a way of sending data from the computer to the car - it does not process any camera data. It has the sole purpose of receiving data from the code along a USB and toggling the correct pins on the RC remote to get the car to turn and drive in the correct direction.



ACCESSING CAMERA DATA STREAM

Input Data:
The main set of input data would be the data stream from the camera, the stream will be in the mjpeg format. This means that each frame of the footage is individually stored without relation to any others. The camera data needs to be streamed from the server, and since jpeg files start and end with specific bytes the positions of these need to be stored and the contents between the two decoded to get a single frame of the image. Each frame starts with the hex: **FF D8**, and ends with **FF D9**.

Pseudo Code:

```
URL = OpenURL('local IP address broadcasting the mjpeg data stream')
#Set the destination of the wifi camera's data
StreamData = ''
WHILE GetCameraData = TRUE:
      StreamData += URL.read()  # bytes of the data stream at a time
      FrameStart = StreamData.find('\xff\xd8')
      FrameEnd = StreamData.find('\xff\xd9')
      IF FrameStart exists AND FrameEnd exists:
            FrameBytes = StreamData[FrameStart to FrameEnd]
            RawCameraFrame = opencv.decode(FrameBytes , int8, AS an RGB
image)
            # Gets a 2D array which contains the pixel colours
            StreamData = ''
      END IF
End WHILE
```

NOTE:

- OpenCV uses numpy's 2D array, using a set width and height to represent each pixel which is stored as a 1D array with a length same as the number of colours. Therefore each pixel can be accessed as image[column,row] = [b,r,g] (opencv swaps the order around)

INVERSE PROJECTION MAPPING:

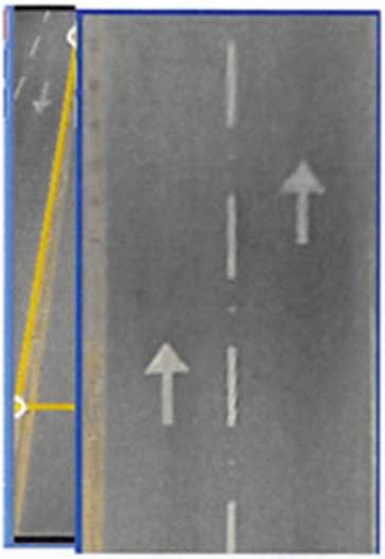
The inverse projection mapping section is a function in the OpenCV module which I am using.

The sort of transformation of the image would be called a perspective transformation since I am taking the initial image which will be showing one point perspective with a converging point in the middle of the horizon.

To get the 'top down' perspective:

- A rectangular shape needs to be drawn on the floor in front of the car with the camera in the correct position so as to mark out the boundaries of what the camera is able to see
- The position of each corner point needs to be recorded in terms of its screen space.
- The new position these corners need to be assigned and should be calculated using the known widths and length of the rectangle.

The perspective transformation function is written in the opencv module and will be used accordingly to shift the image into a more usable format.

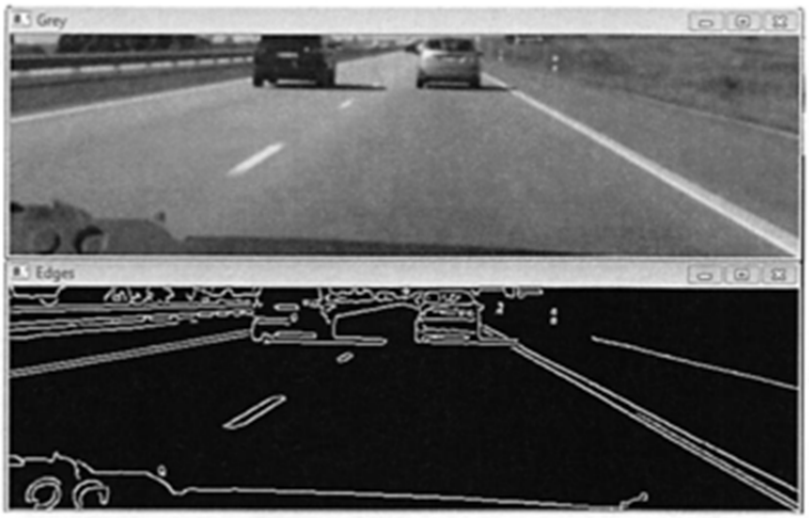


The left image shows a perspective image where a trapezium has been drawn, the road is perfectly straight however becauses of the location of the camera, the road is elongated and narrows at the end due to perspective. To correct this, the length and width of the section of road needs to be known. By taking the furthest two corners and pulling them outwards so as to be directly above the corresponding lower corners a rectangle is formed. By knowing the length and width the new rectangle's ratio can be adjusted to match that of the known lengths. Resulting an undistorted view of the surface in front of the car. This process does also have some caveats- the most impactful one is that no information about what lays behind an object is stored, so when the IPM procedure is run any object will become stretched.

Generate Edge and Collision Map:

The purpose of the collision map is to create a binary image which outlines areas of high contrast in the top down image. These areas of high contrast can be considered to be boundaries be obstacles and the ground. (The input image is black and white.) The black and white image could be based on the intensity of the full colour image or it could be based on colour hue, since we are dealing with contrast, experimentation will be needed to determine whether change in a pixel's colour or in its brightness is most reliable.

A common algorithm is canny edge detection, this is included in the Open-cv module and so I will include an option on whether to use that or my own edge detection system.



(the above image shows the canny edge detection algorithm being operated on the top image, similar to my own algorithm in functionality)

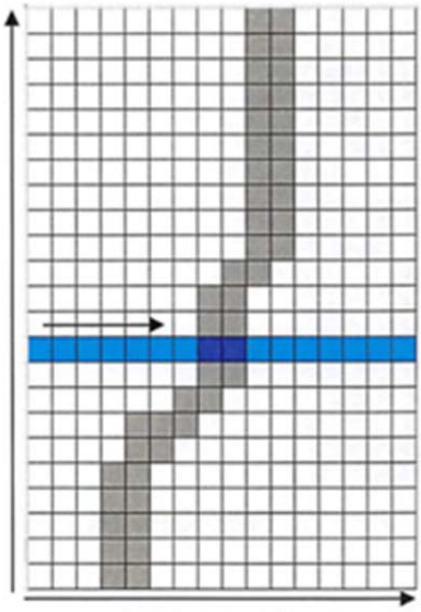
Operations which need to be included to create a useful and usable collision map:

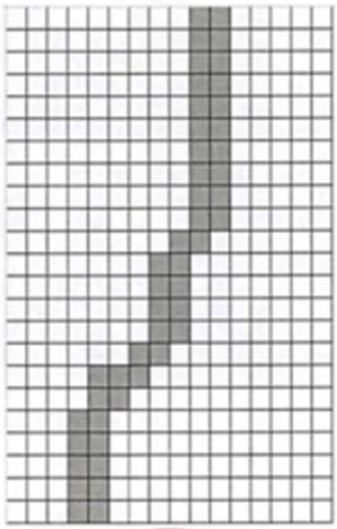
- Noise reduction
  - Since the captured image is likely to contain lots of artefacts due to a low-quality camera - these features need to be removed so the edge detection algorithm does not pick up these and include them and falsely mark them as edges.
- Edge detection
  - An inbuilt solution included in the module to this problem is the canny edge detection.
  - I will program my own edge detection algorithm and take into account the execution time limit- making optimisation a priority

Edge Detection Algorithm Prepositions:
Because the modes, 'ROAM' and 'FOLLOW PATH' operate on different premises the way edges are detected needs to be different for both.

- FOLLOW PATH:
  - This mode will tell the car to follow a line drawn on the floor in front of the car without consideration for obstacles.
- ROAM
  - The car is avoiding obstacles and moving around the surrounding area.

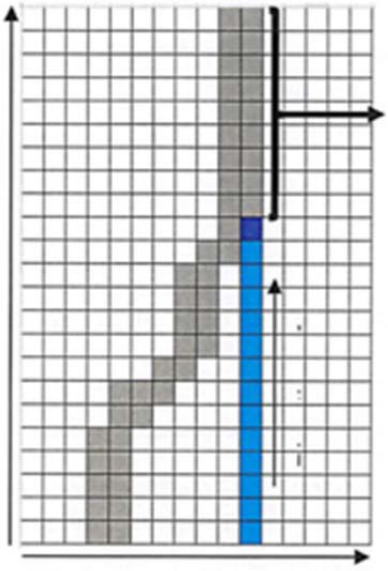
The edge map, is not used as a collision map in FOLLOW PATH mode, when the car is in this mode the edge map is used to determine the center of the line (or path) drawn on the surface. Detecting edges by iterating left to right across the image and calculating the change in intensity the line drawn on the ground for the car to follow is more likely to be correctly identified.





Positio
n of
the car
and its
camer
a view

The above shows that moving horizontally would be more desirable. If the detection would be moving perpendicular to this, a perfectly vertical line drawn in front of the car would not be detected since there would be no change in intensity.



No

The diagram above shows a single column of pixels and the line. By using a vertical edge detection method, no gradient is detected where the drawn line and the column of pixels are aligned.

This is different when in the ROAM mode where detecting edges moving from top to bottom is more useful in determining where an obstacle is. The same logic applies, significant edges are along the perpendicular direction.

