Name : Anthony Guy Candidate Number : 6190 Centre ID : 43309

Computer Science Programming Project

Retro Rampage

Developed by Anthony Guy
2021 - 2022

Analysis

Description of the problem

Outline of problem

A games company has requested a new game to be developed in the style of a racing game. The game will be sold to users for entertainment and should be entertaining enough to encourage the users to purchase more games by the company.

Computer system features

The computer system should have some form or equivalent of python 3 to be able to run the program, as well as a compatible operating system to produce and manage multitasking and window management, as well as inputs and outputs such as keyboard / game controller inputs and display / audio output. The system should also have the package ‘pygame’ installed which manages inputs, outputs and manages the running and displaying of the game.

Stakeholders

Games Company

The games company has the largest influence on the development of this game as they have specified what features they would like to be included, as well as the style and selling of the game. This is important because the developers will not know what they need to code in order to satisfy the requirements of the company. If the standards are not met then the code will not be used and the developers will either have to improve the existing work or start afresh with a new result in mind. These people are the ones who came up with the idea of the game and hired the developers to actually make it according to their requirements / standards.

Developers

The developers are in charge of how the features specified by the games
company is implemented and how the game looks. They can change any feature of the game and have the second most influence on the features of the game. These people are the ones who actually make the game and send it to the games company, including any further updates and fixes to the game after the initial release. Their job is to find a solution that implements the requirement from the company that hired them.

Users

The users will have some influence on the game as they can report bugs and glitches in the game, as well as suggest ideas to the games company - if the games company will allow the idea to be implemented then it will be passed down to the developers. The users also influence the game by how many players use it and enjoy the game - eg. If nobody plays the game or people do not find the game fun then the company will lose money and something will have to be changed. Their main purpose is to buy the game so that the games company can make money from developing it, and in turn the users get to play the game itself. Whether the players enjoy the game is not fundamentally vital to the company, however it is good for them to keep a good reputation otherwise people will not play their games.

Why Computers?

Suited to a program

The game is suited for a program because it allows the computer to understand the information and can be handled by the operating system. Being a single program allows the users to only have one single centralised file that includes everything in order to run the game.

Suitable Features

The features of the game such as being able to produce a fast video output and calculating the different graphics and layers. A person wouldn't be able to think fast enough and show what is happening fast enough to play the game, however a computer is able to do several frames a second while including all the logic and management systems eg. window management and multithreading.

Valuable to stakeholders

The output to the stakeholders for the users is valuable as they need the output to be able to play the game for their entertainment.
The output is valuable to the company as they need the users to be encouraged into purchasing the game and also other games that the company also owns in order to make more money.

Owner Interview

1) **Why do you want this program to be developed?**
   A - *To sell to users and make money, and for people to enjoy my game and ideas.*
2) **How will the software be used?**
   A - *The software will be used purely for entertainment purposes, as it is a game.*
3) **What platforms will the software be developed for?**
   A - *The software will only be available in python3 for now, but may progress to other operating systems in the future. However, python3 can be run on Windows, Linux and MacOS and therefore the game is very compatible with most computers.*

4) **What style would you like the GUI to look like?**
   A - *the GUI will have large and intuitive buttons so that it is easy to navigate for the users, the style will be the same as the main gameplay, as it re-uses assets for the menu and background.*
5) **What are the targeted system requirements for the program to run?**
   A - *The software should require a minimum of a dual core 1Ghz + processor so that anyone can play if they want to, the game will include special features such as no animations for low-end computers to be able to run the game smoothly (although it will not look as good).*
6) **Are there any other extra features that you would like to be included in the game?**
   A - *The game shall have power ups that the player(s) can use to gain an advantage over each other, as influenced by 'Blur' and 'Mario Kart' however uses a 2D Retro themed style of gameplay.*
7) **What is the budget to create and develop this software?**
   A - *The budget for this game is starting at $50,000 but can be increased if needed and as development continues.*
8) **Will the program be sold or free for users?**
   A - *The program will be sold to users in order to make a profit from the game, the game will not be very much to purchase and may only be sold for around £1.00 a copy.*
9) **Are there future plans for a sequel?**
   A - *If this game is successful then we will look into making a second game, however it may be a better idea to just improve upon the current game by adding more content such as more maps and levels that makes the game have more things to do for the users.*
10) **What types of inputs can users use?**
   A - *The users will be able to use keyboard and mouse, or a game controller. If two people are playing then they will either have to use both a mouse and keyboard and controller separately, however not all people have a compatible game controller, so there will be support for one player to use the arrow keys whilst the other player uses WASD control on the same keyboard. The downside to two people using the same keyboard is that not everybody can use WASD controls easily, however there is no other standard way for people to input movements.*

Questionnaire

Overview

Retro Rampage - Questionnaire

This is a questionnaire about the new game being developed - Retro Rampage

What is your age? *

Short-answer text

What is your gender? *

- Male
- Female
- Prefer not to say

What would be a good name for this game? *

Short-answer text

How much are you willing to pay for a racing game?

Short-answer text

What platforms would you get the game from?

- [ ] Steam
- [ ] Microsoft Store
- [ ] Epic Launcher
- [ ] Proprietary Launcher (eg. Website download)

Would you play this game with one or two players?

- ( ) One
- ( ) Two

What colour scheme do you think best suits a racing game? *

- ( ) Red
- ( ) Orange
- ( ) Yellow
- ( ) Green
- ( ) Blue
- ( ) Purple
- ( ) Pink
- ( ) Grey
- ( ) Black
- ( ) Other...

What settings would be useful to you? *

- [ ] Screen resolution presets
- [ ] V-sync
- [ ] Reset screen
- [ ] Volume
- [ ] Flip screen
- [ ] Fullscreen
- [ ] Screensaver
- [ ] Hide cursor
- [ ] Resizable window (Fixed size or not)
- [ ] Other...

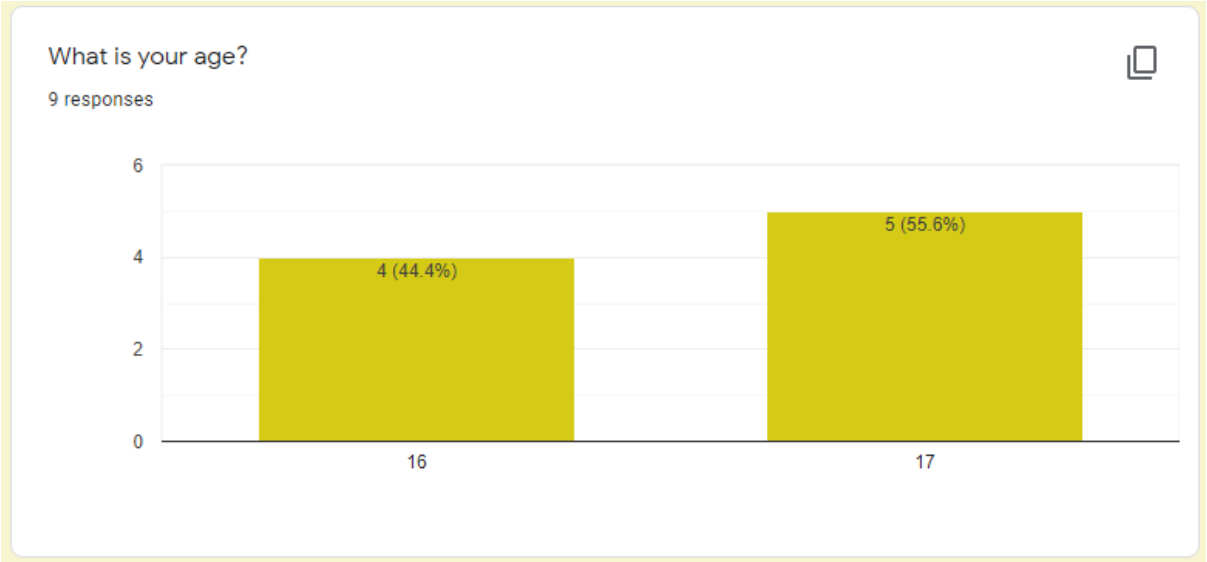
How many levels / maps would you like to see?

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

How many AI cars would you like to race against?

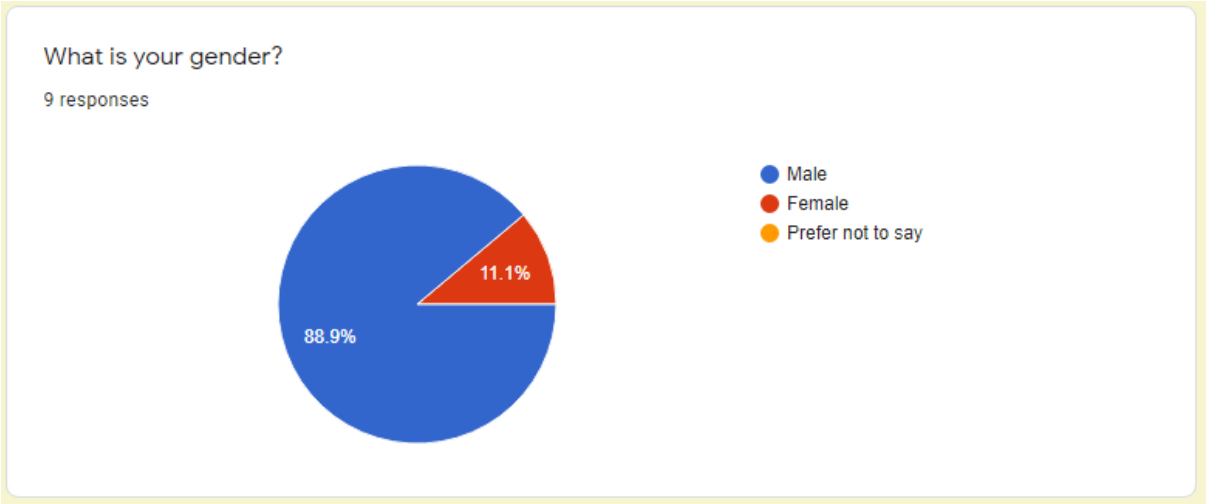
1	2	3	4	5
○	○	○	○	○

Age Demographics



Out of the 9 people that completed the questionnaire, just over half of them were ages 17, with the rest being age 16. This means that our game should be targeted to youths (mainly teens).

Gender Demographics

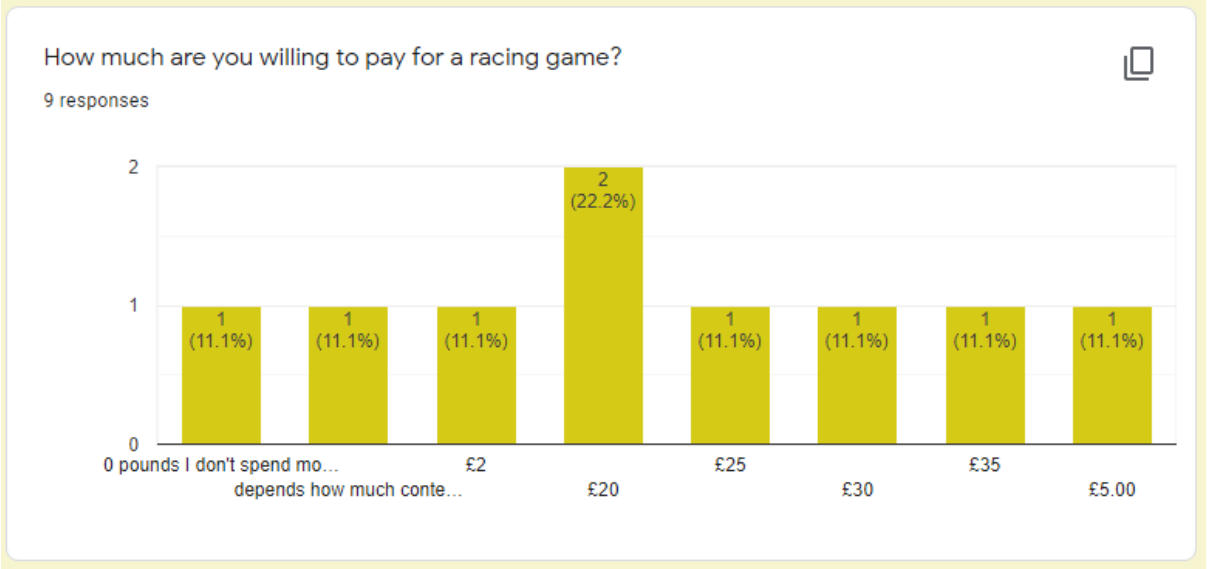


The responses show that 89% of people are males, this means that we should look to implement things that are more appealing to men - such as motorbikes as well as cars.

Name Suggestions

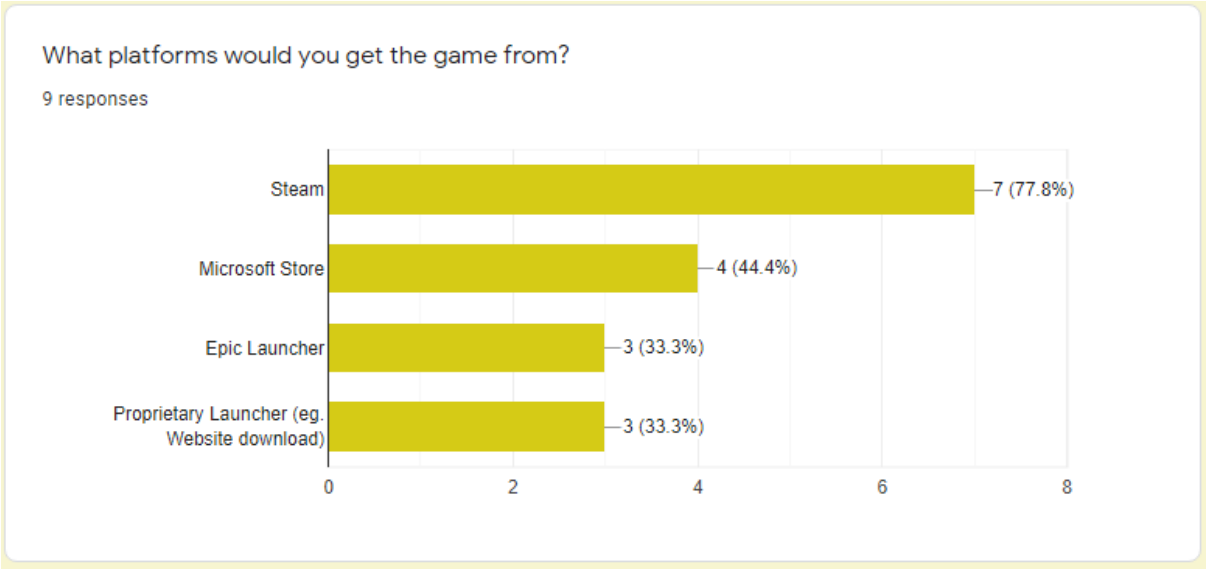
Out of the responses to this question, I think that the most appealing name for this game is ‘Retro Rampage’. This will be implemented into the advertising and game design.

Suggested price range



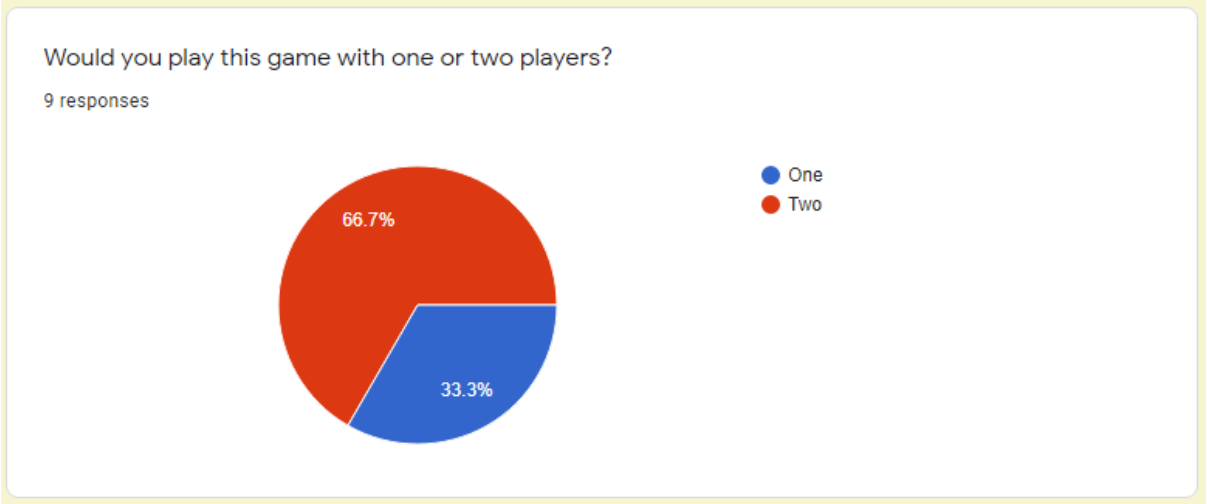
The responses show that most people would be willing to pay around £20 for this game, however they did not know that it would be in the style of an indie game. With this in mind, I believe that the appropriate price per game would be around £3 - £5.