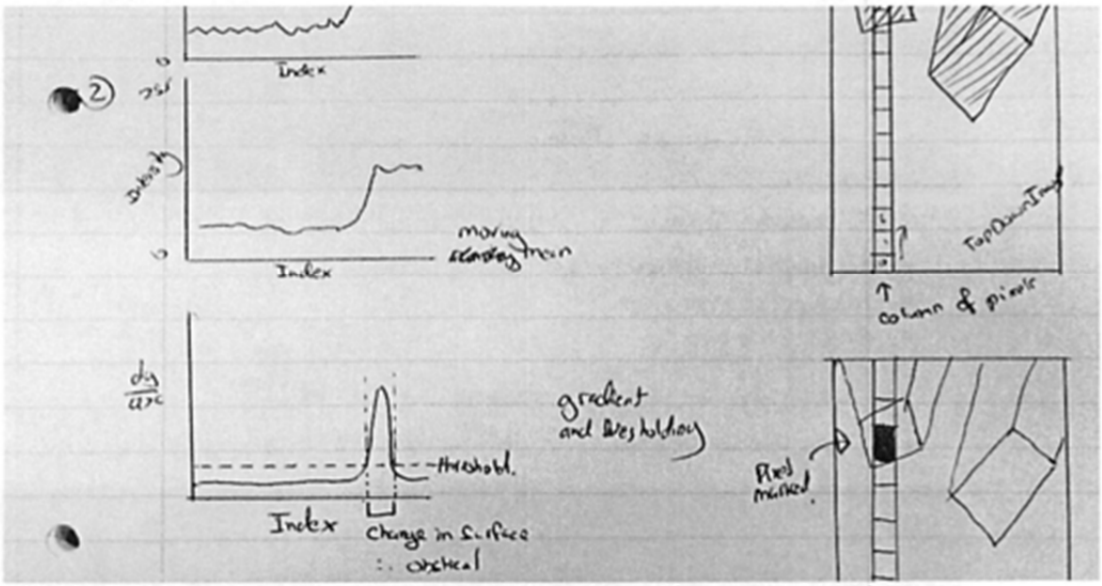
APPROACHES TO EDGE DETECTION:

It is important to consider each row or column individually in the following sections. Each individual row or column can be understood as a single signal with varying intensities. It is the algorithms job to decide when the change in intensity is enough to constitute an edge. This is a simplified process

Gradient-based:

This method takes quite a refined approach to complete the task, overall it can be simplified to calculating the change in intensities and then taking a threshold:

1) Calculate the smoothed intensities of each column of pixels
2) Calculate the change in intensities over each pixel
3) Take a threshold to get a binary image
    a) Potential addition: 'fix' image - joining split edges
        i) Looking at surrounding columns and filling in gaps



The first step in this would be to smooth out the noise and artefacts. After researching there are several smoothing algorithms that would perfectly suitable for this and I wish to test some for performance and suitability for the task. These include methods such gaussian smoothing and moving average, the latter being explained further down.

I am simplifying the problem by focusing on each column individually, this puts the problem down to a signal processing. It can also be simplified by taking the assumption that the surface the car is driving on will be flat and have no contrasting patterns. This method has the disadvantage that each set of data does not consider any other surrounding edges, which means that the algorithm does not fully understand the image and so may result in 'broken edges'.

Gradient Based Edge Detection:

The moving mean is quite a simple thing to implement, however it is important to consider the processing time. Since Numpy's array operations are far faster than separating everything out then performing a function and then combining it back together again, and while the explained algorithms should complete the task; there may be more efficient, but abstract, solutions.

By having additional arrays, similar to the original column of pixels but all the items shifted in index. Then performing operations on the whole array such as subtracting one array from another. But this is something I will experiment with during development

```
Image = 2D array
function MovingMean (Image, width):
       width = int(width)//2
      IF mode is not in 'FOLLOW PATH':
              ArrayToOperate = split the array by every column (Image,number of columns)
      ELSE:
             ArrayToOperate = split the array by every row (Image,number of rows)
      END IF
# Takes the 2D array and separates each column into a separate array so, to call column
of index 3 would be Columns[3]
       EdgeMap = 2D array
       FOR col in ArrayToOperate :
              NewColumn = []
              EdgeMap.append(col[0:width])
              FOR u = width to length of ArrayToOperate :
                     NewOperatorArray.append(average of col[u-width:u+width])
              END FOR
              EdgeMap.append(NewOperatorArray)
              EdgeMap.append(col[len(col)-width-1:len(col)-1])
      END FOR
Return EdgeMap
```

Using the moving average I then need to calculate the change in intensity over each pixel.

- Calculate change before
- calculate change after
- average the two

```
Function Gradient(Array):
     Gradient = []
     Gradient.append(0)
     FOR i = 1 to len(Array)-2:
          gradientlocal= difference between index 'before and current', and 'after and
current' divided by two
          Gradient.append(gradientlocal)
     END FOR
     Gradient.append(Gradient(lenGradient)-1)
     Gradient[0] = Gradient[1]
Return Gradient
```

The pseudo code for the thresholding would be iterating through each index and comparing the item to a constant and saving either a one or a zero whether it is larger or smaller than the constant respectively, this will output a binary image.

Because of the specific nature of the collision map task. I only need to find the first occurrence of an edge moving up from the bottom of the screen, and so it would improve efficiency by looping only until an edge is found and then moving onto the next array rather than finding edges above

(The design continued in this manner.)

Student C

Documented Design

System Summary

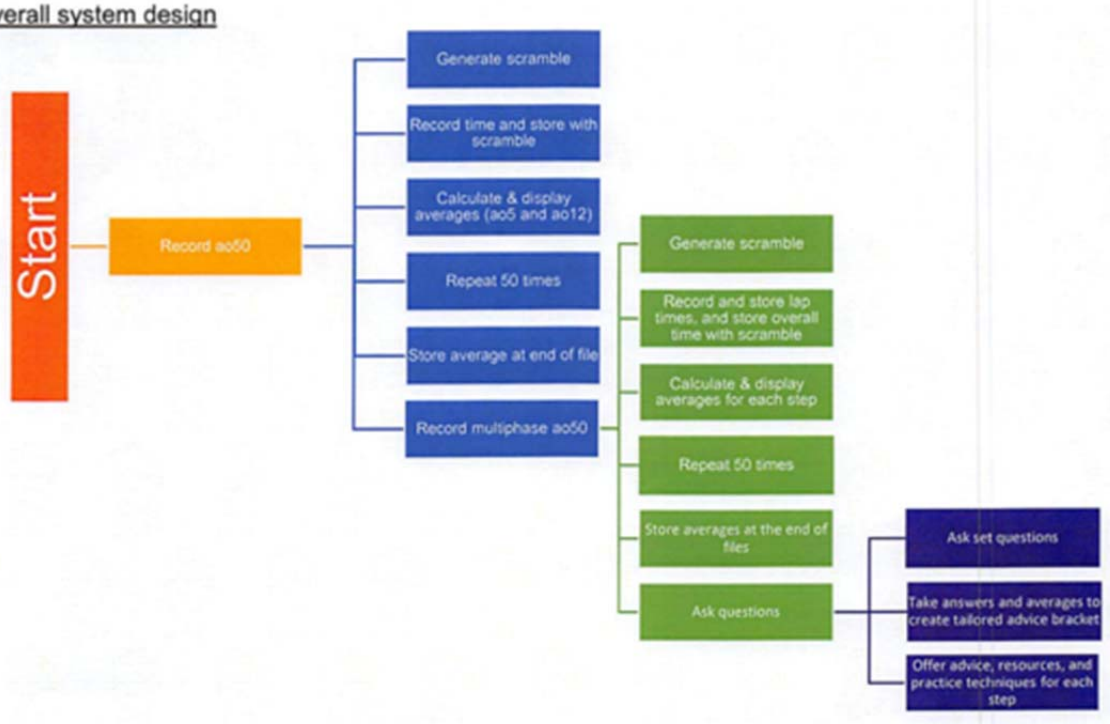
This system is designed to help cubers who are trying to reduce their 3x3 times using the CFOP method and are currently averaging between 10 and 40 seconds. As getting faster at different points requires different techniques, the program will analyse where their issues lie and which techniques they are currently using.

The program will start by taking an average of 50 solves and storing the results in a file. There will then be a multiphase average done, where they stop the timer after each step is completed. Based on the average times for each of the steps versus the data for their overall average, questions will be asked about the method they use e.g. if they average 29 and their cross takes 8s but their F2L only takes 10s, then ask about cross (Do you use x-cross? Can you plan it all in inspection?). Then, the program will offer advice or practice techniques based on their answers and time bracket. Whilst giving scrambles for practicing, solves will not be timed as it encourages the cuber not to worry about how long things are taking them and to focus on getting better and using more efficient practises. Once the user has gotten through a certain number of scrambles for their practice, they will receive another 50 scrambles to do solves without timing them, again encouraging them not to focus on the time. After this, complete an ao50 of timed solves. The program will compare it with the original ao50 to check for improvement, and then let the end user choose to either continue with the program or stop.

The two main components of this system will be:

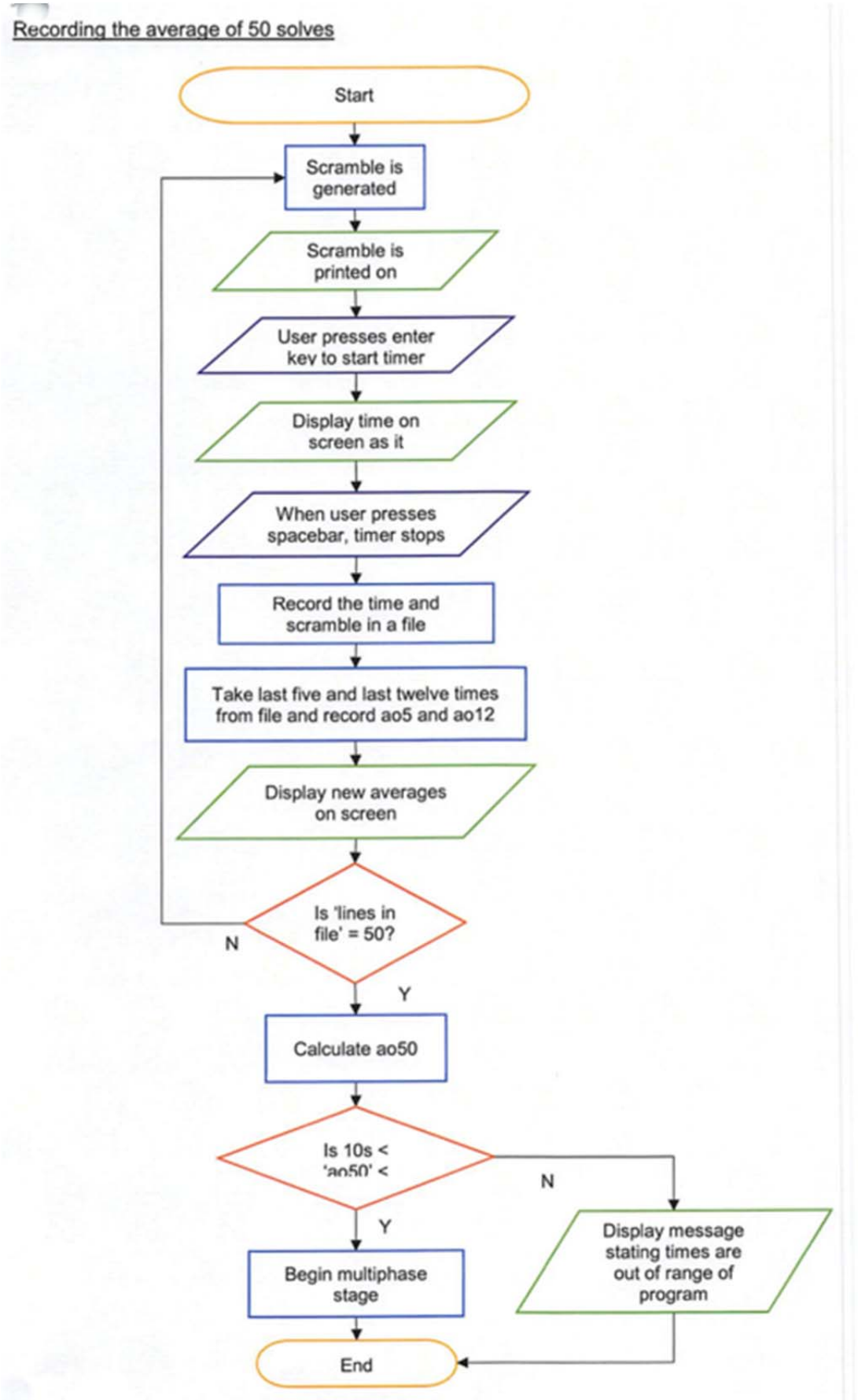
1. The timer and multiphase timer functions – this is the main part of the program, as it is for the ease of the user, rather than having to input all times by hand. By having the timer within the program, it also allows simplicity in terms of the multiphase section, and will eliminate the risk of human error in the input stage.
2. The analysis of question answers with data from the averages – this will be produced based on research of top cubers' averages, methods used, and proven improvement techniques. Combining the analysis of methods along with time will be difficult as quantitative and qualitative variables cannot be compared, but I hope to create overlapping brackets for each section of each component.

Overall system design

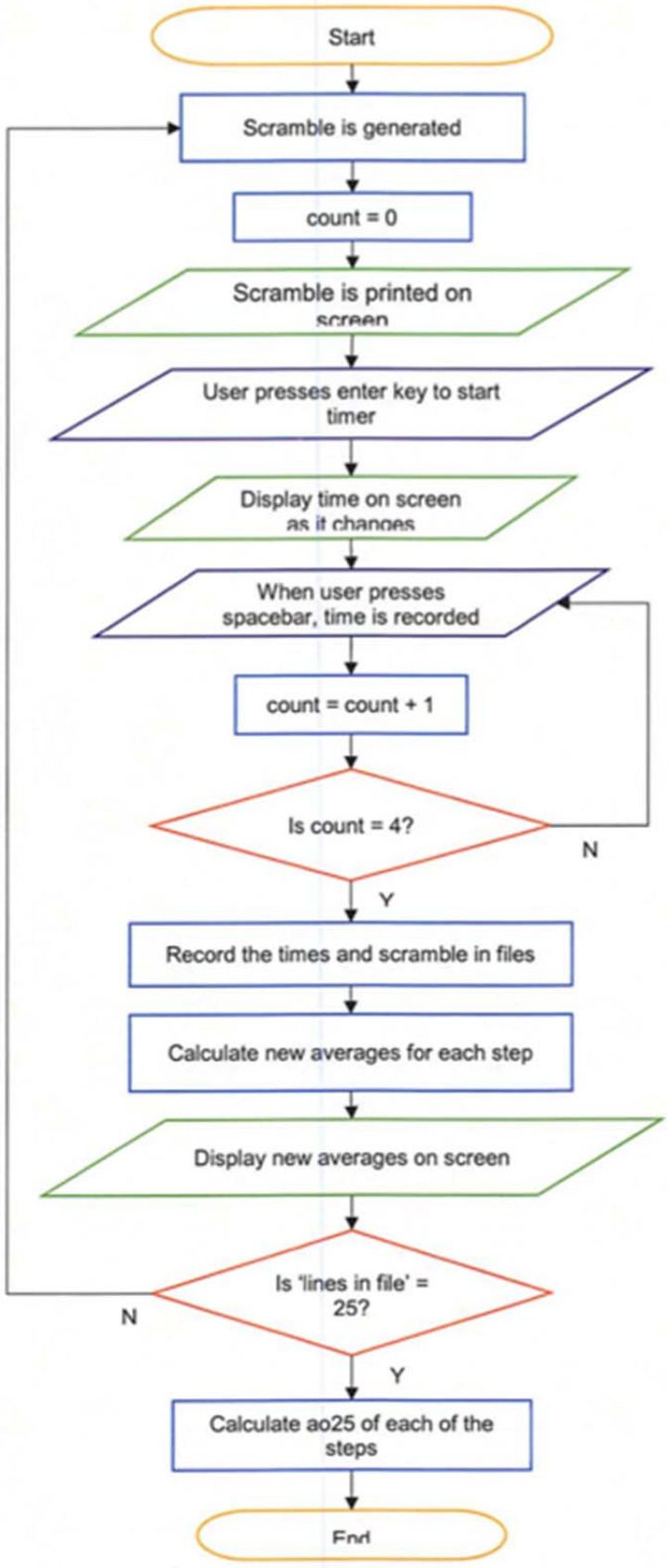


7

Recording the average of 50 solves



Recording the multiphase average of 25 solves



Data dictionary

Data Type	Data items/use	Item names	Sample data
list	Moves for scramble Last 5 times Times to be averaged Times to be averaged Breakdown of solve Data to save to file Lines in file	moves, scramlist self.last5 (of class timer) ao5list, ao12list self.ca/fa/oa/pa (of class multiphase) self.split (of class multiphase) data, multis c/f/o/plines / lines	['RL'] ['-----'] ['19.25']
StringVar	Things to display on the screen As above for multiphase class	self.last(x), self.timestr, self.scram, self.ao5, self.ao12 (of class timer) self.cro/f2l/oll/pll, self.croa/f2la/olla/plla (of class multiphase)	'ao12: 23.78' 'R U B' F2 D U' R' L2 etc'
float	To calculate times To calculate averages	self.start, self.elapsedtime ao5/ao12list[x]	any float
boolean	To check timer's status To check which stage to run	self.running answer	1, 0 True, False
integer	Used to create timestr	minutes, seconds, millis	any integer

Record structures

For 3x3_time_file

This file stores all of the times from the first part of the program.

Record element	Size
Time	7 characters = 7 bytes
Scramble	59 characters = 59 bytes
66 bytes per line	

For multi_time_file

This file stores all of the full times from the multiphase part of the program.

Record element	Size
Time	7 characters = 7 bytes
Scramble	59 characters = 59 bytes
66 bytes per line	

For cross_time_file/f2l_time_file/oll_time_file/pll_time_file

These files store all of the separate multiphase times from the program.