Platforms



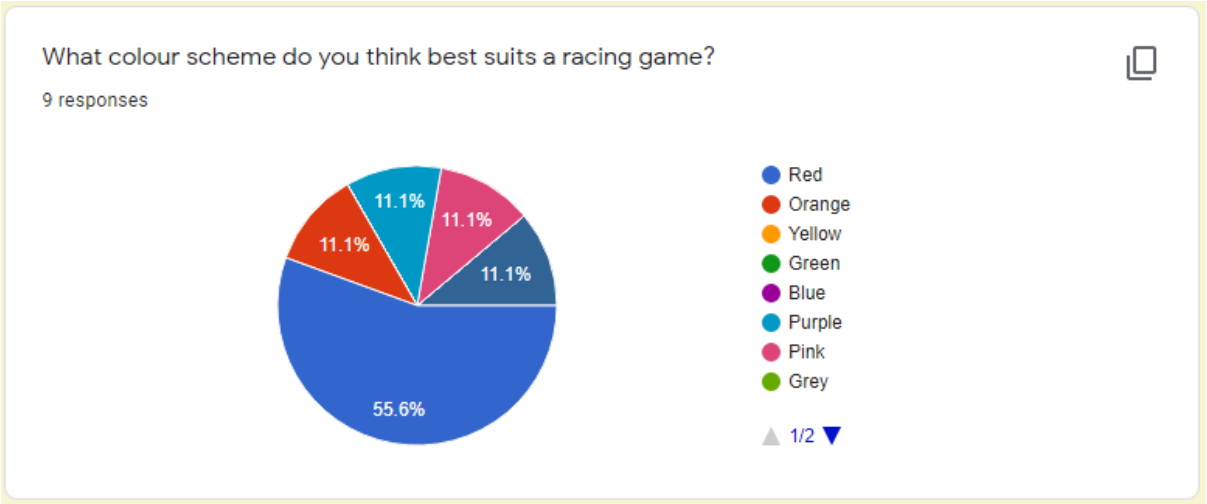
According to the responses, it would be most convenient for advanced gamers that the game is available on steam, but it should also be available on the microsoft store for people who are not as familiar with dedicated game launchers.

Amount of players



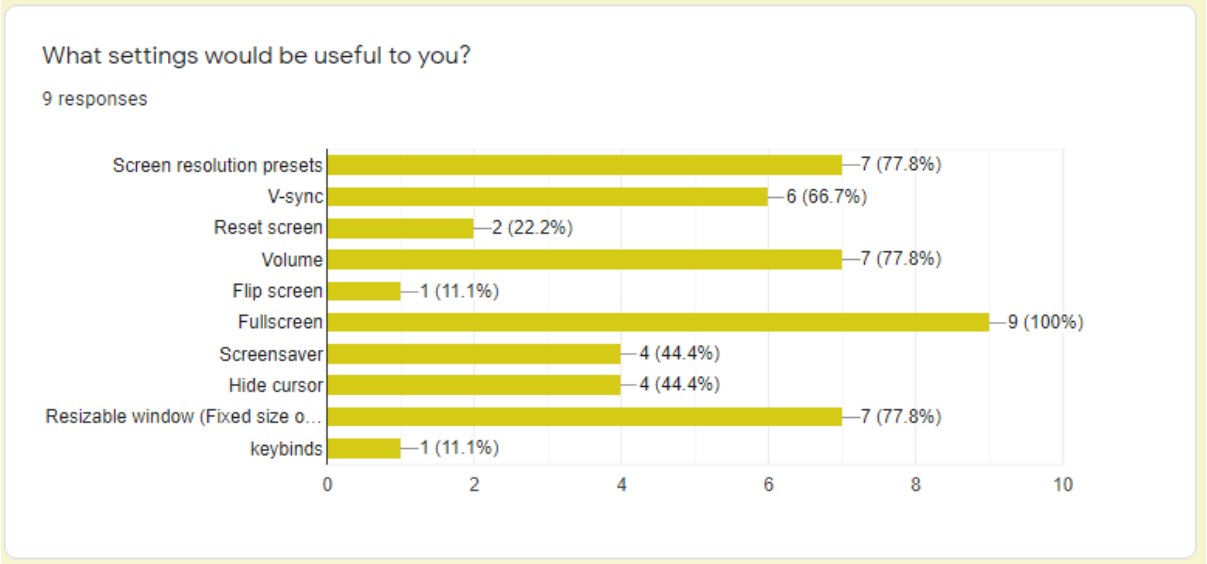
The responses show that if a two player option was implemented into the game, then 67% of the players would use it to play with either a friend or family member. Therefore, this option will be implemented into the game.

Colour scheme



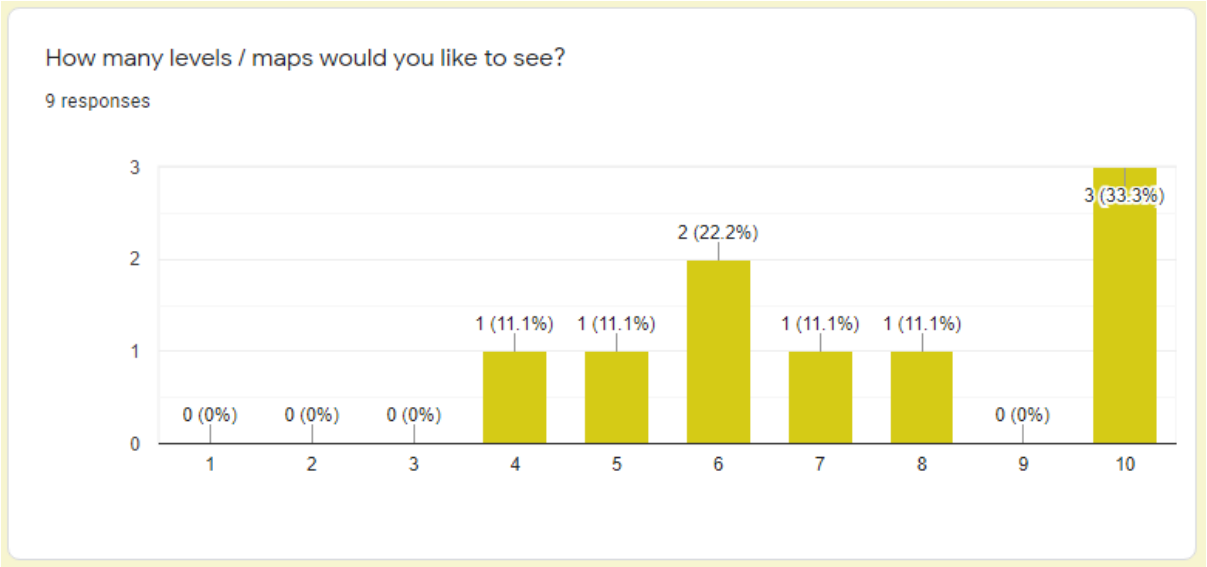
The data above shows that people would be more attracted to a red colour scheme for this game… However it may be more appropriate to use assets for the UI design as well as the gameplay itself.

Video settings



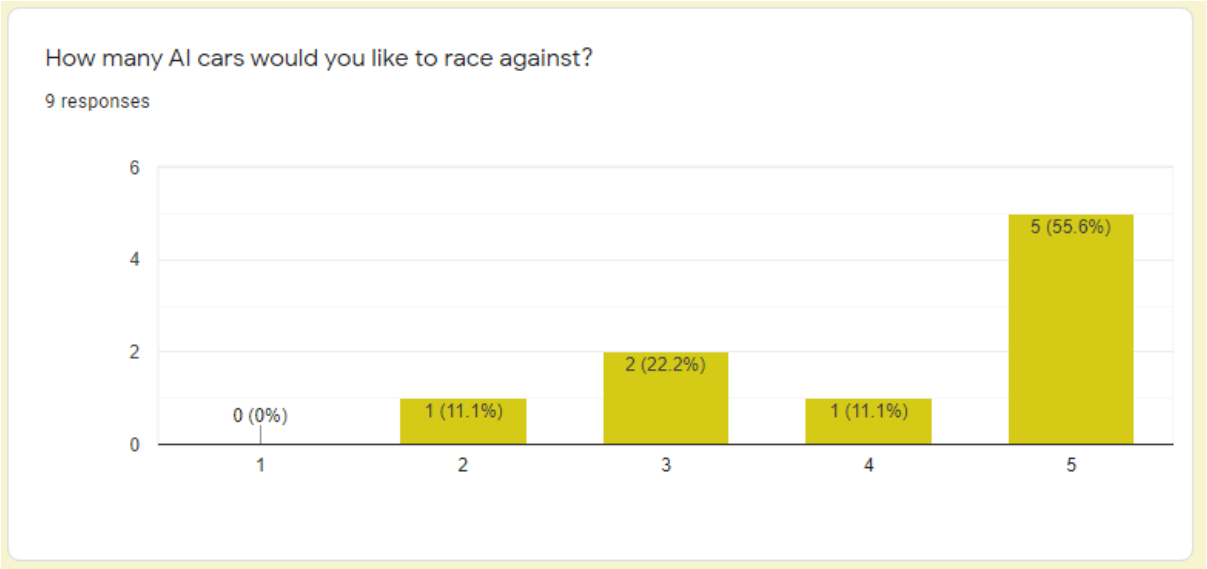
People would find having a fullscreen option very useful for a setting, with a three way tied second of volumes, resolution presets and resizable window option. Having a resizable window could lead to a lot of errors and bugs in the game, so that will not be added but resolution presets and fullscreen options will definitely be added to the game. A screensaver and option to hide the cursor while in game will be added. An option to reset / refresh the screen window may be added as well.

Amount of levels



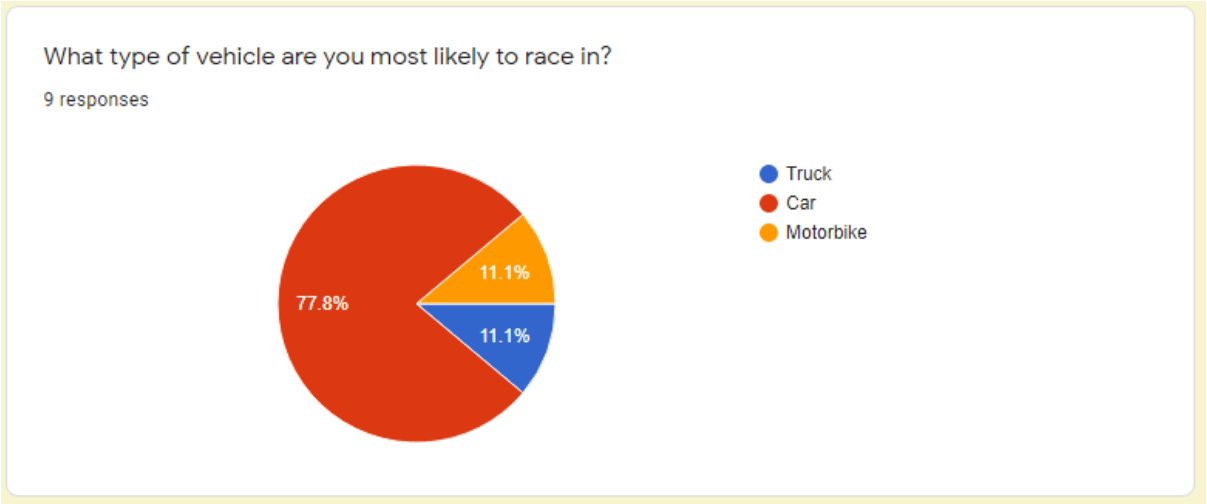
The amount of maps that are implemented into the game will heavily depend on the speed and progress of development, but as people would like to see at least 10 levels implemented… The development team will aim for more than 10 levels to be implemented.

Amount of AI



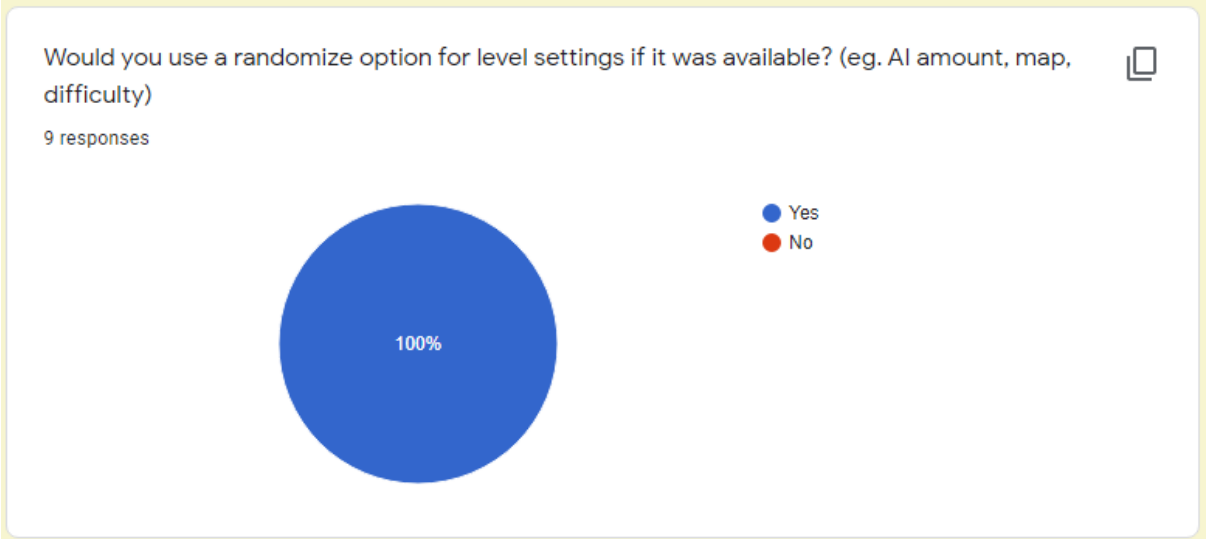
The survey data shows that people would find the game more fun if there is a maximum number of 5 AI cars that the player can race against, therefore there will be an option of up to 5 AI cars to race against in the main menu.

Type of vehicle



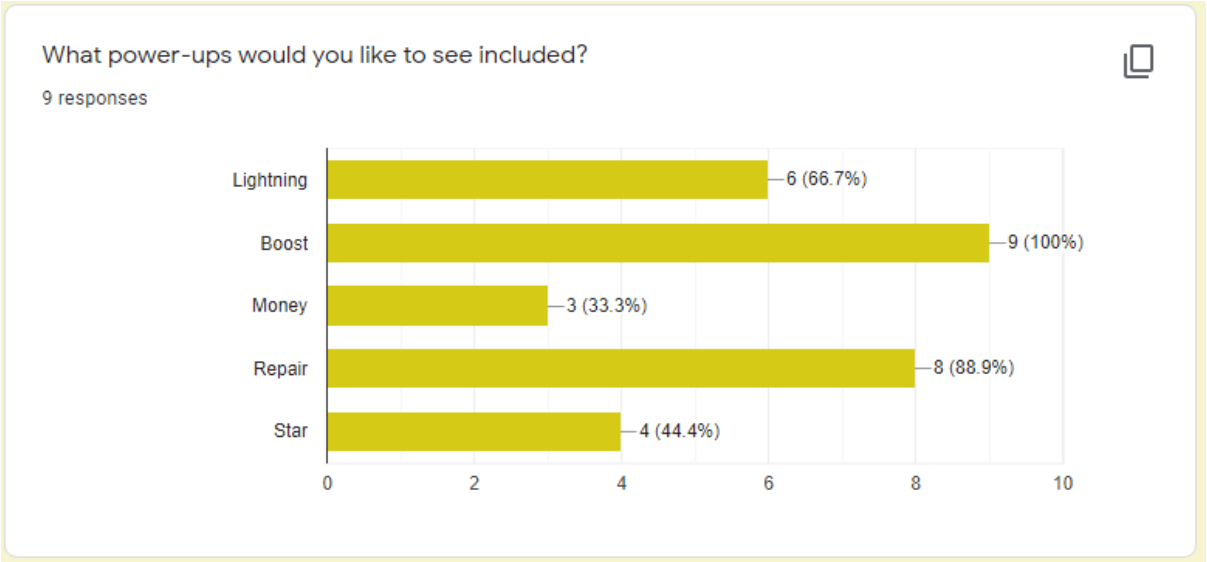
The data above shows that the majority of people would just use cars for their vehicle, but as shown by the previous gender question, some males picked motorbikes and trucks for their vehicles. This is important as it proves that we should implement features that give more interest to men.