Record element	Size
Time	7 characters = 7 bytes
7 bytes per line	

Main Processing Algorithms

Scramble generator pseudocode

```
FROM random IMPORT all

DEF PROCEDURE scramble()

      moves ← ['RL', 'UD', 'BF']
      scramlist ← []

      FOR i IN (20)
            items ← CHOICE(moves)
            IF SUM([ORD(j) FOR j IN scramlist[-2:]]) IN [158, 153, 136] THEN
                  WHILE scramlist[-1] IN items
                        items ← CHOICE(moves)
                  ENDWHILE
            ENDIF
            item ← CHOICE(items)
            WHILE scramlist[-1:] ← [item]
                  item ← CHOICE(items)
            ENDWHILE
            scramlist ← scramlist + [item]
      ENDFOR

      scram ← (' '.JOIN([k + CHOICE(" '2") FOR k IN scramlist]))

      RETURN scram

ENDDEF
```

Averages pseudocode (for main timer)

```
DEF PROCEDURE average(self)
      ao5list ← [' ', ' ', ' ', ' ', ' ']
      ao12list ← [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
      timeFile ← OPEN('3x3_time_file.txt', 'r')
      lines ← timeFile.READLINES()
      IF LEN(lines) < 6 THEN
            self.ao5.SET('ao5: -----')
      ENDIF
      IF LEN(lines) < 13 THEN
            self.ao12.SET('ao12: -----')
      ENDIF
      IF LEN(lines) >= 6 THEN
            FOR i IN (5)
                  ao5list[i] ← FLOAT(lines[-(i+1)][:5])
            ENDFOR
            ao5list.SORT()
            ao5 ← SUM(ao5list[1:4])/3
            self.ao5.SET('ao5: ' + STR(ROUND(ao5, 2)))
      ENDIF
      IF LEN(lines) >= 13 THEN
            FOR i IN (12)
                  ao12list[i] ← FLOAT(lines[-(i+1)][:5])
            ENDFOR
            ao12list.SORT()
            ao12 = SUM(ao12list[1:11])/10
            self.ao12.SET('ao12: ' + STR(ROUND(ao12, 2)))
      ENDIF

ENDDEF
```

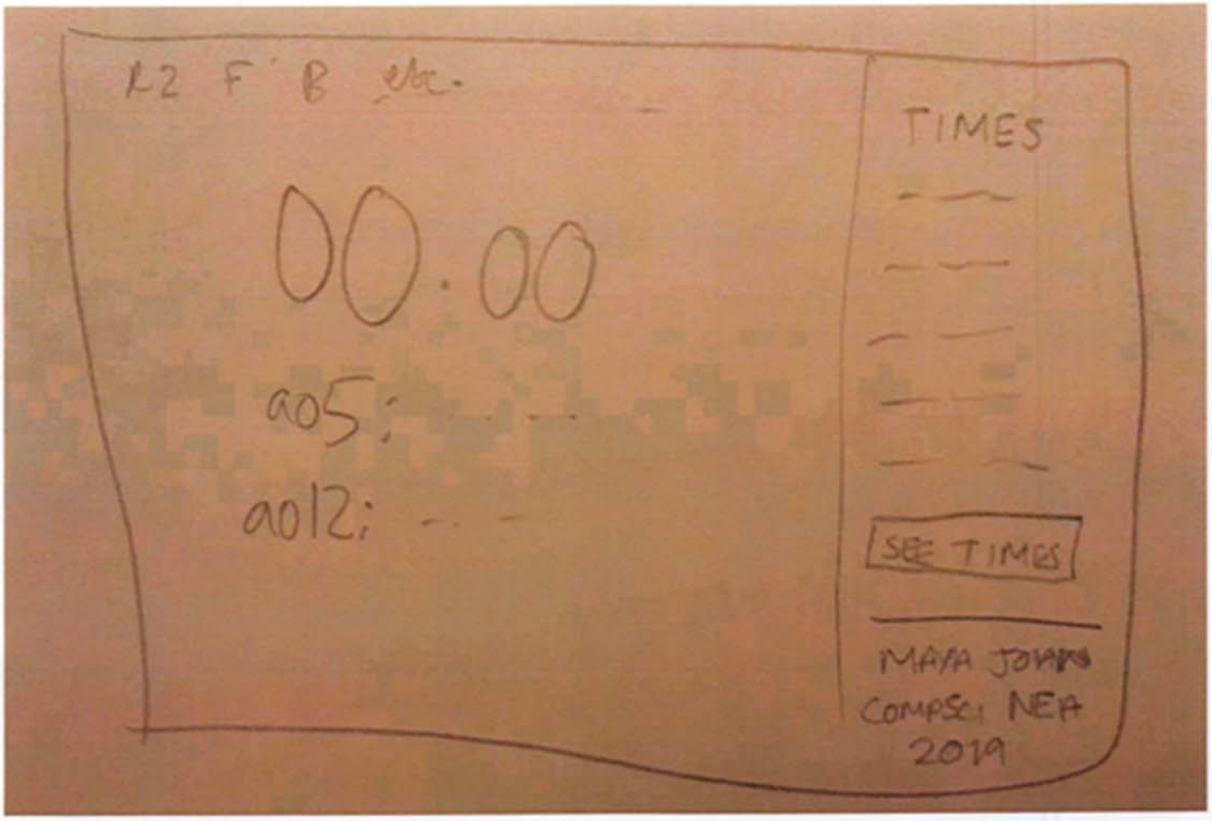
Class Diagrams

timer
- makeWidgets() - update() - settime() - Start() - Stop() - storage() - average() - setzero() + scramble()

multiphase
- makeWidgets() - update() - settime() - phaseset() - Start() - Stop() - storage() - average() - setzero() + scramble()

Interface design

When I first came up with the idea for this project, the first thing I did was design the primary UI for the timer. I will use Tkinter, an inbuilt module in Python, to create a screen that looks like this. I also plane to use a monospaced font, so that the time doesn't jump around on the screen as the width of the numbers changes.



Testing

Level	Criteria	Mark
4	Clear evidence, in the form of carefully selected representative samples, that thorough testing has been carried out. This demonstrates the robustness of the complete or nearly complete solution/thoroughness of investigation and that the requirements of the solution/investigation have been achieved.	7–8
3	Extensive testing has been carried out, but the evidence presented in the form of representative samples does not make clear that all of the core requirements of the solution/investigation have been achieved. This may be due to some key aspects not being tested or because the evidence is not always presented clearly.	5–6
2	Evidence in the form of representative samples of moderately extensive testing, but falling short of demonstrating that the requirements of the solution/investigation have been achieved and the solution is robust/investigation thorough. The evidence presented is explained.	3–4
1	A small number of tests have been carried out, which demonstrate that some parts of the solution work/some outcomes of the investigation are achieved. The evidence presented may not be entirely clear.	1–2
	No evidence presented	0

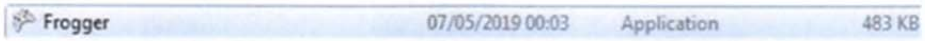
Student A

4. Testing

Objective No	Description	Data/Action	Expected Result	Actual Result	Evidence
1	Running .exe file	Double clicking on the executable	Game opens	Game opens	1
2	Loading splash Screen	Game Loading	The game loads the splash screen	Loads splash screen	2
3	Pressing buttons except 's'	Key input	Does nothing	Did nothing	3
4	Clicking everywhere on the splash screen	Mouse input	Does nothing	Did nothing	4
5	Pressing 's'	Key input	Game loads	Game loaded	5
6	Pressing 'enter'	Key input	Game loads	Game loaded	6
7	Pressing 'moving keys' WASD or arrows.	Key input	Frog moves	Frog moves	7
8	Frog cannot move out of boundaries of the game.	Key input	Frog will stay within boundary	Frog moved out of game boundaries	8
9	Frog gets hit by car	Game code	Frog will lose a life and reset to original position	Did as expected	9
10	Life -1	Game code	Life goes down by 1	Lifes went from 3 to 2	10
11	Score goes up when player goes up in Y cord	Game code	Scores goes up	Did as expected	11
12	If timer goes down to 0, player loses a life	Game code	Player loses life	Lost a life	12
13	If times goes to 0 and player is on last life gameOver	Game code	Game over plays	Game over	13

Test Evidence

1.

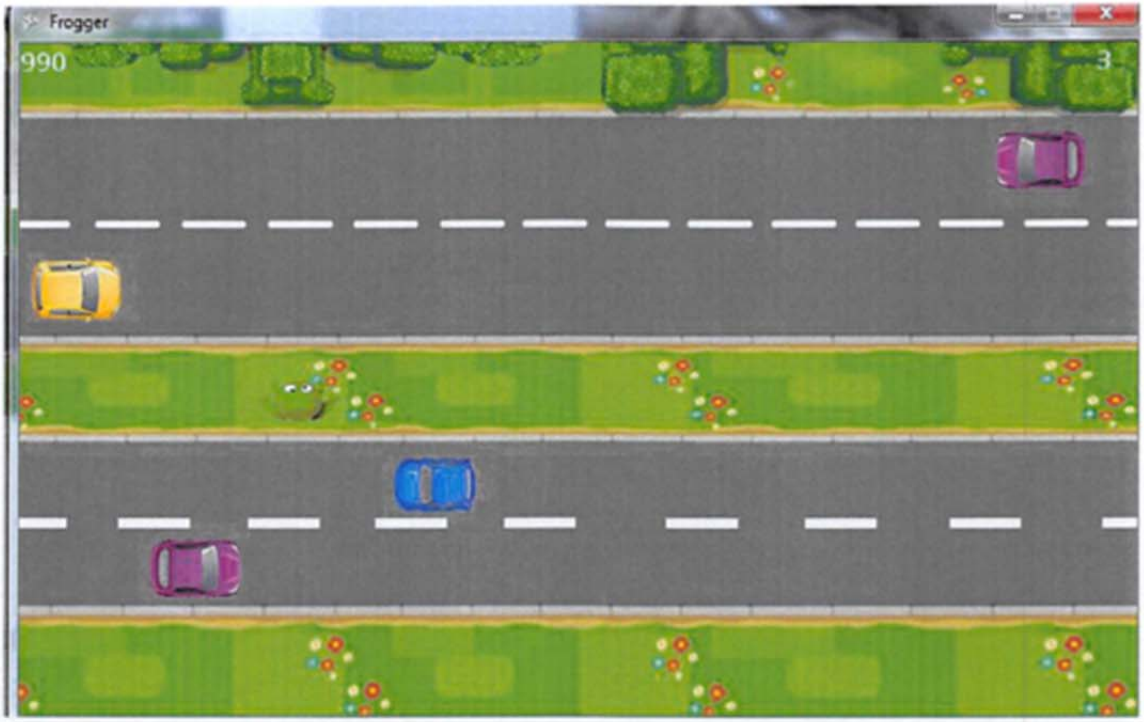


2.

3 + 4.

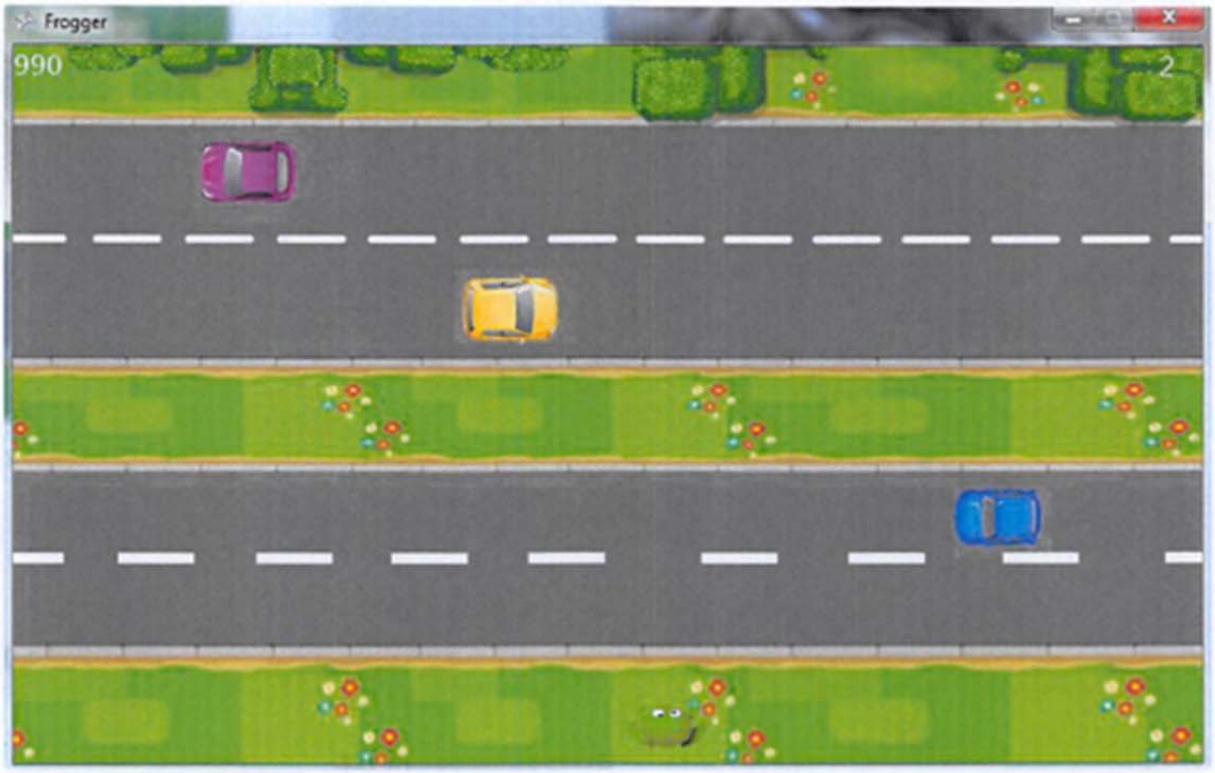
5 and 6.

7.



8.

9.



10.

11.

12.

13.

Student B

4.Testing

Test number	Test description	Expected outcome	Actual Outcome	Success or failure
1.	The online function	A game running on one computer will connect another computer system on the network with the same program running. The two programs should have updated board when one of them places a counter.	Both PC's successfully connected and played a game of join-4 over the network.	Success.
2.	Board size	The program should be able to handle a wide range of different inputted widths and lengths. 4 tests will be conducted: a board size of 3x3, 4x4, 20x20 and 7x6 should all be tested. The 3x3 grid should be rejected for being too small. The 20x20 grid should be rejected for being too large. 7x6 and 4x4 should both be allowed to run, open and be played on.	All 4 tests were successful, the 20x20 and 3x3 grids were rejected while the 7x6 and 4x4 grid ran successfully and loaded the board correctly.	Success
3.	Check victory	Trying it out on a grid of 4x4 and 7x6, the game should be able to display a victory message for both the red counters winning and the blue counters winning.	A victory message for both red and blue counters were displayed when the game found a 4 in a row.	Success
4.	AI	The AI should be able to block my moves on the board while working towards winning by itself.	Success, the AI successfully blocks my moves while managing to work towards winning itself.	Success

Evidence of testing

1. Here's a YouTube link that shows the test being carried out using my laptop and PC: **https://www.youtube.com/watch?v=0MAO-Nbs_GQ&feature=youtu.be**
2. Here's a YouTube link that shows my board sizes being tested: **https://youtu.be/ilF9R8dYjGk**
3. Here's a YouTube that shows the check victory being tested: **https://youtu.be/ChBa7oyUnw8**
4. Here's 2 YouTube links showing how the AI works and wins:
   Link 1: **https://www.youtube.com/watch?v=r8l3lEym5uc**
   Link 2: **https://youtu.be/yb3XdlL9AO4**

Student C

Testing

In order to test how effectively my program has met all the objectives I outlined in my analysis, as well as how robustly coded it is and how successful a solution it provides to the problem I initially proposed to solve, I have conducted a number of intricate tests to test the various functionality of the application. These are shown in the table below. In aspects of the application that can be tested with different types of input data, namely the move-making procedures, I have included tests conducted with normal data that should be considered valid by the system, as well as tests conducted with erroneous data that should be considered invalid by the system; the inputting of boundary data cannot be particularly applied to my implementation of a chess engine.

It is important to note that tests 1-13 are performed in two-player mode, and all the AI tests apart from difficulty changing will be done in the default AI difficulty - medium. In certain tests, due to the volume and longevity of it, I have split them into sub-tests in order to provide a more intuitive hierarchy of the tests.

I have compiled all these tests into a single video which I have uploaded on Youtube to view at the link: https://www.youtube.com/watch?v=eIb6yK5bVZs&t=8s

Test No.	Test	Test Procedure	Expected Result	Objective(s) Tested	Actual Result Timestamp
1	Loading application's GUI window	The application will be started, where the GUI window through which the	A GUI window containing an 8x8 chess board with all the piece	1a, b, d	0:00

		chess games are played through and all the other functionality is accessed through is loaded	sprites in their initial positions will be displayed, along with a button for changing the game mode, 3 buttons for changing the difficulty to easy, medium or hard, a button for recommending a move and a text space for the move to be displayed, an undo button, and a text area to display the evaluation values of the board state after white's last move and black's last move		
2	Moving pieces on the GUI via drag-and-drop	Some pieces will be selected and moved a couple of times to valid squares via drag-and-drop, where the user clicks and holds onto a piece, drags it to a square to move it to, and then drops it in the square	The user should be able to move a piece to a valid square via drag and drop	1c	0:12
3	Piece movement validation	8 subtests will be conducted for the test for each piece, detailed below: a. The queen pieces of both sides will be moved to valid squares that are in each line of movement a queen piece can move in - vertically upwards and downwards, horizontally left and right, diagonally upwards to the right and left, and diagonally downwards to the right and left b. The rook pieces of both sides will be moved to valid squares that are in each line of movement it can do - vertically upwards and downwards and horizontally left and right c. The bishop pieces of both sides will	The moves should be considered valid by the system and the user should be able to perform the moves, where the board array and GUI GridPane should be updated to reflect the new board state	2ai, ii, iii, iv, v, vi	0:24

		be moved to valid squares in each line of movement it can do - all four diagonals d. The knight pieces of both sides will be moved to valid squares that form an L shape with the original square e. The king pieces of both sides will be moved to all the adjacent squares next to the king that are valid f. The pawn pieces of both sides will be moved from their starting positions to squares two spaces in front of them, and then once the pawn pieces are not in their starting positions, to squares one space in front of them			
4	Erroneous piece movement validation	A piece of each piece type from both sides will be moved to some squares that do not count as a valid move for that piece type - to do this, subtests will be performed for each piece type: a) The queen, rook and bishop pieces will be moved to squares that are not within its line of movements, and attempt to jump over pieces b) The knight piece will be moved to pieces that squares that are not in an L shape with its original square c) The king piece will be moved to squares that are not adjacent to it and hence invalid squares to move to d) The pawn piece will be moved to	The moves should be considered invalid by the system and the user should be unable to perform those moves	2ai, ii, iii, iv, v, vi	1:55

(The testing continues in this manner.)