Randomisation



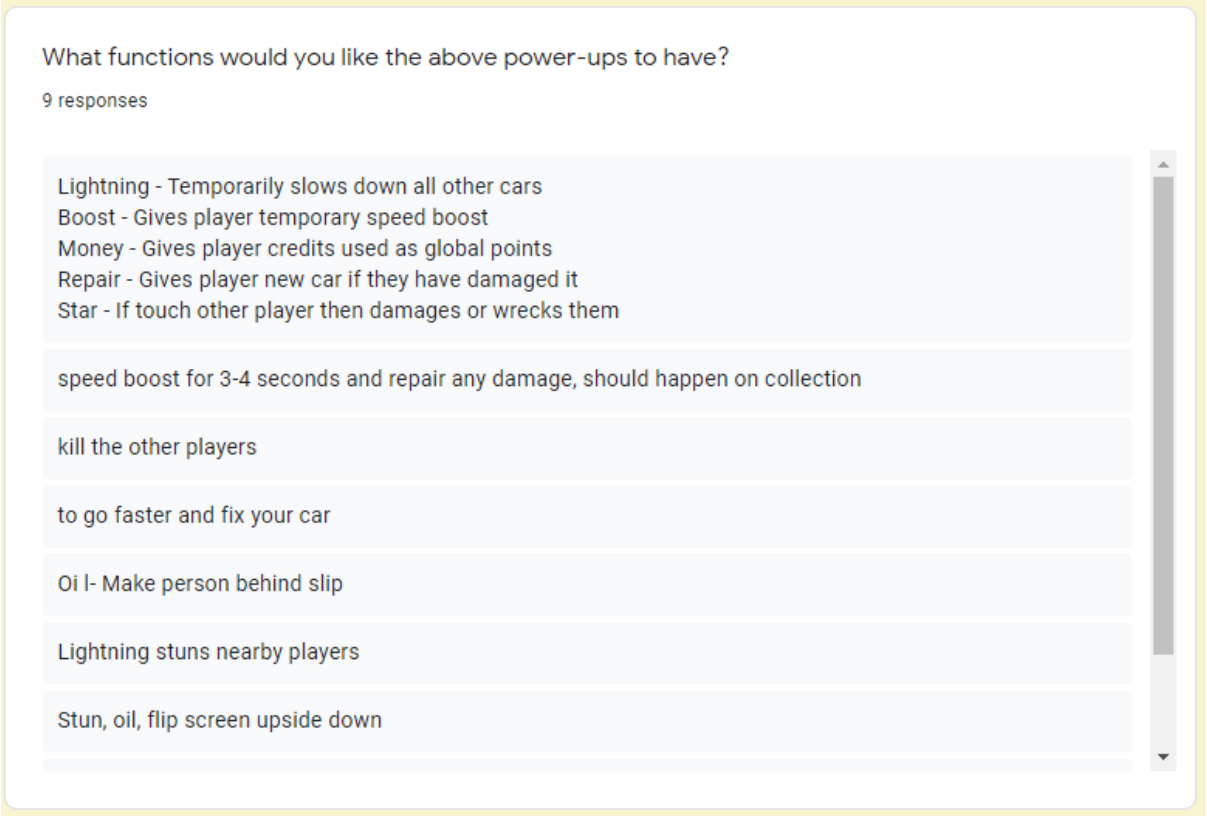
Out of the 9 people surveyed, all of them said that they would like an option to play a race where all of the options are randomised. Therefore to make more people play and buy the game we will implement this option so that players can have an infinite amount of campaign levels to race.

Power-Ups



The data above shows that all of the people surveyed would like a boost power-up to be implemented. 89% of people said that they would also like a repair and lightning power-up to damage other players' cars and be able to repair their own. This would be good to implement as it gives the game another aspect of fun and something more competitive apart from just racing around a track.

Power-Up functions

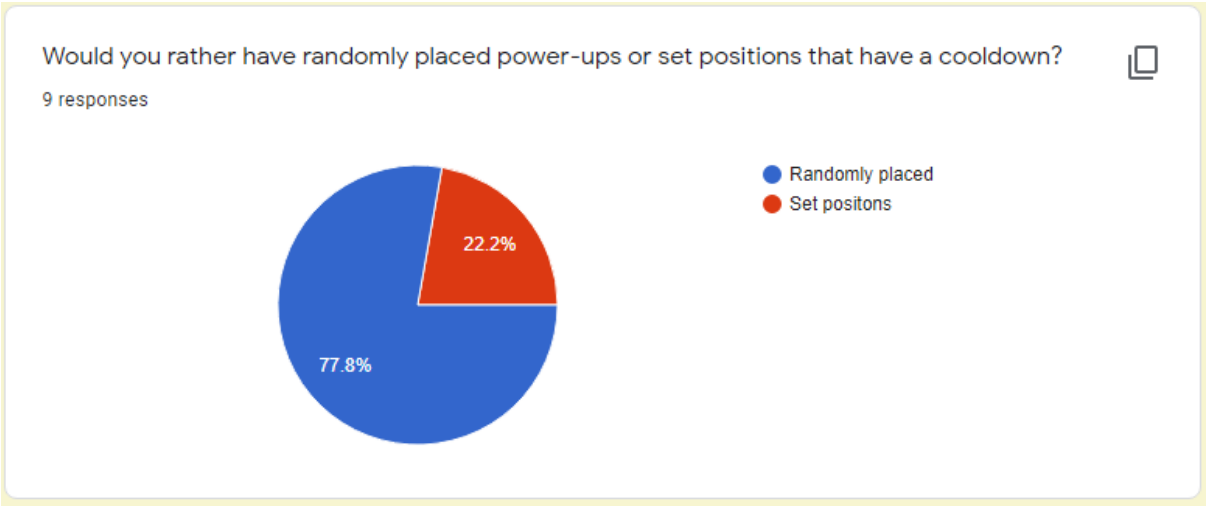


stun-Lighting, temporary speed - boost, repair-fix damage

Increasing speed, decreasing damage

The responses above imply that the following functions should be assigned to the functions:
Lightning - Slows down every other vehicle apart from the person who activated it
Boost - Gives the player that activated a temporary speed increase
Money - (not implemented)
Repair - Fixes damage to car that the player has activated the power-up
Star - gives the player invincibility, repairs their car and damages any other car that they hit

Power-up placement



The people that were surveyed said that they would more than 75% prefer randomly placed power-ups compared to set positions that they regenerate from. This will be implemented as the users have stated in order to keep the users happy with the game and make it look more enticing for users to purchase and play.

Market Research

The pictures above are from a game called Formula Retro Racing. It is an indie-based game on steam that has a very similar style to Retro Rampage. In this game, players can see a 3D version of a racing game with very basic graphics… players have the ability to wreck others, but does not include any form of power-ups. This is important because it will give Retro Rampage a competitive advantage over games like this one, purely by including more features even though it is only a 2D based game. This game also includes a split-screen PVP mode for up to 4 players, as well as controller support. This means that including these features work well with customers and therefore should be implemented into Retro Rampage as per the specifications. There is a best lap time and race time included in this game that would be a good thing to implement into Retro Racing if there is sufficient time to implement it. Having a leaderboard means that people will have infinite competition apart from the AI that will be implemented.

At the time of writing this… the last update to be implemented into Formula Retro Racing was 6 days ago, and the game was released on the 15th May 2020. Having regular updates is important as it helps fix the never-ending amount of bugs and glitches in the game. It also shows that the developer is active and still working on the game. We should try to do the same, as it will show the community that we are dedicated to producing the best game we can for them and also keeps them happy as we reduce the amount of bugs and glitches in the game for a smoother experience.

The GUI of our game will be built using the same assets as the main gameplay. We believe that this is a good decision as it will fully immerse players in the style of the game. It would look very weird if a classic indie game had a very modern and sleek GUI design.

The main limitation of our game compared to this game is that ours is limited to being a 2D game rather than a 3D game. This is due to using pygame and python as the base language to build the game on. A 3D game can be achieved using python, however it would not be applicable for the amount of time and money allocated by the customer.

Top 10 Mario Kart Characters

The game above is called Mario Kart 64… and it is a very successful and still popular franchise of games. Players mainly like this game because of the cartoon-style racing and the amount of power-ups and game modes. From this game, our game should implement the same style of GUI with the format that players choose the race type and character (vehicle). The large buttons make it very easy and intuitive for the players to easily pick the option that they want to choose. The formatting and organisation of the button layout for the players to pick the race type is very clearly laid out. Retro Rampage should have a similar format where players pick the amount of players first, then the race below and finally a page where players can each pick the vehicle and colour that they choose.

Similarly to Formula Retro Racing, Mario Kart also has a leaderboard. This would be a good thing to implement into Retro Rampage as it gives the players a clear view of what place the player finished in as well as a way to view all of the places that the rest of the AI or other players finished in.

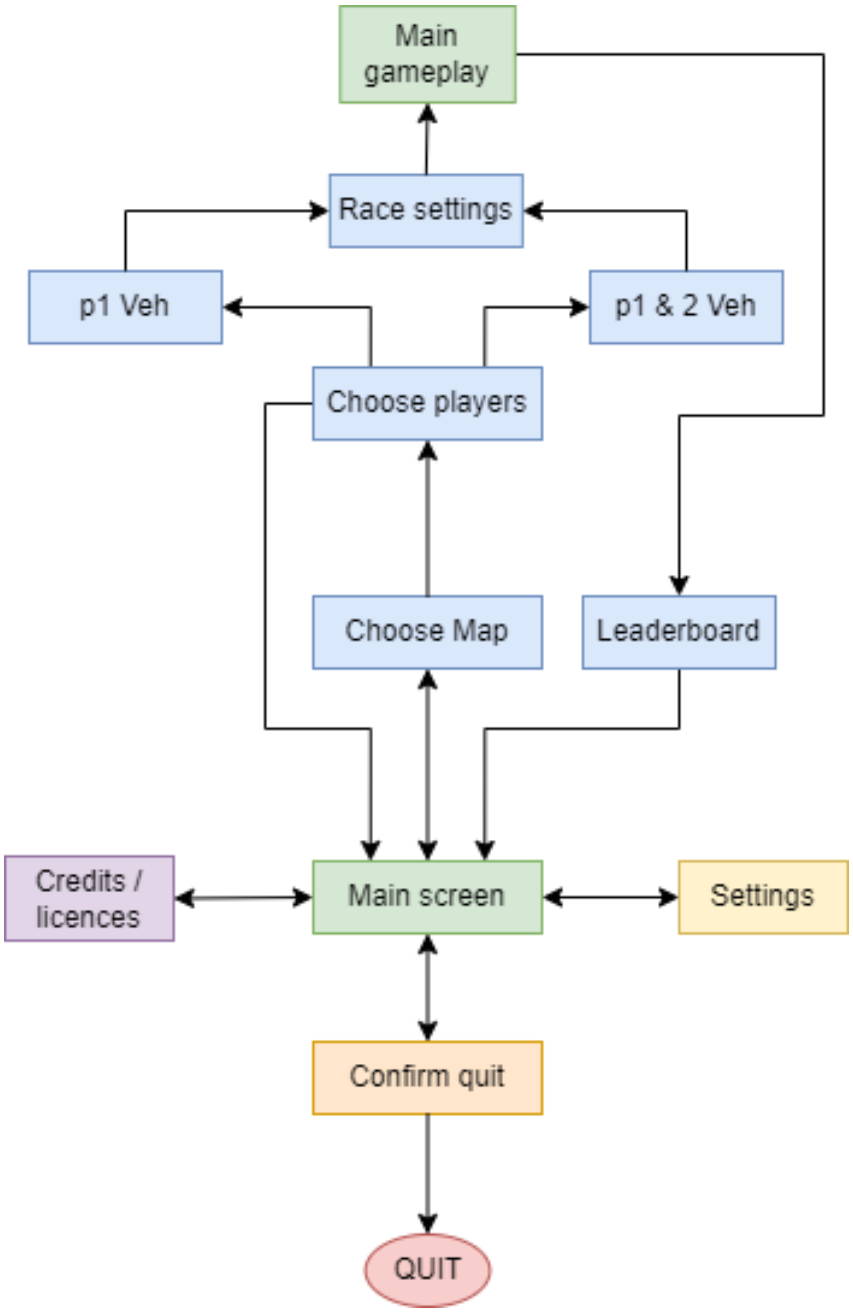
The limitations of Retro Rampage will mainly be that it is not a 3D game, so visualising and finding a good way to view what is happening in the game without any depth will be hard as it is much easier for the player to see what is close and what is far in a 3D game, however it is something that a 2D game lacks.

The functionality of the game will also be bound by the functionality of the library used to build the game on, called ‘pygame’.

Client Proposal

After researching similar markets to Retro Rampage, we have discovered that the best features that attract people to the game are a leaderboard, clear and spaced out buttons that are well organised by category and have as little text as possible including relative icons for the users to glance at, and to have a 2 player split screen design for the multiplayer mode.

The client likes the idea of implementing a leaderboard to see who won the race at the end, and they also have ideas of how they would like the buttons to look and be organised by category. The organisations are:



The diagram above is a flow diagram of the UI organisation and directions. The arrows indicate the different directions that users can travel between the windows.

Stakeholder Requirements

Requirements:

- An intuitive and easy to use UI
- Multiple maps for users to play on
- Different vehicles and colours
- A boundary system to stop players from cheating in the game
- The ability for users to change settings of the game such as resolution and fullscreen
- A two player mode for users to play with friends
- Settings for v-sync, screensaver and volumes of sounds
- At least 6 maps, with 10 being the max for people to play on
- 4 different types of power-ups that spawn randomly throughout the game and give players certain temporary advantages

Possible additions:

- A leaderboard where players can see who won and who didn't
- Up to 5 AI cars to play against while in the game that are fully autonomous
- Different levels of difficulty for said AI
- The entire program and required libraries compiled into one .exe for ease of use

Software and Hardware requirements

X32-bit system +
200mb RAM
200mb free storage space
1Ghz 2-core CPU +
(No GPU required)