13	Heuristic evaluation function	Moves will be made on both sides and the GUI will be shown to display text after each move that shows the evaluation of the board state after each move	The GUI should display the evaluation values next to the appropriate text displays: Evaluation after last move by White, and Evaluation after last move by Black, where the appropriate evlauation value of the board state after the corresponding side's last move will be displayed next to the appropriate text displays	7b	6:35
14	Game mode changing	The game mode button will be pressed before a game to change the game mode from two-player mode to AI mode and vice versa, and it will be shown under each game mode that the appropriate game mode ensues - ie an AI plays or another local player is required to make moves of the opposing side. It will also be pressed after a game has started	Upon pressing the button before starting the game the game mode should change, where in AI mode an AI wil play and in two-player mode two people can play locally. Pressing the button should change the text on the button to reflect the current game mode - however, pressing it should do nothing after a game has started.	5a, b	7:21
15	Playing against AI	A sample of playing against the AI will be shown, where the AI will respond to moves played by the user by calculating and playing optimal moves	Whenever the user plays a move in AI mode, the AI should respond and calculate an optimal move which will be played	6a	7:56
16	AI piece capturing	A sample of playing will be shown where the AI performs capturing of the user's pieces	If the AI deems a capturing move in the given board state to be an optimal move, it will perform the move and capture the piece	6b	8:16
17	Show the AI putting the user in check	A sample of playing will be shown where the AI puts the user in check, where the user will then be unable to perform moves that do not remove the king from check	If the AI deems a move that puts the player's king in check to be an optimal move, it will perform the move and put the king in check.	6c	8:50
18	Show the AI putting the user in checkmate	A sample of playing will be shown where the AI puts the user in checkmate and hence wins	Whenever the AI is able to put the user in checkmate, it will do so and the appropriate text will be displayed on the GUI	6d	9:05
19	Show different	The GUI buttons will be	The user should be able	6e	9:54

	difficulty levels of AI - Easy, Medium and Hard	used to play against the AI in the three difficulties, where the AI plays to a level according to the difficulty. It will be shown that the difficulty can be changed at any stage in the game	to play against the AI in the three difficulties, where the AI plays to a level according to the difficulty, where the difficulty should be able to be changed at any stage of the game		
20	Move recommending	The GUI button will be used to generate a recommended move for the user to play which will be displayed on the GUI, in AI mode for the user's side, white, and in two-player mode for both sides	Upon pressing the button a relatively optimal move will be calculated for that board state and displayed on the GUI	7a	12:16
21	Undoing moves	The GUI button will be used to undo moves, where the test will be conducted in AI mode to show the user being able to undo multiple moves at any one time which can then be redone, where both the AI's and user's move are undone, as well as in two-player mode, where pressing the button undoes each individual player's moves, which can then be redone	Upon pressing the GUI button the move played should be undone with the previous board state being displayed on the GUI, which can be done multiple times	8	14:30
22	Show minimax algorithm operation	A debug console will be open, and moves will be performed in AI mode where whenever the AI plays a move, all the evaluation values getting backtracked through the tree from the leaf nodes to find the optimal move will get printed in the debug console, where each value will be accompanied with a string showing if the value is being backtracked to a maximising or minimising node	The evaluation values being backtracked should be printed to the debug console whenever a move is made	6a, b, c, d	15:27
23	Show alpha-beta pruning optimisation operation	Every time the alpha and beta values are changed in a certain recursion instance of either of the functions when the AI makes a move, the new value will be printed to a debug console. Alpha and beta represent the best possible evaluation values the maximiser and	Every time the alpha and beta values are changed in the instances of either function when the AI makes a move they should be printed to the debug console, with a string accompanying if the value belongs to alpha or beta	6a, b, c, d	15:48

		minimiser respectively can currently get in that stage of the tree or above, and are changed in the nodes when an even better value for the maximiser/minimiser is backtracked.			

The video should ideally be watched in full-screen to be able to fully view the GUI and its text displays in a clear format. The timestamp of each individual test is included in the description of the video, as well as in the table above.

Additionally, I have summarised the results of each test along with screenshots in the table below.

Test Number	Screenshot	Observed Result
1		The GUI Window loaded with all the GUI features and buttons expected to load up.
2		The drag-and-drop operation worked as expected, with the user able to click, hold and drag a piece to a square to move it to, and then release to move the piece there.

3a		The system allowed the queen piece to be moved to all the squares that constitute as valid moves for that piece type.
3b		The system allowed the rook piece to be moved to all the squares that constitute as valid moves for that piece type.
3c		The system allowed the bishop piece to be moved to all the squares that constitute as valid moves for that piece type.

3d		The system allowed the knight piece to be moved to all the squares that constitute as valid moves for that piece type.
3e		The system allowed the king piece to be moved to all the squares that constitute as valid moves for that piece type.
3f		The system allowed the pawn piece to be moved to all the squares that constitute as valid moves for that piece type.

21		The undo button worked, which allowed the user to undo moves they have played any number of times that was desired, and worked in AI mode where only the user's move were undone, and in two-player mode where the moves performed by either side were individually undone
22		The minimax algorithm operated as expected, with all the values being backtracked through the maximising and minimising nodes of the tree being printed correctly to the debug console
23		The alpha-beta pruning optimisation operated as expected, with every time an alpha or beta value was changed in a certain recursion instance of one of the functions, the new value was correctly printed to the debug console

Overall the tests showed the program to be correctly working as expected, with the actual outcomes of each test matching the expected results, and thus showing my program to successfully meet all the objectives I had outlined, in addition to being able to deal with different input data types and generally perform in a robust manner.

Evaluation

Level	Criteria	Mark
4	Full consideration given to how well the outcome meets all of its requirements. How the outcome could be improved if the problem was revisited is discussed and given detailed consideration. Independent feedback obtained of a useful and realistic nature, evaluated and discussed in a meaningful way.	4
3	Full or nearly full consideration given to how well the outcome meets all of its requirements. How the outcome could be improved if the problem was revisited is discussed but consideration given is limited. Independent feedback obtained of a useful and realistic nature but is not evaluated and discussed in a meaningful way, if at all.	3
2	The outcome is discussed but not all aspects are fully addressed either by omission or because some of the requirements have not been met and those requirements not met have been ignored in the evaluation. No independent feedback obtained or if obtained is not sufficiently useful or realistic to be evaluated in a meaningfully way even if attempted.	2
1	Some of the outcomes are assessed but only in a superficial way. No independent feedback obtained or if obtained is so basic as to be not worthy of evaluation.	1
	No evidence presented	0

Student A

5. Evaluation

Objective:	Has the objective been met?
1. Inform the player of how to play the game in the first screen.	Yes, this objective has been met as in the first screen there is sufficient information for the player to continue
2. Have working solution.	Yes, this objective has been met as the game build completely and it works without crashing.
2.1: The score section works according to the player movements and the further the player moves the more score he/she earns.	Yes, this objective has been met as in the game the user earns more and he/she moves up.
2.2: If the player hits a car he/she loses a life and therefore the game resets keeping the score since he/she will have 3 lives.	Yes, this objective has been met as the position of the frog resets and the lives get deducted by one.
2.3: Have a working timer that makes the player lose a life if times goes down to 0, this encourages faster thinking and competitiveness amongst players.	Yes, this objective has been met as the timer is working correctly and as shown if the timer does go down to 0 the player loses a life.
2.4: : If the player loses all lives then a prompt should come up allowing them to press a button to continue, this button will reset the game.	Yes, this objective has been met as such event takes place.
3.Allow the player to move inside the game, using directions buttons on the keyboard indicating the direction. This will require a set speed and delay between each movement to stop the player from zooming across the entire map.	Yes, this objective has been met as such event takes place. There is a 7 seconds delay timer which restricts the players movements.
4. Have multiple levels which in this case indicate the level of difficulty of the times tables. This will test the players to see who can get to the hardest level the fastest and how many lives he/she will have left.	No, this objective has not been met because of the amount of time I had. This objective would have created the game much more interesting and enjoyable.
5. Have a high score table with names. (An extra)	No, this objective has not been met because of the amount of time I had.

6.Every X amount of SCORE the player would be asked a times tables question, this is how the game will have an educational factor.	No, this objective has not been met because of the amount of time I had. This is the main objective of the game therefore the game currently doesn't have a full purpose.

Feedback

I asked the previous two ICT students to fill out a questionnaire at the start of the process to give me their feedback on their experience with the program and anything that they think could be done to improve the program.

Student1:

This student said that he thought the program would be very useful for new children approaching the topic of times tables, he also mentioned that the idea of being asked questions as you move through the game was a good idea. He also liked how the game looks, nice and clean. However, he said that the program needs to be finished and slightly extended before it can be used for the children. He said that the idea of having a high score board would really add to the competitiveness of the frogger idea as the amount of time was limited.