Success Criteria

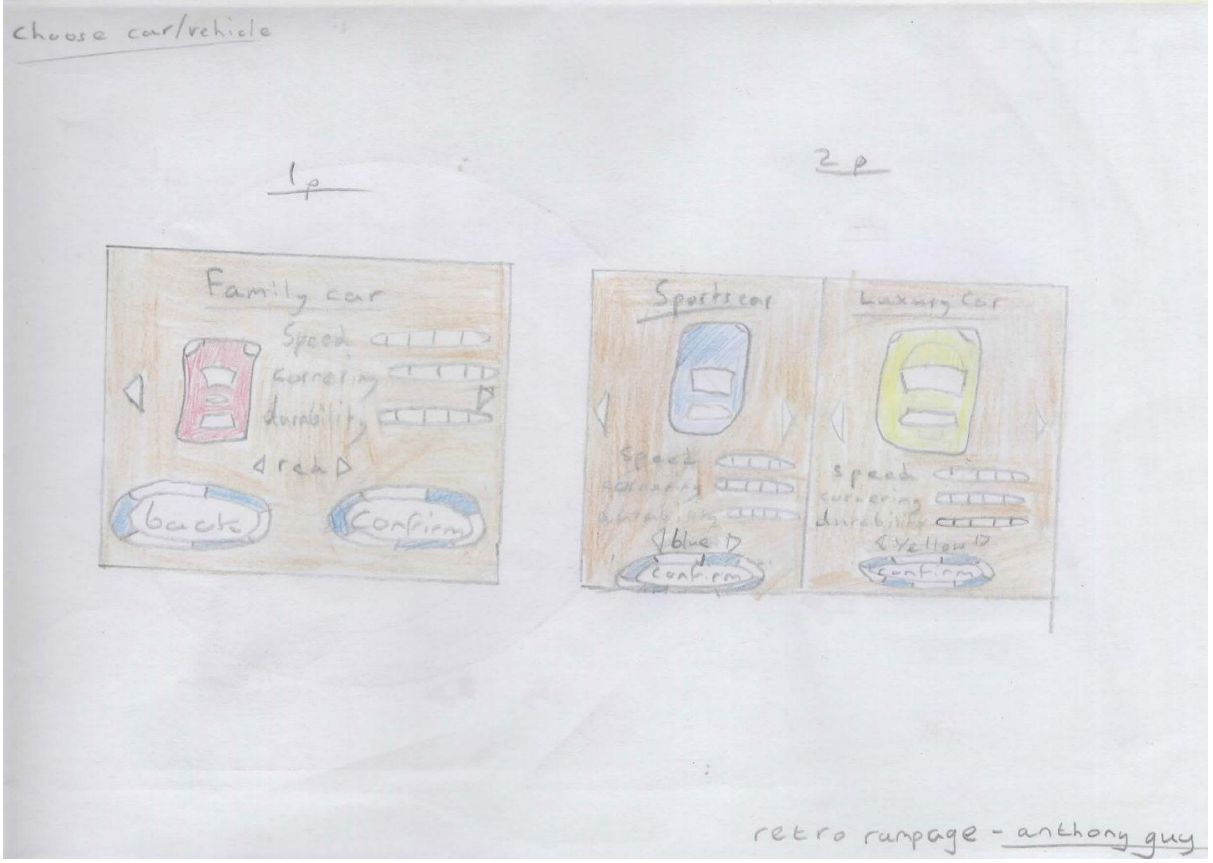
Number	Criteria	How to implement	Alternative
1	Main menu to start and change settings	Home screen with multiple buttons to move to other screen	Keyboard shortcuts to other windows like settings
2	Confirm quit screen	Yes or no button in centre	Quits without asking or waits before quitting
3	Travel between screens / windows	Screens stitch together and move across screen	Windows change with no animation
4	Users can change settings	Screen from main menu with options	Button in pause menu or corner to page

5	Users can view credits and licences	From main menu through button	Shows licences on launch of game
6	Choose 1p or 2p	Window from main menu	Directly from main menu
7	Select specific map or choose random	New window after 1p or 2p	On same window as AI amount and lap amount
8	Choose vehicle and color	After map and amount of players has been chosen, one by one player	Split-screen player control independent at same time
9	Amount of AI	Window after vehicle chosen	Before on main menu or after map selection
10	Confirm settings	Separate window before starting race	On same window as the race starts
11	Countdown to race start	Text in middle of screen	Text centred at top of each screen
12	Minimap to see where players are	Centred and fixed in middle of screen	Rotates depending on player position
13	Position of each player	Top left / right of screen	Both centred in middle top of screen
14	Amount of laps and progress	Centred at top of screen	Centred at bottom or above minimap if implemented
15	Ability to pause game	By pressing esc key	By pressing Enter or other button
16	Change settings from pause menu	Button to go to same window as from main screen	Custom window with limited settings
17	Option to change power-saving mode for low-end cpus	In settings menu	Upon first time launch
18	Option to randomise all race settings	Button Underneath 1p or 2p for each	On same page as map window
19	Leaderboard	After race has ended	After championship has ended

20	Ability to chain races into single championship	By selecting multiple maps at once on map window	Separate windows for each map chosen
21	Manually change hardware acceleration	Through settings menu it can be changed	Make a first time startup screen where users can change it
22	Change the speed of cars and AI	When choosing options for AI	When choosing the map
23	Change the menu scroll speed	Through an option in settings	In a manual config file separate from the game
24	Manually change the car scale	Under the same option in settings as changing the menu animation speed	In a manual config file
25	Manually change the menu and game scale	In settings next to the car scale	In a manual config file

Design

Hand-drawn designs



retro rampage - anthony guy

Race settings

AI amount
1

Map - 1. figure 8, Sumer

P1 vehicle - Sports car

P2 vehicle - Race car

Player difficulty

AI difficulty
2

Spectators
on

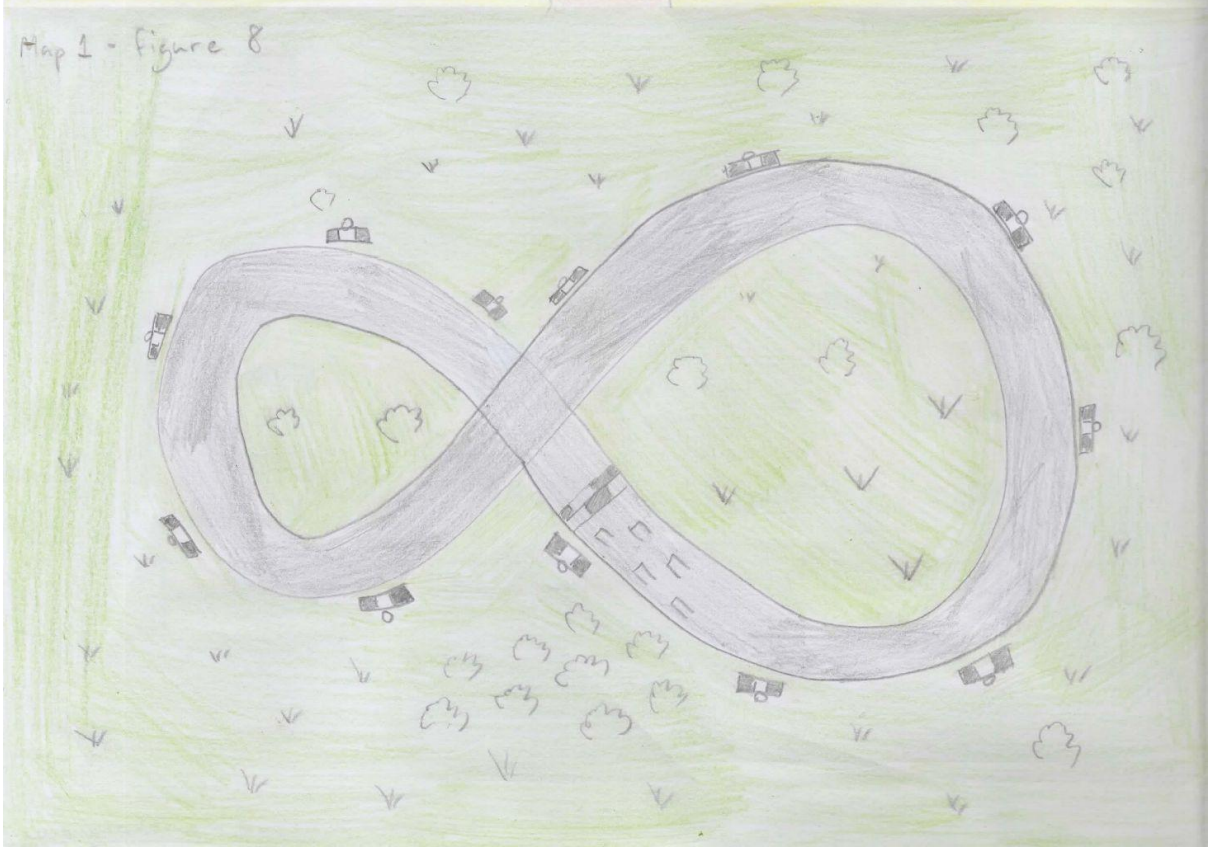
AI color
Red

Map Season
random

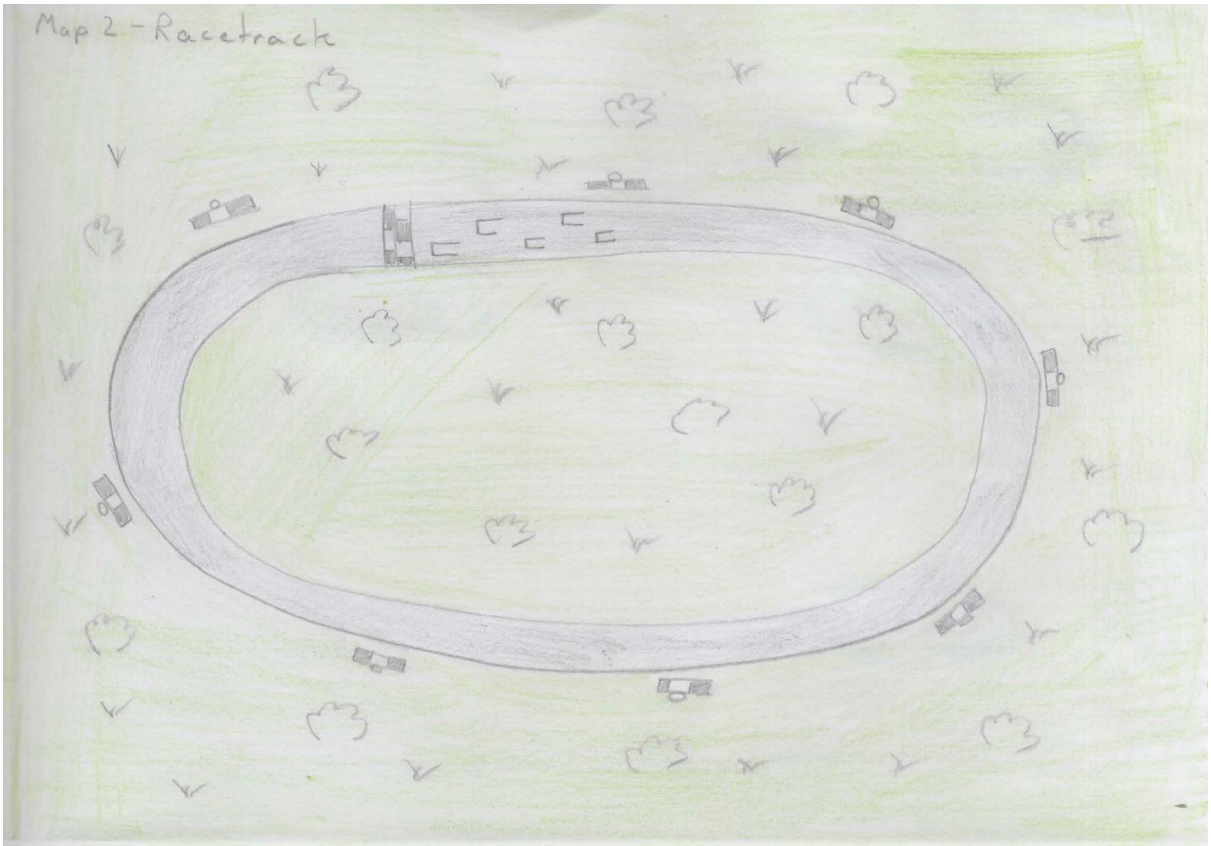
Laps
3

Start Game

Map 1 - figure 8



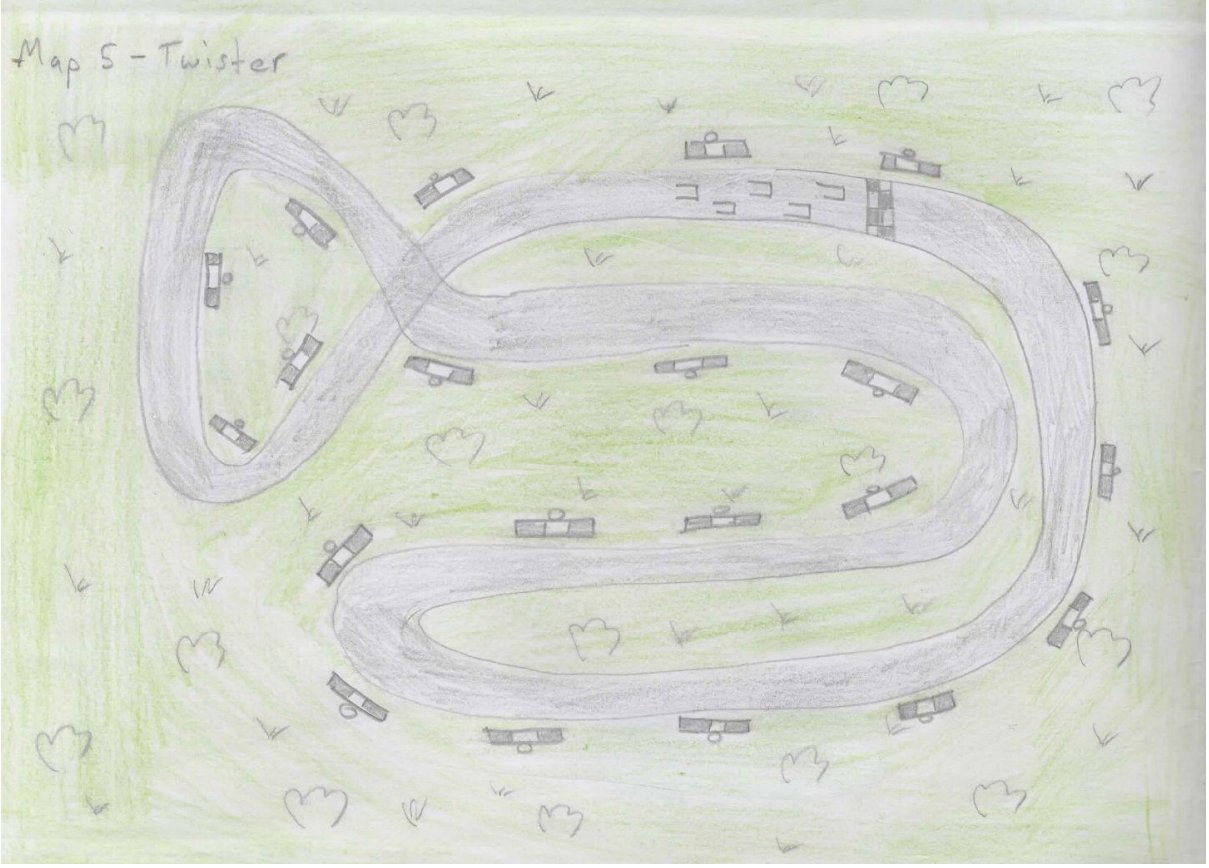
Map 2 - Racetrack



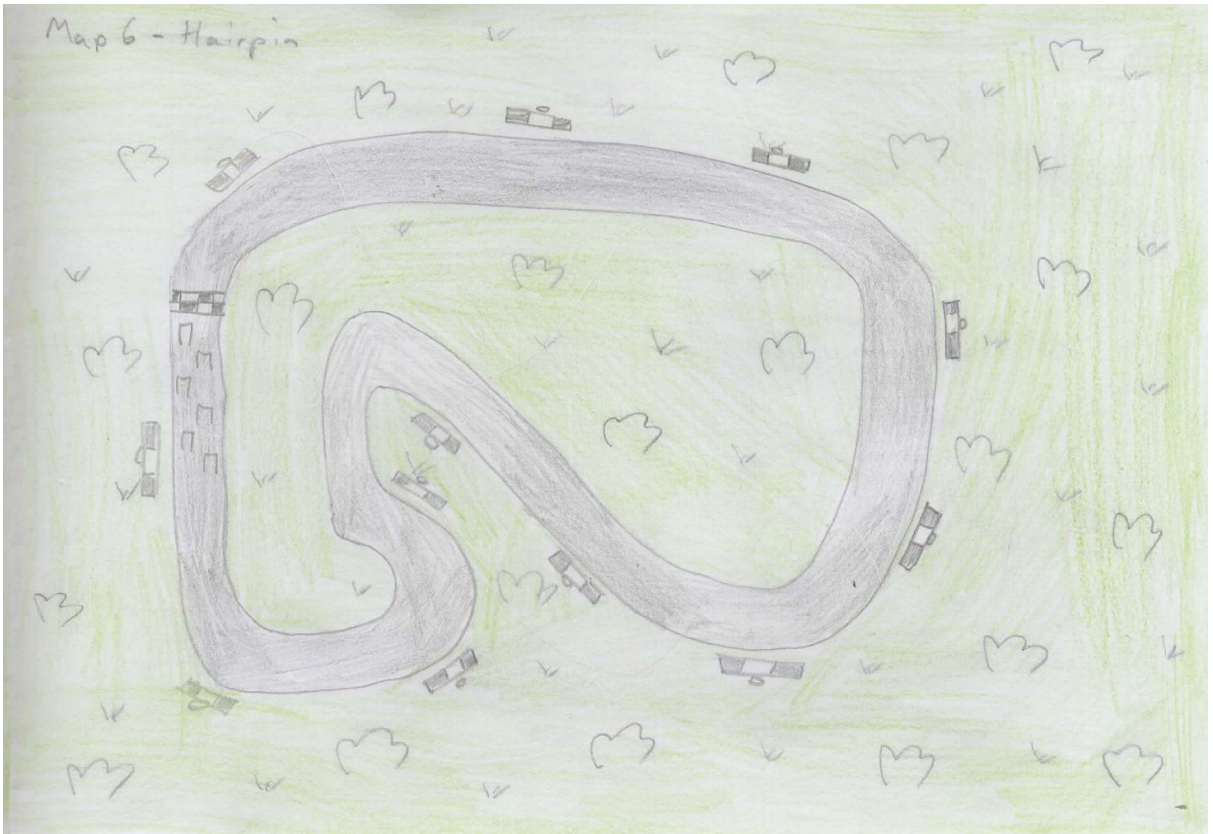
Map 3 - Clover leaf

Map 4 - Mountain Hills

Map 5 - Twister



Map 6 - Hairpin



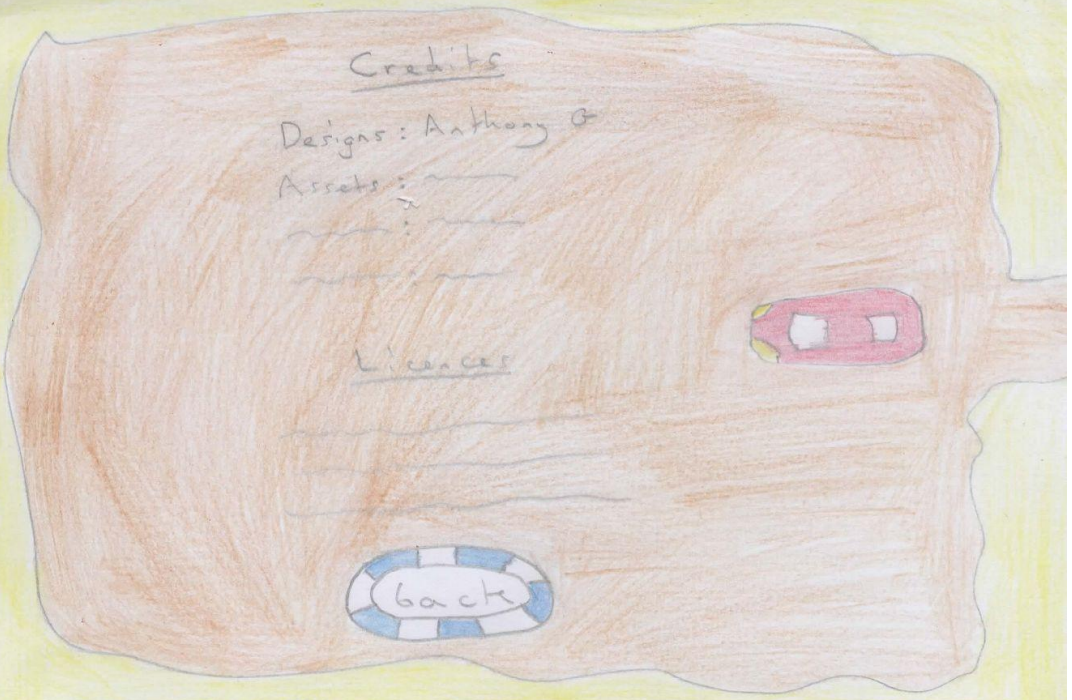
Map 7 - aerodrome

Map 8 - Hatchet

Map 9 - Ripple

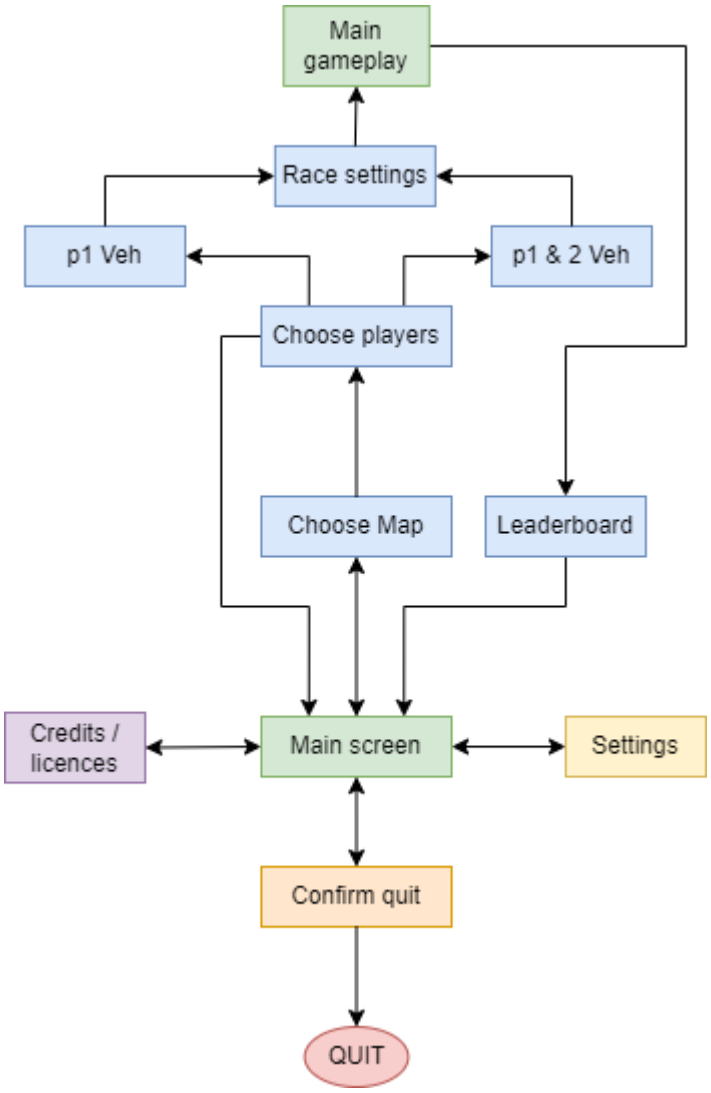

Map 10 - Teardrop
Leaderboard
Quit
Finished!
Settings
1st - [Player 1 name]
2nd - [Player 2 name]
3rd - Dave
4th - Bob
5th - Jimmy
6th - Andrew
7th - Thomas
8th - Sam
Map - 1. figure 8
AI difficulty -
Player difficulty -
Season - summer
AI amount - 5
Laps - 3
Main menu
retro rampage - anthony Guy