Student2:

This student said similar things to the first one in terms that he found the game interesting, and the intention and objective was educational therefore he was really interested. He had a strong standing on how undeveloped the game is and how premature its state is. To add additional feedback, he mentioned how to improve the game by changing the map layout and it could get repetitive if kept in the same format

Analysis of feedback:

The overall feedback was unison, the game isn't ready for use, its premature and undeveloped. From the feedback I got that my draft Is good looking, the idea of questions popping up was good and intentions behind the draft were there for them to see. To improve further they suggested that I change the layout of the map in different levels and have the high score board fully implemented.

From my personal thoughts I believe that I have made a good attempt at the solution, its not of any use at its current state as there were time restrictions which meant I couldn't implement the additional features as I wanted. However, if made to fully work this game would be really helpful for children to learn times tables as its engaging, colourful and has a timer which makes the user think fast while answering questions. In my eyes this game would be far more interesting and beneficial to the child that is learning times tables from pen and paper.

Possible extensions:

As outlined in the feedback an extension to the program would be the high scores table, this feature could help the teacher track who is the fastest at learn their time tables and therefore could reward the fastest one. If this game was to be used daily inside a classroom then the high score would probably be one of the most useful extensions.

Another extension is the map layout, in the feedback the map layout would have gotten 'repetitive'. To counter this the solution would have to have more than 1 maps that is randomly called at the start meaning that there would be more than just 1 map to play from which would enhance the experience from playing this game.

Conclusion of project:

It is clear from the feedback that the idea and intentions were good however the project was unfinished and premature, the feedback I received contained positive thoughts however there was room for improvement.

Student B

Evaluation

Effectiveness of Result

Overall, I believe the final result of my project to be highly effective. It performs the task it was designed for perfectly within the scope it is defined. While there is area for improvement, new features can easily be added to future versions due to the modular nature of Windows Forms. In addition to this, the trackedVariables class can be reused to store any data that should need to be passed around the system for a future feature.

Comparing the Result to the Objectives

- Receive user input and arrange it in a logical format for payroll calculations.

My final project is able to do this via the 5 box, 10 line interface. The automatic summations are carried out in a logical direction; right then down. As a result, it is clear to any user how my program operates.

- Save given inputs to a database, which it can then load and edit as required.

My final project is able to save and load with no issue. It can edit a loaded file and save it again, including the ability to overwrite an existing file. This fits my objective perfectly, and as I wanted it to.

- Take a variety of data inputs, and have the ability to be moulded to the user's requirements to a fair degree (E.g. within what is possible in this scope).

The user is able to input incomes and deductions of up to £9,999,999,999.99, and more if accounting for the multiple lines of input the system can take. These multiple lines can be added and removed such that there are any number of lines between 1 and 10 on the paysheet at any time.

- Calculate some of the numbers for the user as they are input.

This is not exactly how the program functions. Granted, the AutoSum calculates the sum of all given values on the paysheet when enter is pushed, but this isn't technically the original objective. In this case, however, I believe this is better, as it is less of an inconvenience to the user. If values were summed constantly, it may disturb the inputting process of a user, and cause more issue than it is worth.

- Run with a minimum of exceptions to be handled - use exception handling primarily for unavoidable errors.

It is hard to say whether this objective has been achieved without attempting to rewrite parts of my code without relying on exceptions to break out of functions. Given my final project only uses exceptions to break out of loading and saving, I'd say this objective is within reach.

- Perform the above while remaining easy to use – This can be achieved by adding support functionality.

I have added both a user manual and a support contact option to my project; this means that even if the user does not immediately understand how the program works or has additional requests for its functionality, they can use the support functions to gain this understanding and prompt the developer to fix a potential issue. As a result, my project meets this objective.

Improvements in Future

If I were to improve this program in future, I would like to add a method of exporting the data to a .csv or other format. This would massively increase the utility of my project, as the contents could be moved between programs should the need arise (E.g. Transfer of files between offices/companies/etc.). In light of this new aspiration, changing the window heading to match the name of the current file would also be a useful addition. In fact, a file window navigation for saving and loading could then become an important feature.

Student C

Evaluation

Achievement of Objectives

In order to critically evaluate how successful my coded solution was, I have analysed each objective I initially set out to meet in my analysis, and how successfully and to what extent I managed to meet each one. Under certain objectives I have also listed potential improvements that can be done - however, as each objective has been successfully met, these improvements would of course go beyond the requirements initially outlined, and/or provide a possibly more effective implementation of a feature to meet a certain objective.

Objective 1 - Aesthetically Pleasing and Intuitive GUI Design

a). Fully displayed GUI chess board

I have managed to achieve this objective through the usage of the FXML framework which enabled me to represent the chess board as a GridPane GUI element, where I made it have 8x8 cells to represent the 8x8 grid held in a chess board, and added rectangle objects of alternating black and white colours to represent the squares of the board, which were placed in the appropriate cells of the GridPane. I managed to make the chess board fit the "aesthetically pleasing" quota by having all the cells be of the same size, and consequently have all the rectangle objects be the same size to produce an even chess board that maintains the square ratio, with the width and length being equal. Furthermore, I have also annotated the board with row and column values of each square on the side to make it easier for the user to reference a square with its row and column value, going beyond the requirements. This can all be seen in the testing.

b). Sprites for each chess piece

This objective was successfully achieved, as shown in testing, by representing the piece sprites with ImageView objects, in which I used open-source chess piece sprites in order to make them "aesthetically pleasing", as otherwise I would have had to draw them using some software which woul have likely not produced as amicable a result. The pieces are placed in their initial positions at the start of a chess game as intended, as the board state at the start simply reflects the default value of the board array, which is the starting board state of a chess game.

c). Drag and drop movement

This objective was successfully achieved as shown in testing by coding the appropriate event handlers where the user can click on a piece they wish to move, hold down on the mouse button and drag the piece to the square they wish to move it to, and then release the mouse button to drop the piece there, moving the piece to that square provided it is a valid move. This enables the program to be user-friendly and intuitive, although to potentially make it even more intuitive a further extension could be implemented that highlights on the board the squares that a certain piece can move to upon clicking the piece, allowing the user to immediately see the valid squares of movement and reduce the likelihood of them making an invalid move and having to try again; although arguably not showing the moves they can prompts the user to learn more quickly whether certain moves are valid or not in a certain board state without having to rely on visual feedback, and thus whether or not to implement such an extension would be up to preference.

d). Buttons for various engine functionality

As shown in testing all the buttons for the additional functionality of the program are present in the positions I had planned them out to be in in the design section, and are fully operational, thus meeting the objective. Using FXML I have been able to position the buttons in the intuitive manner I had them planned them to be, where they are spaced

apart and positioned next to any text displays which output results from the button, and thus meet the quota of "aesthetically pleasing".

Objective 2 - Piece Movement and Validation

a) . Each piece type can only be moved by the player to a valid square for that board state

This objective has been successfully met as shown in testing, where using the drag-and-drop operation the user is able to drag the pieces of the different piece types to squares of their choosing, where the move is then only completed if it is completely valid in that board state, where this works for each piece type including the Queen, Rook, Bishop, Knight, Pawn and King. If a move is invalid, the move does not get completed and the user must retry another move.

b). Piece capturing

This objective has been successfuly met as shown in testing where again using the drag-and-drop operation the user can move pieces to squares that contain pieces of the opposing side, and then provided the move and capture is valid, the piece will be moved to that square and the piece previously in that square is removed from the board. If the move/capture is invalid, the capture move does not go through and the user must retry another move. Relating back to the improvement on the drag-and-drop, an additional feature could potentially be implemented where valid capture moves could be highlighted on the board using a different colour to alert the user to a possible capture move.

c). Castling

As shown in testing this objective has been successfully met where the user is able to perform a castling move provided all the requirements for the certain castle are met, and if the requirements are not fully met the user is unable to castle. Both the user and AI are able to perform castling moves.

d). Turn alternating

This works as intended as shown in testing, where after a certain side makes a move that side is prevented from making another move, until the other side has made a move. This works in both two-player mode and AI mode, although an improvement that is not directly related to the mechanism could be to indicate on the GUI which side is to currently play, to make it more obvious to the user.

Objective 3 - Check Function

This objective was fully achieved as shown in testing where whenever a side is put into check, that side is prevented from making any move unless that move removed the side's king from check. It works in both two-player mode and AI mode, where in AI mode it is also applied to the AI such that the AI must always make a move that removes it from check if it

(The Objectives continue in this manner.)

This objective was successfully achieved as shown in testing, where after each move is made the evaluation of the board state after the move is displayed on the GUI next to the appropriate text dependent on the side that made the move, where the value is either displayed next to "Evaluation after last move by White" or "Evaluation after last move by Black", and works in both AI mode and two-player mode as required. The evaluation value is negative to indicate an advantage to black, and positive to indicate an advantage to white, where the lower the value the higher the advantage of black, and the higher the value the higher the advantage of white. Furthermore, going beyond the requirements, it also displays which side is currently at an advantage in the current board state. The heuristic evaluation function itself internally works by sumating the piece weight values across the board, the mobility values of all the pieces across the board, and taking into account a bonus if a side has castled or not, where the value for each individual piece is multiplied by a side multiplier of either 1 or -1 dependent on whether it is white or black.