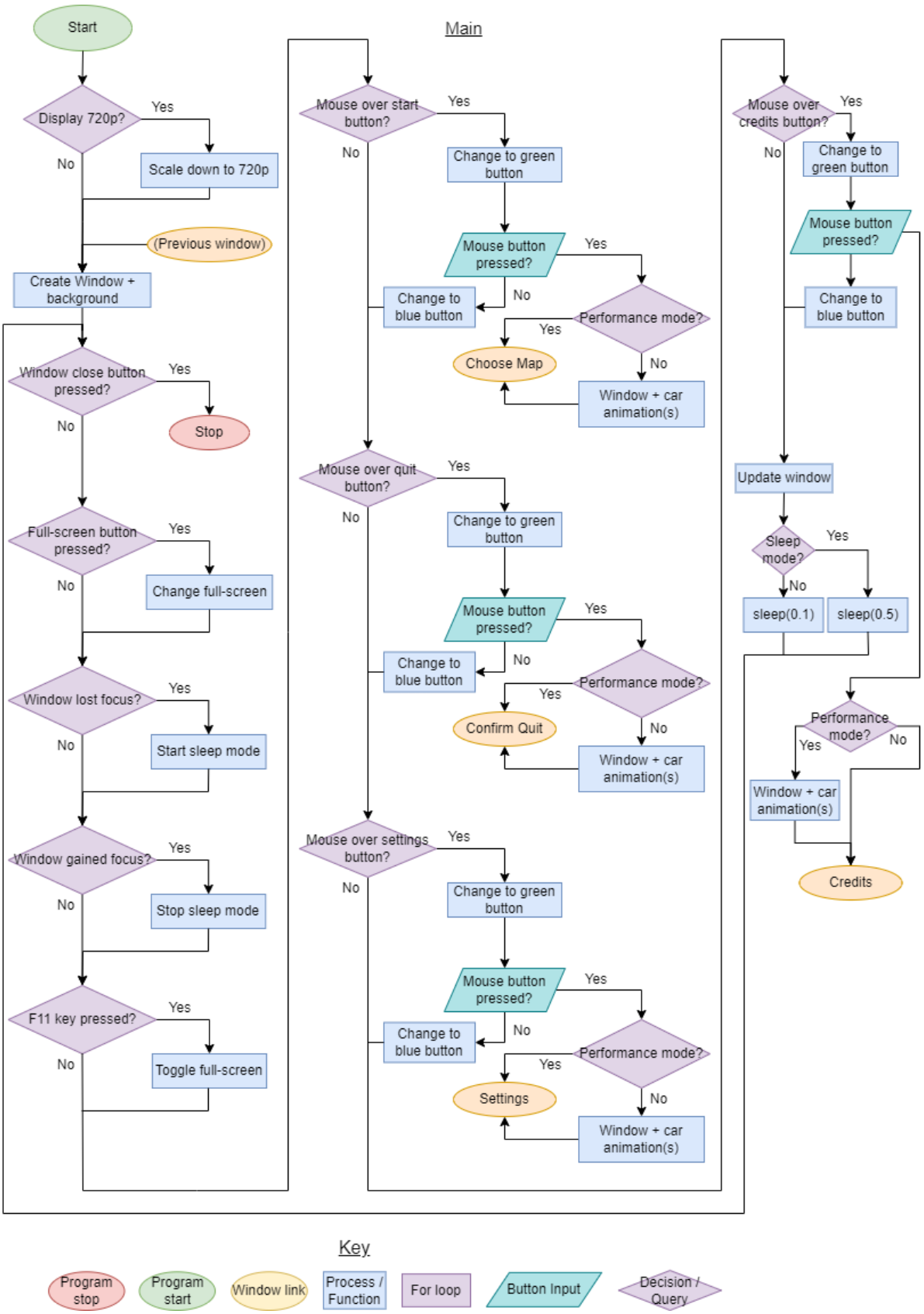
Credits

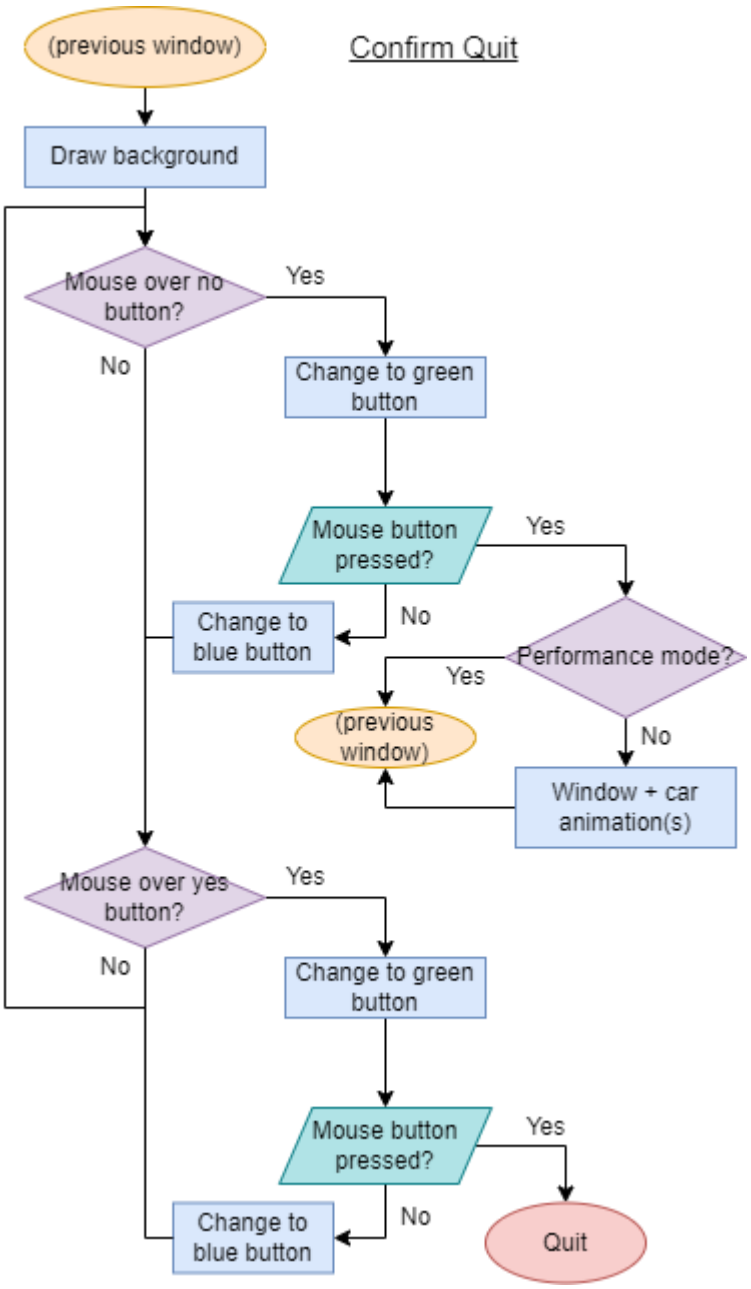


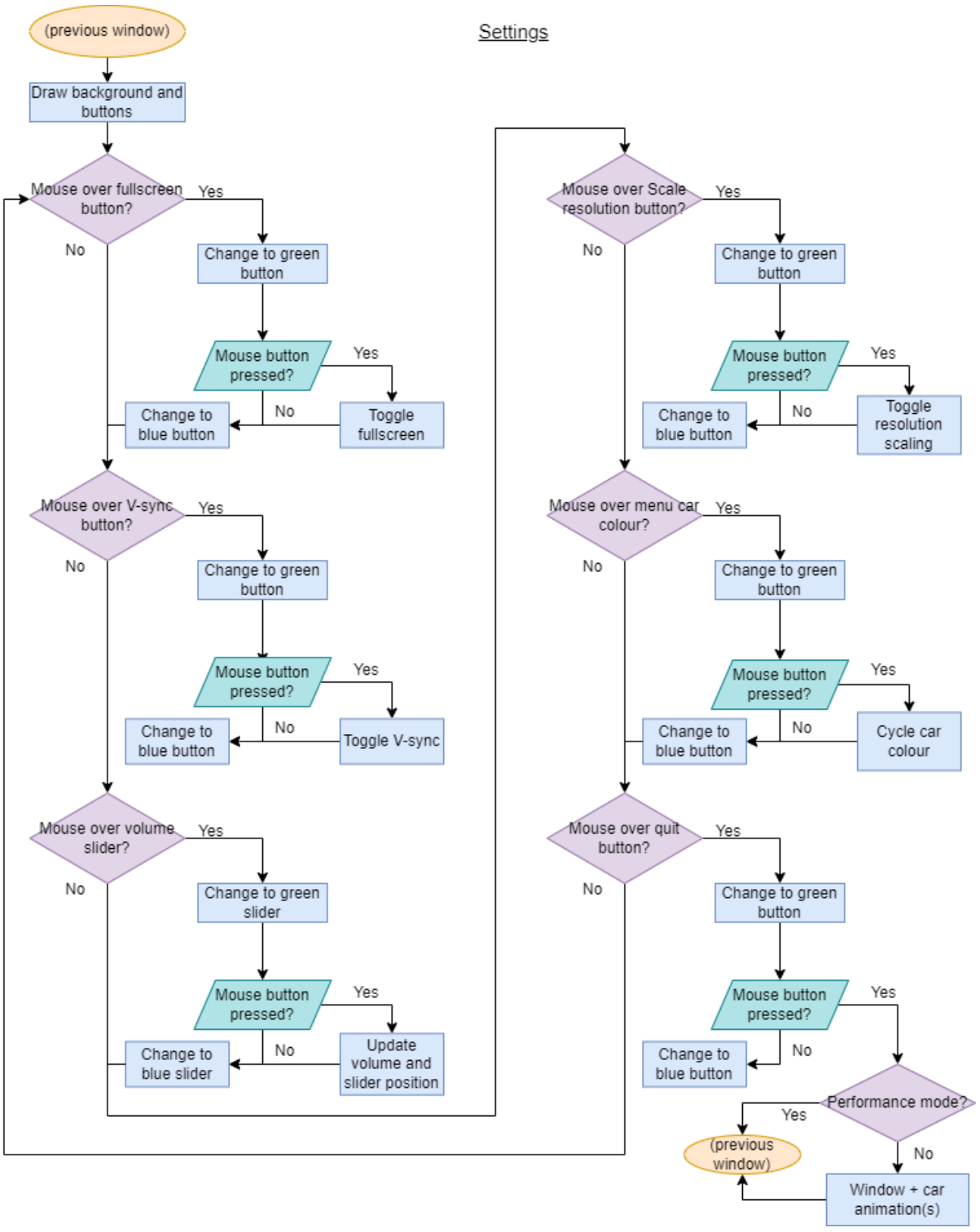
Settings

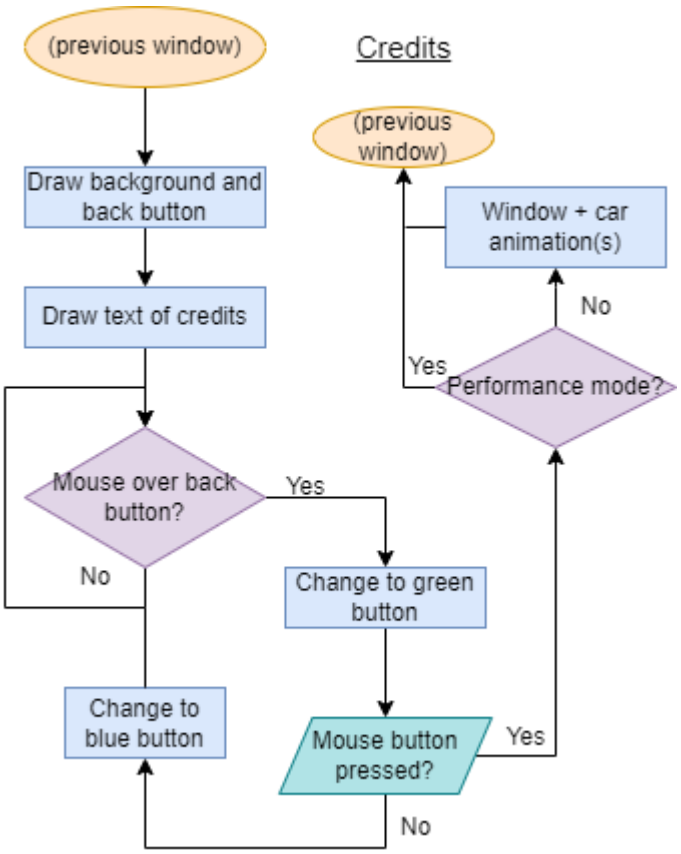
Flow Charts


Main gameplay
Race settings
p1 Veh
p1 & 2 Veh
Choose players
Choose Map
Leaderboard
Credits / licences
Main screen
Settings
Confirm quit
QUIT

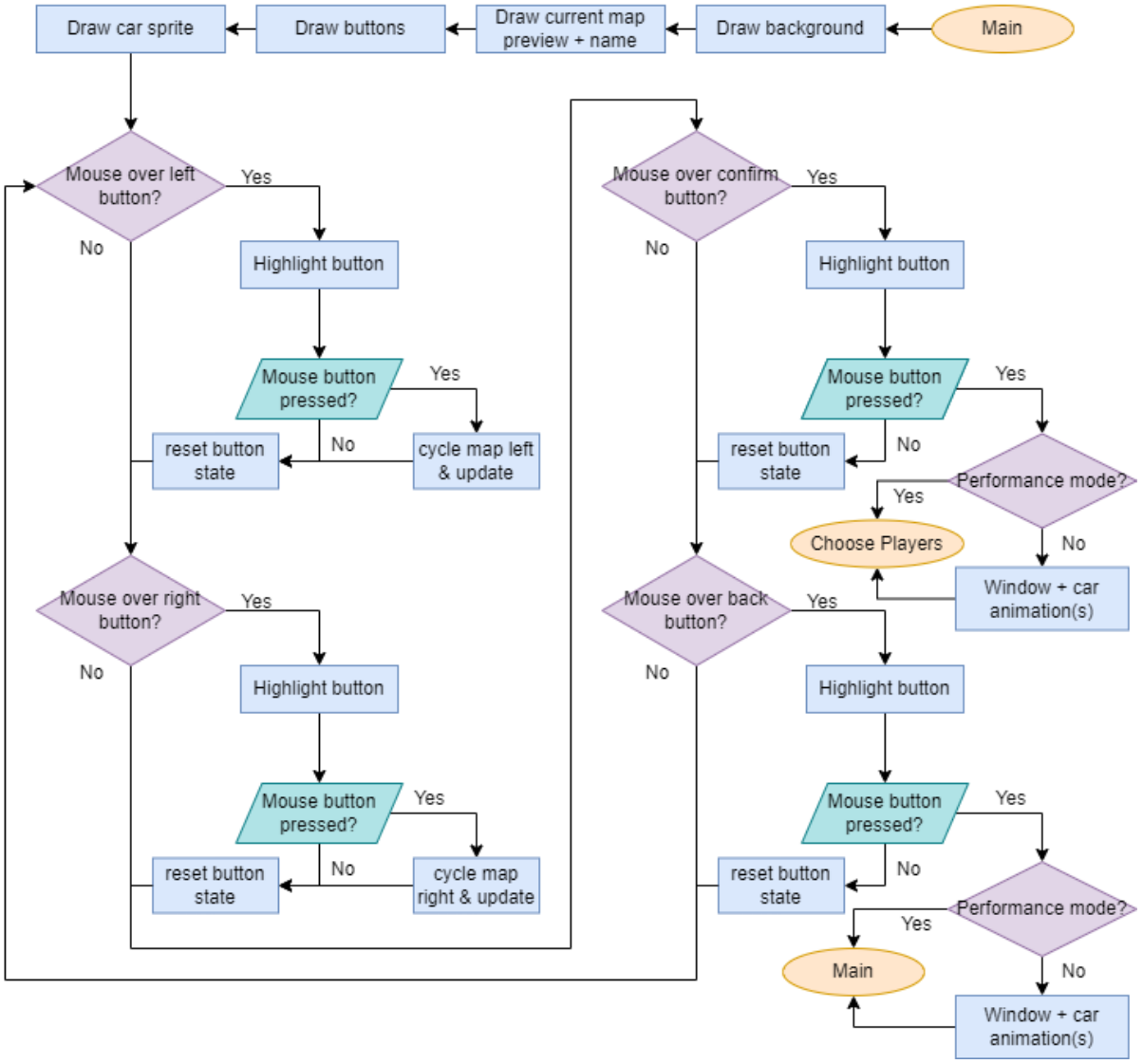

Main
Start
Display 720p?
Yes
No
Scale down to 720p
(Previous window)
Create Window + background
Window close button pressed?
Yes
No
Stop
Full-screen button pressed?
Yes
No
Change full-screen
Window lost focus?
Yes
No
Start sleep mode
Window gained focus?
Yes
No
Stop sleep mode
F11 key pressed?
Yes
No
Toggle full-screen
Mouse over start button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Performance mode?
Yes
No
Choose Map
Window + car animation(s)
Mouse over quit button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Performance mode?
Yes
No
Confirm Quit
Window + car animation(s)
Mouse over settings button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Performance mode?
Yes
No
Settings
Window + car animation(s)
Mouse over credits button?
Yes
No
Change to green button
Mouse button pressed?
Change to blue button
Update window
Sleep mode?
Yes
No
sleep(0.1)
sleep(0.5)
Performance mode?
Yes
No
Window + car animation(s)
Credits
Key
Program stop
Program start
Window link
Process / Function
For loop
Button Input
Decision / Query

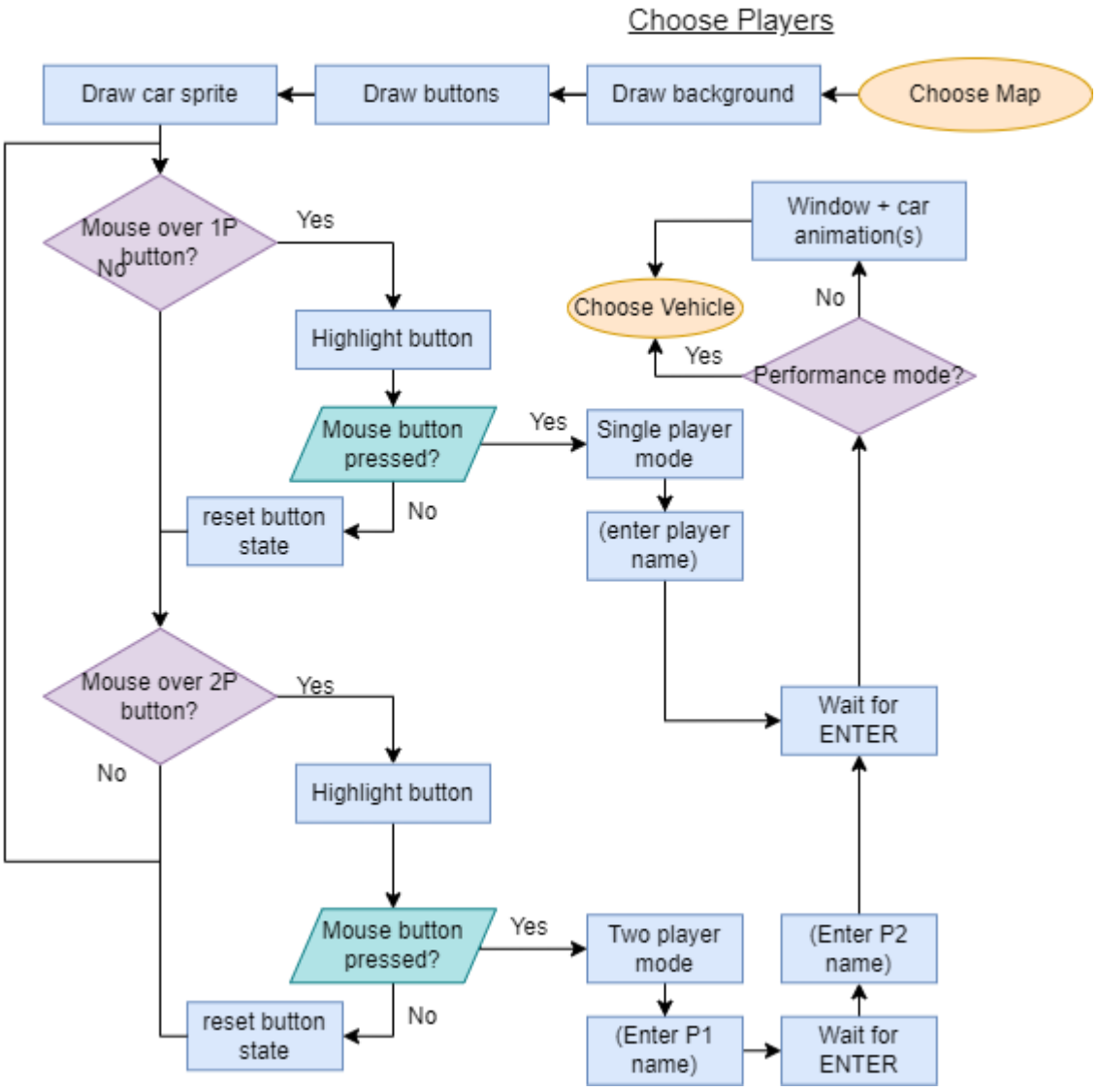

Confirm Quit
(previous window)
Draw background
Mouse over no button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Performance mode?
Yes
No
(previous window)
Window + car animation(s)
Mouse over yes button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Quit

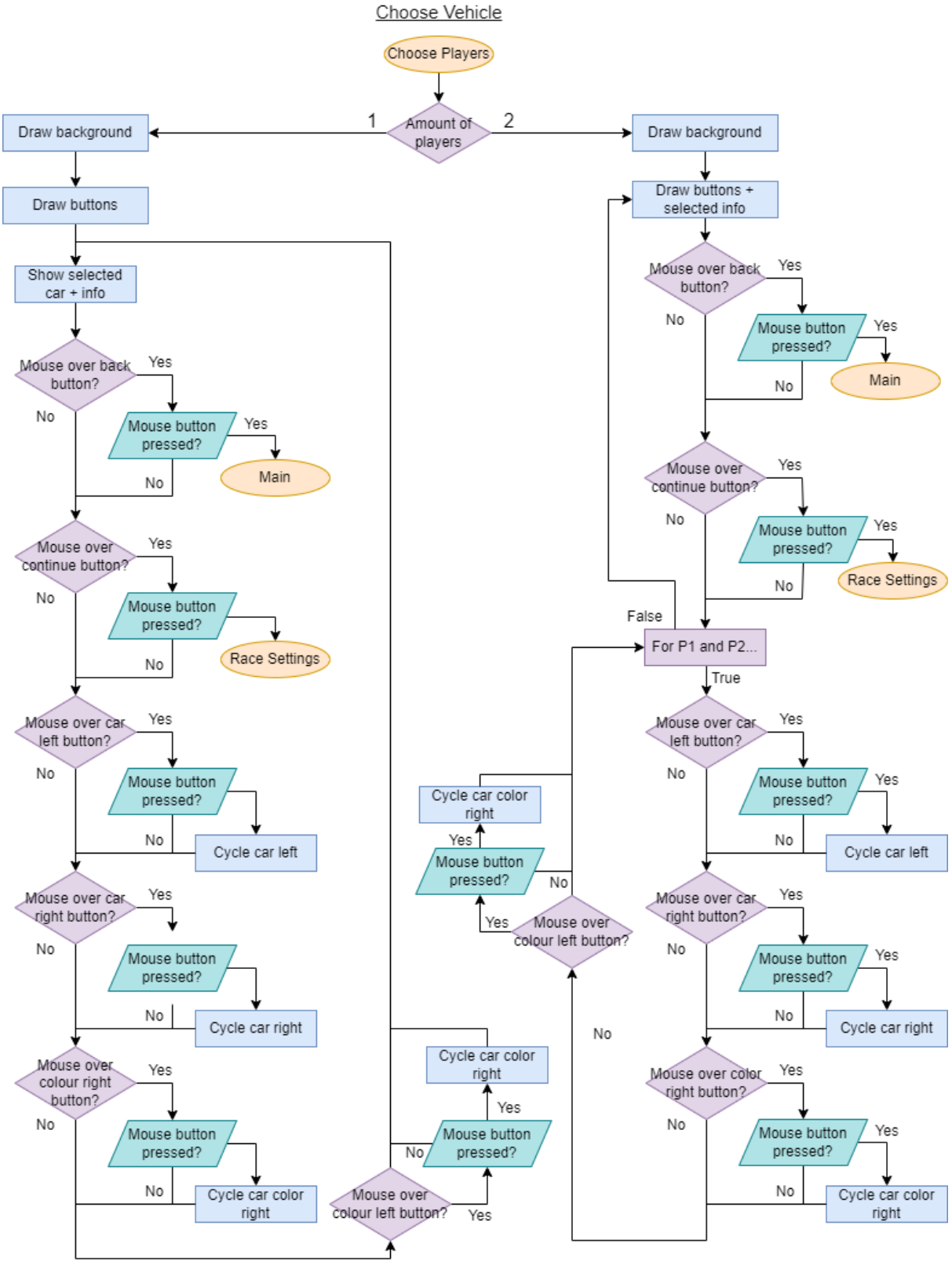

Settings
(previous window)
Draw background and buttons
Mouse over fullscreen button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Toggle fullscreen
Change to blue button
Mouse over V-sync button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Toggle V-sync
Change to blue button
Mouse over volume slider?
Yes
No
Change to green slider
Mouse button pressed?
Yes
No
Update volume and slider position
Change to blue slider
Mouse over Scale resolution button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Toggle resolution scaling
Change to blue button
Mouse over menu car colour?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Cycle car colour
Change to blue button
Mouse over quit button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Performance mode?
Yes
No
(previous window)
Window + car animation(s)
Change to blue button

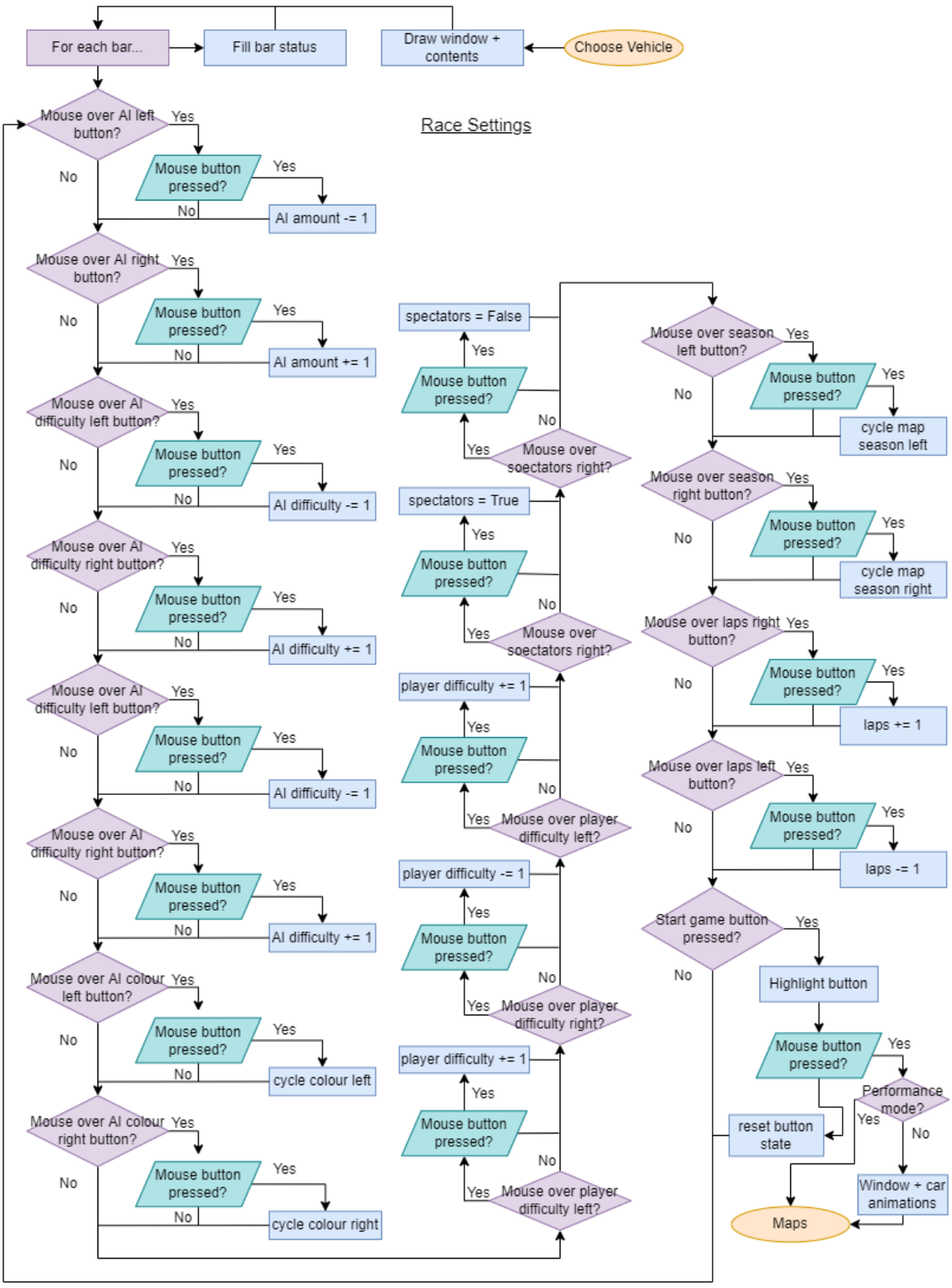

Credits
(previous window)
Draw background and back button
Draw text of credits
Mouse over back button?
Yes
No
Change to green button
Change to blue button
Mouse button pressed?
Yes
No
(previous window)
Window + car animation(s)
No
Yes
Performance mode?

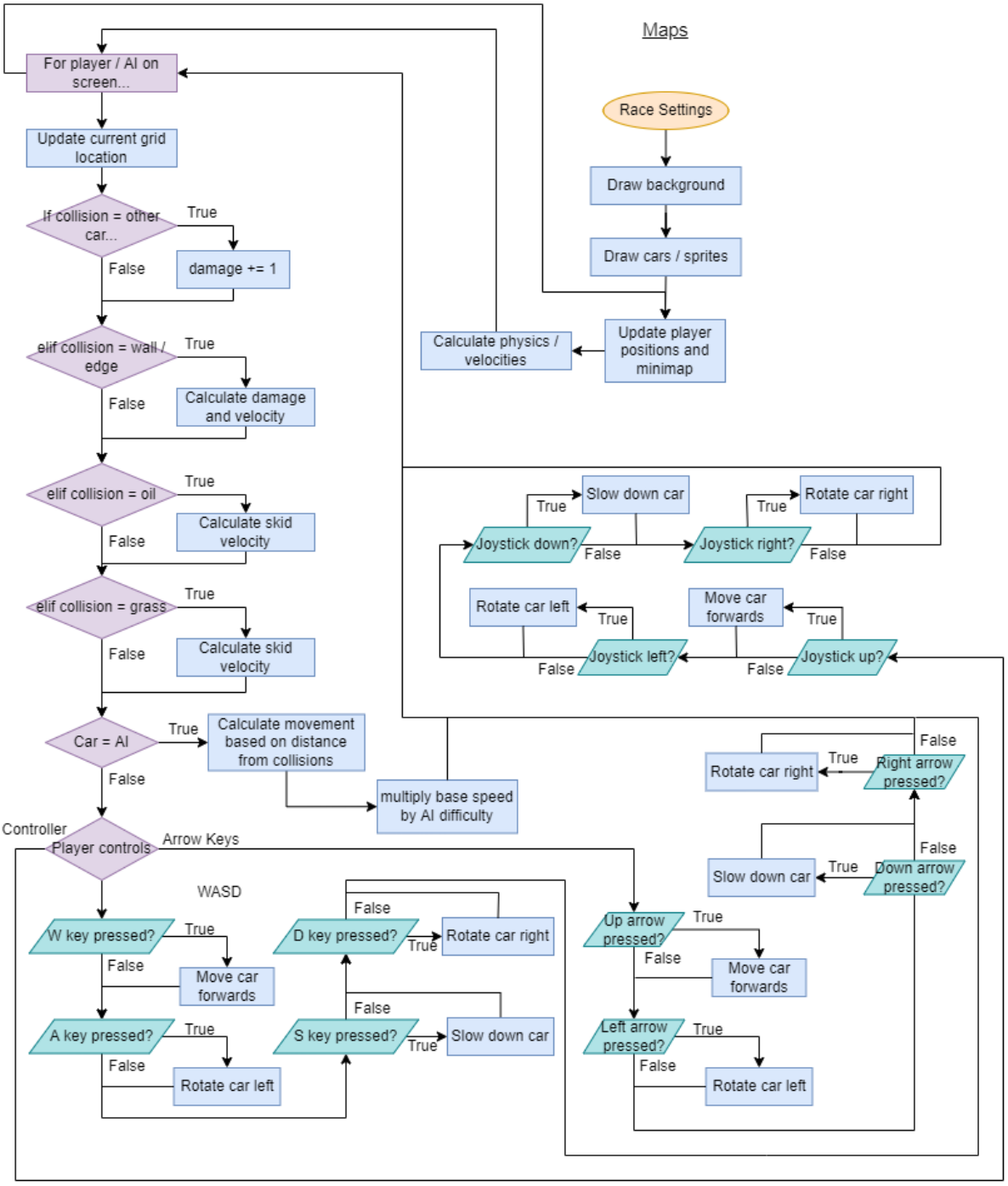
Choose Map

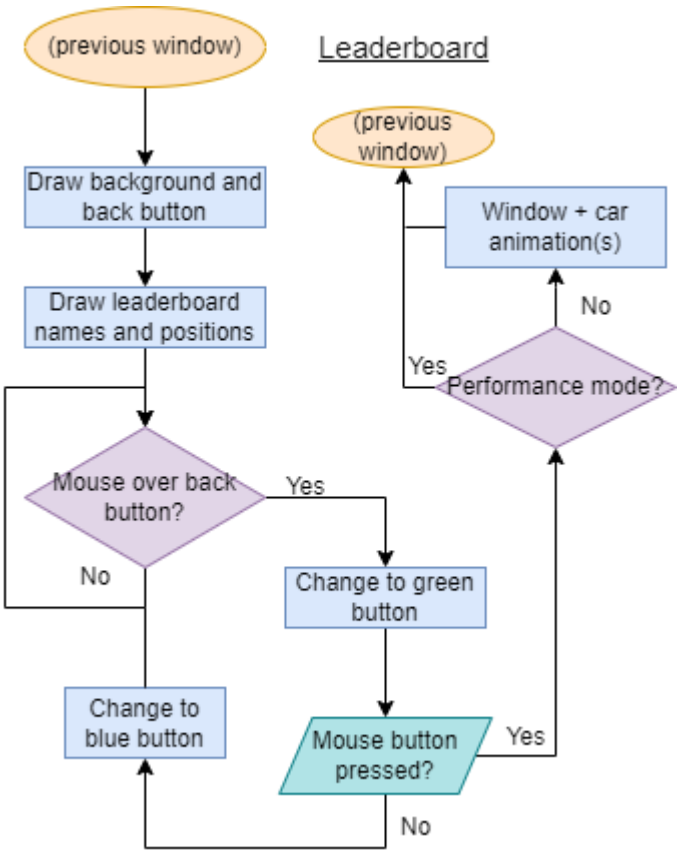

Draw car sprite
Draw buttons
Draw current map preview + name
Draw background
Main
Mouse over left button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
cycle map left & update
Mouse over right button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
cycle map right & update
Mouse over confirm button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
Performance mode?
Yes
No
Choose Players
Window + car animation(s)
Mouse over back button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
Performance mode?
Yes
No
Main
Window + car animation(s)


Choose Players
Draw car sprite
Draw buttons
Draw background
Choose Map
Mouse over 1P button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
Single player mode
(enter player name)
Mouse over 2P button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
Two player mode
(Enter P1 name)
(Enter P2 name)
Wait for ENTER
Wait for ENTER
Window + car animation(s)
Choose Vehicle
Yes
Performance mode?
No


Choose Vehicle
Choose Players
Amount of players
1
2
Draw background
Draw buttons
Show selected car + info
Mouse over back button?
Yes
No
Mouse button pressed?
Main
Mouse over continue button?
Race Settings
Mouse over car left button?
Cycle car left
Mouse over car right button?
Cycle car right
Mouse over colour right button?
Cycle car color right
Mouse over colour left button?
Cycle car color right
Draw background
Draw buttons + selected info
Mouse over back button?
Mouse button pressed?
Main
Mouse over continue button?
Race Settings
False
For P1 and P2...
True
Mouse over car left button?
Cycle car left
Mouse over car right button?
Cycle car right
Mouse over color right button?
Cycle car color right
Mouse over colour left button?
Cycle car color right


Race Settings
For each bar...
Fill bar status
Draw window + contents
Choose Vehicle
Mouse over AI left button?
Mouse button pressed?
AI amount -= 1
Mouse over AI right button?
AI amount += 1
Mouse over AI difficulty left button?
AI difficulty -= 1
Mouse over AI difficulty right button?
AI difficulty += 1
Mouse over AI colour left button?
cycle colour left
Mouse over AI colour right button?
cycle colour right
spectators = False
Mouse over soectators right?
spectators = True
Mouse over soectators right?
player difficulty += 1
Mouse over player difficulty left?
player difficulty -= 1
Mouse over player difficulty right?
player difficulty += 1
Mouse over player difficulty left?
Mouse over season left button?
cycle map season left
Mouse over season right button?
cycle map season right
Mouse over laps right button?
laps += 1
Mouse over laps left button?
laps -= 1
Start game button pressed?
Highlight button
Performance mode?
reset button state
Window + car animations
Maps
Yes
No


Maps
Race Settings
Draw background
Draw cars / sprites
Update player positions and minimap
Calculate physics / velocities
For player / AI on screen...
Update current grid location
If collision = other car...
True
damage += 1
False
elif collision = wall / edge
True
Calculate damage and velocity
False
elif collision = oil
True
Calculate skid velocity
False
elif collision = grass
True
Calculate skid velocity
False
Car = AI
True
Calculate movement based on distance from collisions
multiply base speed by AI difficulty
False
Controller
Player controls
Arrow Keys
WASD
W key pressed?
True
Move car forwards
False
A key pressed?
True
Rotate car left
False
D key pressed?
True
Rotate car right
False
S key pressed?
True
Slow down car
False
Joystick down?
True
Slow down car
False
Joystick right?
True
Rotate car right
False
Rotate car left
True
Joystick left?
False
Move car forwards
True
Joystick up?
False
Up arrow pressed?
True
Move car forwards
False
Left arrow pressed?
True
Rotate car left
False
Rotate car right
True
Right arrow pressed?
False
Slow down car
True
Down arrow pressed?
False



Sub - Program Explanations

Screen Transition Animation(s)

All screens share the same style, and have a dirt background with a sandy beach wrapping around all of the sides of the screen. Each button and interface will be overlaid on top of this base background for each window. The sides of each window will also have gaps in the sand to show the user that there is a window for them to go to using the buttons on screen. If the user presses a button on screen that leads to another window (eg. the Quit button on the main screen) then there will be an animation that plays. Said animation will first rotate the car in the centre of the screen towards the gap in the sand where the next window will be. Then the entire window will move off-screen whilst the car stays in the centre of the screen. The car is used to keep the users' focus on the middle of the screen while everything is moving as without this car to focus on then the users end up feeling nauseated and puzzled on what to look at. The car also serves as a reason to have the matching gaps in the edges of sand that correspond to the other windows. The gap in the sand ends up leaving a nice transition between the windows with the car being the main focus.

Buttons

All of the buttons will work the same. They are made up of two end pieces of track and filled in with as many as needed straight pieces. The text for the name of the button is then overlaid on top of the road pieces… creating a button! When the player's mouse hovers over the button the button will turn green to show the player that clicking while the button is green will result in the action of the button.

Main screen

The Main / home screen is the first thing that the user(s) see when they first open the game. It is important that this menu is pleasing to look at and is also easy to use in order to give a good first impression to the user. The main window actually starts with the system first booting up, and checks what resolution the game should play at - if the screen is 1080p then it is left alone… however if the window is any other resolution then the window is scaled up / down until it fits the screen properly. The window is then created and filled with the appropriate background that is generated using the assets for the game. Firstly, the game checks if the window close button has been pressed and acts accordingly - without this the window would not close even if the user pressed the close button. The code then checks the next window button, the fullscreen or 'maximise' button and acts accordingly to make the button work as expected. The next two decisions are related to power and performance saving. As when the user is focused on the window then the code will refresh at 144hz for smooth gameplay, however this leads to a lot of performance requirements for the hardware so when the user loses focus of the game, the refresh rate of the window drops significantly to around 2hz, meaning that there is more hardware performance left for the other processes currently running on the system. After this is checked, code then checks the first key - the F11 key is used in almost all applications as a full screen shortcut, and so if the key is pressed the game will toggle full screen. The main screen has 4 buttons in total. The first one that the game checks is the start button, and the method is the same for all buttons in the menus. For every button, the first decision is whether the mouse is over the button itself. If the mouse is over said button… then the button will be replaced from blue road to green road to indicate that the mouse is over the button. Once the mouse is pressed over the button then the appropriate action will be carried out. If the user does not press the button and moves the mouse off the button… the button will return to the previous blue style to indicate to the user that the mouse is no longer active and clicking the mouse button will not result in any actions. For the main window, if the start button is pressed then the game will animate to the start window (but will only animate if performance mode is off). If the quit button is pressed then the game will animate to the confirm quit window. Similarly, if the settings button is pressed then the game will move to the settings window and if the credits button is pressed then the game will move to the credits window and will finally update everything on the screen that has changed since the last frame. The last action of every window before it loops is to check if the window has focus or not - this is so that the refresh rate can drop if the user loses focus as stated in the previous explanation.

Confirm Quit

The confirm quit window is very simple. It contains two buttons to confirm whether or not the user actually wishes to quit the game or if they accidentally pressed the quit button… and is simply a convenience for the user to give them an option to go back to the screen that they were on before they pressed the quit button. If the first button is pressed (yes button) then the game will exit and close, whereas if the no button is pressed then the game will move back to the window that the user previously came from.