A potential further improvement to the heuristic evaluation function itself could be to add more factors for the evaluation, such as positional advantages for certain pieces, where there are certain squares that are more optimal for certain piece types to be located in - i.e. knight pieces are more of a threat when positioned in central squares as opposed to the sides.

Objective 8 - Move Undoing

This objective was successfully achieved, where, as shown in testing, the user can undo moves any number of times by pressing the GUI button, where the user can then reconsider and redo the moves made. In AI mode the undo button undoes both the AI's move and the user's move to allow the user to redo their move, and not the AI which would be pointless, and in two-player mode the undo button undoes each individual move made by both sides, as required. This internally done through the use of a stack that stores board states that have occurred throughout the game.

User Feedback

In order to gain third-party perspectives of how effective my coded solution was, I sent my program to Denis Ianev, my original client, and requested his feedback on how well it suits his needs and meets the requirements that he initially requested for, shown on the next page.

Dear ,

Thank you for sending me a copy of your chess engine program - I've managed to use it extensively over the past few days, and I'm very glad to say it really has surpassed my expectations. I've been quite surprised with the level at which the AI plays - having initially started on medium level, due to it being quitechallenging I quickly lowered it down to easy level, where I have managed to enjoy good quality games against the AI which have been very educational - through repetitive play I have been able to start identifying weaknesses in my strategies and choices of movements, and have furthermore been able to learn quite effectively by simply observing the moves and choices made by the AI itself. After a while I was able to move on to the medium difficulty, which I am still currently playing against, where the AI plays to a very good level, and the difficulty variability allows for a constantly effective learning curve, where I do not stay satisfied with being able to play against the AI at one difficulty but rather constantly endeavouring to move on to the next difficulty. What really impressed me however was the additional functionality built in to the engine, some of which I suggested would be good to implement, some of which I didn't. I've found the recommending move feature to be extremely useful, whenever I come up with a move which I think is good, it's very handy to then be able to compare this selection with the move suggested by the engine, and furthermore, as you have informed me the recommended move is purposefully only relatively optimal, it's very beneficial for me to attempt to critique even the recommended move, and try and see if I can come up with an even better move. The board state evaluation feature has also proven to be significantly useful, where I can see how effective my move was after it's made, and it's also helpful to even see the board evaluation after the AI's move, as I learn a lot from just observing the AI move making decisions. This feature coupled with the move undoing feature is superb, as whenever I make a move which the engine tells me is not very effective, I am able to undo it and try another move which I think would be even better. The ability to change to two-player mode is also particularly convenient in the event that chess club is not on and I do not have access to a chess board, yet I would be interested in engaging in a game against another person as opposed to the AI, where I stil have access to the features such as board state evaluation.

Despite the application being extremely good, there are some aspects which I think could be further improved upon. Adding an even wider range of difficulties could be quite useful, with difficulty levels even easier than the current "Easy" mode for players even less experienced, and more difficulty levels to add more nuanced categories between the other two existing modes, as well as levels even harder than "Hard" for players who are quite experienced but still want to improve. Also, I thought that the recommend move feature could display the move by highlighting on the board the piece that should be moved and where it should move to as opposed to just writing it as text, which would be more intuitive; and furthermore it could also be quite useful if squares that constitute as valid moves are highlighted on the board to make it more obvious for the user.

Thank you very much,

The feedback submitted by overall proved to be highly positive, with him praising the various features I implemented in my final program in terms of their ability to improve his performance in chess, as well as the raw capability of the AI's chess playing. It can notably be seen that he found the combination of the board evaluation feedback, move undoing and move recommending to work together in a largely effective manner to improve one's playstyle and decision making process, and he also seemed to be appreciative of the decision I took to make the recommended moves not very optimal - the moves are only calculated with a depth of 2 of the minimax algorithm. He did also submit some areas for potential improvement, which seem to be similar to the areas I myself identified through

the analysis of how well I had achieved each objective, with a focus on increasing difficulty adjustability.

To gain another perspective, I also submitted my program to another relatively new member of the club who I had surveyed earlier, Himshun Yu, who also provided some feedback for me to analyse:

Re: Chess Engine Feedback

01:19

To:

Dear ,

Thanks for the chess engine program you sent me. I've used it for a bit now and I'm actually quite impressed by it - playing against the AI at the different difficulty levels has helped me being to notice different kinds of patterns and strategies to play that I hadn't realised before, and I'm glad that the different difficulties have a meaningful difference - I find the "Hard" difficulty too difficult to play against most of the time! Upon improving however, I'm certain the harder difficulty will be crucial in making me become even better. I find it really helpful you included feedback on the moves I make, so I can learn to better recognise what moves are effective and what aren't, and I really like the move recommending feature too.

I think to improve it you could possibly highlight squares on the board that I'm allowed to move a piece to, to make it a bit more obvious? Also I think it'd be better if the program alerts you when you are in check, as a lot of the time I don't realise that. Other than that it's a really good program.

Best Wishes,

Get Outlook for Android

From:

This feedback also seemed to be largely positive, with the raw capability of the AI being praised in addition to the other features I implemented which has enabled him to improve his chess playing. He also suggested some improvements that were mostly associated with the GUI, namely square highlighting which seemed to be a recurring aspect to implement - initially, as aforementioned I claimed that implementing such an improvement would be based on preference as arguably not highlighting squares that constitute as valid moves would enable the player to more quickly learn what moves valid and what are not, where

they would be unable to rely upon a visual prompt - however, the fact it is being suggested by a user has led me to believe it may indeed be a useful improvement. Having the program alert the user when a check instance arises also seems like a fairly intuitive improvement.

Potential Improvements and Extensions

Through the self-analysis of my program in addition to the feedback I received from the users, I have managed to amalgamate a list of suggestions that can be implemented as improvements or extensions the exisisting system.

- **Move validity board highlighting** - Whenever the user clicks on a piece to move, the squares which constitute as valid moves for that piece could be highlighted a certain colour on the board, indicating to the user more obviously and intuitively the squares the user can move it to, preventing the user from attempting a move they think is valid only to find out later the move is rejected by the system and is actually invalid.
- **Move recommending board highlighting** - Whenever the user clicks the recommend move button to have a move recommended to them, the recommended move could not only be displayed as text on the GUI but also have the square containing the piece that should be moved, as well as the square it should be moved to, highlighted to make it more obvious and immediate to the user what piece should be moved. This also combats the fact that the piece string value is printed as the piece that should be moved on the GUI, which the user may sometimes not understand which exact piece it is referring to. This colour would also need to be different to the colour used for valid move highlighting.
- **Clear turn indicating** - The side whose turn it is to make a move could also be displayed on the GUI to make it more clear to the user, particularly in two-player mode
- **Check indication** - When a side enters check, a text could be displayed on the GUI to indicate this to make it more obvious to the user
- **Timed modes** - More game modes could be implemented as options for the user to choose, including timed modes where the user has a certain amount of time per move - this could be implemented in AI mode where obviously only the user has a certain amount of time per move, and in two-player mode where both players have a certain amount of time per move. The amount of time per move could be selected by the user prior to a game, and if a player runs out they automatically lose. Another time mode could also be implemented in which totally the user has a certain amount of time for all their moves - i.e. 60 minutes, and if the user's total time taken for all their moves so far exceeds that limit they automatically lose. In two-player mode both players would have this time limit.

- **Increased difficulty range** - More difficulty levels of the AI could be introduced, where this could potentially be implemented as a slider to allow for a more fluid difficulty adjustment. Introducing more difficulty levels would of course require more factors to be adjusted to produce the difficulty levels, which could be done by not only changing the depth of the minimax tree search but also using methods such as iterative deepening to limit the amount of time the AI has to calculate a move. Further adjustment could also be done to the heuristic evaluation function, where mobility values could not be calculated for certain pieces to provide a further number of adjustments for the difficulty - positional bonuses for pieces could also be ignored, which will be explained next.
- **Heuristic evaluation function improvements** - To allow for evaluation values of board states to be calculated with even more precision, further factors could be taken into account, namely piece positional advantages. For each piece type, there are certain positions on the board that enable the piece to be of a more potent threat - for example, bishop pieces being on the leading diagonals; that is, the longest diagonals stretching from the corners of the board to other opposite corners - knight pieces being in the centre, pawn pieces occupying the centre 2 squares, etc, which could be taken into account to provide a better board state evaluations and enable an even more competitively capable AI.
- **Minimax improvements** - Further optimisation algorithms could be applied to the minimax search to enable even more optimal moves to be calculated, particularly the quiescence search - this optimisation is designed to avoid the "horizon effect", where due to the minimax seach stopping abruptly after a certain depth, if a particularly negative event were to occur on the layer just underneath the last layer of the tree produced, this would go unnoticed by the algorithm. Hence the quiescence search optimisation checks for any of these negative events before finally returning an optimal move, which is done by searching all the "quiet" leaf nodes which are all the leaf nodes containing board states in which no winning "tactical" moves can be performed.

Final Conclusions

Overall, I believe the outcome of my project has been successful, where it has managed to meet all of the objectives and requirements that I initially set out to achieve, and has garnered positive feedback from the clients that I designed the progam for. I look forward to potentially further extending the project's features and capability beyond A-Level, in particular attempting to further improve the capacity and power of the AI to even higher standards.

Notes

Contact us

T: 0161 957 3980
E: computerscience@aqa.org.uk
8am–5pm Monday to Friday

aqa.org.uk