Settings

The settings window starts by drawing the background and buttons on the screen. This window contains 6 buttons that all have different uses. The first button is a button to toggle between fullscreen and windowed modes, and displays as a 'fullscreen' button. The second button appears as a 'V-sync' button, and toggles between V-sync on and off. The third interaction is a little different… as it is a slider for the volume of the game. This slider will have an underlined title of 'Volume' and will contain a grey / white style, including a circle on the end of the slider for the user to interact with. The entire time that the user holds the mouse button and is on top of this circle then the slider will move to the position that the mouse is at. The fourth button is so that the user can toggle between having the window automatically scale to their screen size or manual 720p / 1080p. The button name will change to the current state of the resolution setting, and will cycle as follows - Auto, 1080p, 720p, 480p. However, users will be warned below the button whenever they are not using Auto resolution that using fullscreen with the incorrect resolution could cause compatibility issues depending on the graphics driver and OS.

Credits

The credits window is the most simple of all the windows. This screen only has one button and just shows the user the text on screen containing the credits for the game. The single button is the back button and allows the user to go back to the window that they were previously on.

Choose Map

This window contains 4 buttons and a preview of the map that is currently selected. There will be side buttons to switch between the maps, and the other 2 buttons are a back button and continue / select button for the user to navigate to and from the window. The main difference between this window and the others is that the car that people focus on will not stay in the middle… and will instead stay underneath the map preview and then drive around the map when the user changes to another window.

Choose Players

This window is very simple and similar to the confirm quit screen… where the user(s) have 2 options - one for 1p and another for 2 players. There will not however be a back button on this window since the user has already chosen a map and there will be a lot of software setup based on if the user chooses 1p or 2p, and having a quit button would cause issues.

Choose Vehicle(s)

This window is by far the most complicated of them all… since this window, depending if it is one or two players, will have a visualisation of the selected car, and a lot of text and bars showing the user of the stats of their selected vehicle. There will be arrows to cycle their vehicle and colour right / left adding up to a maximum of 10 buttons on this screen. At the bottom of each screen there will be a confirm button that will turn green and stay green once the user has clicked on it. The game will only progress to the next window once both buttons are green.

Race settings

Even though this screen will have 15 buttons… Lots of the buttons are the same. There will only be 6 options for the user to change. There are more buttons as there needs to be a way for the user to increase and decrease the value that they want to change. The values for the race that the users can change are: AI amount, AI difficulty, AI colour, Player difficulty, Spectators and map season. The most important one is the amount of laps which the user can also change. On this screen the user is also informed of the previously chosen options such as the map and player 1 / 2 vehicle(s). In the bottom middle of the screen there will be a large ‘Start Game’ button which will start the actual gameplay.

Map(s) / Gameplay

At the start of this window… the background and sprites are first drawn. These will stay on the screen but be moved around therefore they only need to be drawn once. The loop of the window that produces each frame will start and begin by updating the screen to actually show what has happened from the previous frame. The game will then calculate any pending velocities and physics that haven’t been carried out yet. The next process is iterative and will be carried out for each vehicle that is on the screen including the AI. The first thing in the iteration is to find the grid location of where the car is to minimize screen updates and only update the area around that vehicle. This way, the screen does not have to update parts that have not changed since the last frame and is very efficient. The users cannot see this grid as it is only a method to update the screen. The game carries out all of the different collision checks such as walls, other cars, oil and grass. The next thing is to check if the current car is an AI or a player. If the vehicle is an AI then the movements are based on the distance from nearby collisions and the walls of the road. There will be a random input applied to this to ensure that the vehicle stays on the road but does not just drive down the middle all the time. However if the current vehicle is a player then the game will check the appropriate keyboard / controller inputs before joining back to the iteration loop. Once this loop has been completed for each vehicle then the minimap and player positions are updated before the next loop starts and updates the screen. The update is at the beginning of each loop so that the background is drawn as soon as possible when the player first loads in, to avoid the possibility of a few ms of black screen.

Leaderboard

This screen is only to show the players and AI the position that they came at the end of the race. All of the said information will be shown through text, and the only buttons will be a back button to go back to the main screen.

Alpha + Beta Testing

Main

No.	Feature	Explanation
1	Quit button	Check that the button navigates to confirm quit window and highlights on mouseover
2	Start button	To check that the button navigates to choose map window and updates properly
3	Settings button	Check that button navigates to settings window and updates properly
4	Credits button	To check that the button navigates to credits window and updates properly
5	Window + car animations	Ensure the smooth transition of windows and proper rotation of car during animation
6	Mouse cursor	For all windows where the cursor is shown, exchange the usual cursor for a custom one

Choose Map

No.	Feature	Explanation
7	Confirm button	Check that the button navigates to choose players window
8	Back button	Check that the button navigates to the main window
9	Map left button	Ensure that the button cycles the map preview left once

10	Map right button	Ensure that the button cycles the map preview right once
11	Map name and preview	To check that the preview and name match, and that the preview shows up properly
12	Window + car animations	Make sure that the car stays in the correct spot and that the animations are smooth

Choose players

No.	Feature	Explanation
13	1 Player button	Check that the button links to the single player choose vehicle window
14	2 Player button	Check that the button links to the two player choose vehicle window
15	P1 Name	Make sure that the user can only enter one name at a time and that they can only enter appropriate values (eg. no numbers or symbols)
16	P2 Name	Only enter P2 name after P1 name and have pressed ENTER
17	Window + car animations	Make sure that the car rotates properly and the window animation is smooth

.Choose Vehicle (1P + 2P)

No.	Feature	Explanation
18	Car name	Make sure that the name always matches the current car

19	Car preview	Make sure that the proper car and colour are showing according to the player's settings
20	Speed, cornering and durability text	Ensure that text stays in proper position and is always on screen
21	Bar percentages	Check that the bar stats are properly filled in to tell the player the stats of that particular car, and that they change for each car
22	Car colour text	Make sure that the colour text always matches the colour of the car
23	Back button	Ensure that the back button links to the main window and that it updates properly
24	Confirm button	Ensure that the confirm button, if single player, links to the race settings window and updates properly. If there are two players… then the confirm button should not change the window until both are clicked. Once one of the buttons is pressed the controls are disabled and the confirm button will stay green.
25	Car left button	Make sure that the button properly cycles the car and all related info left once
26	Car right button	Make sure that the button properly cycles the car and all related info right once
27	Colour left button	Check that the car colour left button properly cycles the car's colour and text left once
28	Colour right button	Check that the car colour right button properly cycles the car's colour and text right once

29	Window + car animations	Make sure that the car properly animates during window animation and that all assets stay in the right position

Race Settings

No.	Feature	Explanation
30	Setting titles	Make sure that the titles are all in the correct position, size and underlined
31	Setting values	Ensure that the values are correct and correspond with any bars
32	Setting controls	Ensure that the left and right controls for each value properly cycle the values in the correct direction. Do not allow values to go out of range
33	Selected map and vehicle(s)	Make sure that the text for the current map, season and player vehicles is properly centred at the top of the screen
34	Start game button	Check that the start game button properly links to the correct map and updates correctly
35	Car animation	Make sure that the car animation goes off screen without the window moving, then fades to black or loading screen while the game loads.

Main gameplay

No.	Feature	Explanation
36	Map background	Make sure that the map background has been properly generated in the correct position and season
37	Player vehicles	Check that the player can control their vehicles using the appropriate controls, and that their car is the one that they selected
38	AI vehicles and control	Ensure that the AI works properly and manoeuvres the track(s) efficiently. Also check that the AI colour setting matches their vehicles
39	Powerups	Make sure that power ups only appear on the track and that they work properly
40	GUI	Ensure that the GUI is properly updated according to what is going on in the game, and includes visuals for player position, car integrity and current speed ect…
41	Start positions	Make sure that all of the AI and players correctly start in the positions, and the players start at the front if easy and work backwards depending on difficulty
42	Map crossovers	Ensure that the cars can only go straight and cannot cut off a part of the track
43	Car damage	Check that the car stats are taken into account when the car takes damage and that the damage is shown through black lines on the car as well as a GUI feature

44	Car repair	Make sure that the car damage is repaired gradually for every repair powerup that is collected by the car and remove any damage effects that were applied to the car when it was damaged.
45	Mouse cursor	For the entirety of the main gameplay… hide the cursor as it is not needed. This is the only window where the cursor will not be visible

Leaderboard

No.	Feature	Explanation
46	Quit button	Make sure that the quit button links back to the confirm quit screen, and that it update properly
47	Settings button	Check that the settings button links to the settings window and that it updates according to the mouse position
48	Main menu button	Make sure that the main menu button properly updates and that the button links to the main window
49	Leaderboard	Properly show the leaderboard in order and centre of the screen, and also include little icons of the car that the player was in for convenience
50	Map settings	Also include a small section to the right of the leaderboard that shows the map settings that were used for that race

51	Window animations	There is no window car at this point, so just have the screen either change to the next window or have no animation

Variables

Name	Scope	Type	Default Value	Explanation
RED	Global	Constant tuple	255, 0, 0	RGB values for red
WHITE	Global	Constant tuple	255, 255, 255	RGB values for white
V_LIGHT_GREY	Global	Constant tuple	200, 200, 200	RGB values for very light grey
LIGHT_GREY	Global	Constant tuple	170, 170, 170	RGB values for light grey
GREY	Global	Constant tuple	100, 100, 100	RGB values for grey
BLACK	Global	Constant tuple	0, 0, 0	RGB values for black
RED_CAR	Global	Constant tuple	232, 106, 23	RGB values for red car
YELLOW_CAR	Global	Constant tuple	255, 204, 0	RGB values for yellow car
GREEN_CAR	Global	Constant tuple	57, 194, 114	RGB values for green car
BLUE_CAR	Global	Constant tuple	47, 149. 208	RGB values for blue car
BLACK_CAR	Global	Constant tuple	93, 91, 91	RGB values for black car
Debug	Global	Boolean	False	Activate debug mode if true
Force_resolution	Global	Boolean	False	Debugging disables auto resolution

Screen	Global	Integer	0	Selects the default display to load
menu_animation	Global	Boolean	True	Enable/Disable animations
Mute_volume	Global	Boolean	False	Set if any sound should be played
Music_volume	Global	Int	0.2	Volume of music channel
Sfx_volume	Global	Int	0.25	Volume of sound effects channel
Desktop_info	Global	Tuple List	Pixel width, height	Figure out screen resolution and scale game accordingly
Display	Global	Class	Pygame initialisation	Stores pygame data for window settings
Display_resolution	Global	Tuple	Display width, height	Holds resolution data for display
WIDTH	Global	Constant	Window pixel width	Stores the window width
HEIGHT	Global	Constant	Window pixel height	Stores the window height
CENTRE	Global	Constant tuple	Centre pixel	Used as a reference point
Window	Global	Pygame surface	(pygame surface)	Game render surface
Window_resolution	Global	Constant Tuple	Window width, height	Holds resolution size for Window
Display_scaling	Global	Boolean	False	Set scaling between render and screen

icon	Global	Image	Window icon	Loads icon image for window
all_sprites	Global	List	Car sprites list	Groups all car sprites together
tile_scale	Global	Tuple	Width scale, height scale	Scale for adding tile assets
menu_scroll_speed	Global	Int	20	Menu animation speed
menu_car_speed	Global	Int	6	Speed of car rotation during animations
screen_updates	Global	Tuple list	[ ]	Regions of screen to be updated between frames
loaded_assets	Global	List	[ ]	Holds list of all loaded assets in memory
loaded_sounds	Global	List	[ ]	Holds list of all loaded sounds in memory
Player_amount	Global	Int	0	Amount of players
P1_veh	Global	Tuple	Vehicle, colour	Stores player 1 vehicle data
P2_veh	Global	Tuple	Vehicle, colour	Stores player 2 vehicle data
P1_name	Global	String	Player name	Stores player 1 name
P2_name	Global	String	Player name	Stores player 2 name
Map	Global	Int	0	Selected map int
AI_amount	Global	Int	0	Selected amount of AI

FPS	Global	Constant	Frames per Second	Controls the speed of the game
origin_pos	Local, Car	Int	Start position	Saves the original start position
origin_rotation	Local, Car	Int	Start rotation	Stores the original start rotation
image_dir	Local, Car	Str	Directory path	Stores the directory path of the image
image	Local, Car	Image	Car asset	Load car asset as image
size	Local, Car	Tuple	71, 131	Size of car asset for sprite
rect	Local, Car	Rect	Image Rect	Rect of car image
pos_x	Local, Car	Int	Centre x	Starting x position of sprite
pos_y	Local, Car	Int	Centre y	Starting y position of sprite
rotation	Local, Car	Int	0	Angle of rotation of sprite
rect	Local, Car	Tuple	Size of asset	Pixel width and height of asset
scaled_rect	Local, Car	Tuple	Scaled size of asset	Pixel width, height
x	Local, main_window, confirm_quit_window, credits_window	Int	pos_x	X position of asset on screen

y	Local, main_window, confirm_quit_window, credits_window	Int	pos_y	Y position of asset on screen
x	Local, tile	Int	Scaled x position	X position for grid tile system
y	Local, tile	Int	Scaled y position	Y position for grid tile system
img	Local, tile	string	Directory of asset	Stores path to requested asset
image	Local, tile	Image	Asset	Load asset as surface
loaded	Local, tile, FOR	Surface	Loaded asset	Stores loaded asset for displaying
raw_surf	Local, tile	Surface	Original Asset	Saves original state of asset for future
font	Local, draw_text	Font file	Text font	Formats font drawn on screen
render	Local, draw_text	Font rendering	Render of font	Renders style of font to draw on screen
loaded	Local, draw_asset, FOR	Surface	Asset	Stores loaded asset as surface
raw_surf	Local, draw_asset	Surface	Asset	Loads default asset state
surf	Local, draw_asset	Surface	Asset	Stores transformed asset
img	Local, menu_background	Image	Background image	Stores background image

image	Local, menu_background, (FOR loop)	Image	Asset	Stores asset to paste onto background image
playing	Local, menu_music	Boolean	(status of music)	True if music is playing, False if not
audio	Local, play_sound, (FOR loop)	Sound	(loaded sound)	Loaded sound name and file
sound	Local, play_sound	Sound	Load sound	Load sound from path
boot_1	Local, play_sound	Sound	Load sound	Load sound from path
boot_2	Local, play_sound	Sound	Load sound	Load sound from path
update	Local, update_screen, (FOR loop)	Tuple	Rectangle of area to update	Used to update specific parts of screen
pos	Local, get_mouse_pos	Tuple	Position of mouse	Stores mouse position on display
clock	Local, main	Clock (pygame)	Pygame clock	Used for screen and game updates
window_sleep	Local, main	Boolean	False	Keeps track of window focus for slowing updates
menu_loop	Local, main	Boolean	True	True for time user is in menus
current_window	Local, main	String	main menu	Holds currently shown window
prev_window	Local, main	String	None	Stores previous window
car	Global	Class	Car class	Functions to manipulate car asset

bg	Local, main	Image	Background image	Stores current background image
new_bg	Local, main	Image	Background image	Stores new background for animations
intro_bg	Local, main	Image	Background image	Stores blank background for intro
event	Local, main, (FOR loop)	Pygame event	Pygame event	Used to check events like window focus and full screen button
pressed_keys	Local, main	List	Pressed keyboard buttons	Used to check F11 for full screen
mouse_pos	Local, main	Tuple	Mouse position	X and Y coordinates of mouse position
pos_x	Local, main	Int	X pos	X pos for button
pos_y	Local, main	Int	Y pos	Y pos for button
buttons	Local, main	List	Mouse button values	Used for mouse clicks
button	Local, main, (FOR loop)	Boolean	Mouse button value	Used to check mouse buttons
rotation	Local, main, (FOR loop)	Int	Rotation of degrees	Used to rotate car asset
offset_y	Local, main, (FOR loop)	Int	Y pixel offset of window	Used for window animations
current_window	Local, main	Image	New background image	Used for window animations

Functions

Name	Inputs	Outputs	Explanation
(Car) __init__	None	Creates variables for class	Used to initialise variables for Car class
(Car) rotate	degree	Rotation of car asset	Used to rotate car asset
(Car) move	x, y	Movement of car asset	Used to move car around and off screen
(Car) draw	update	Draw car on screen	Draw car on screen
main_screen	curr_bg, pad_x, pad_y	Draw main window on screen	Draw all assets for main window on screen
confirm_quit_window	curr_bg, pad_x, pad_y	Draw confirm quit window on screen	Draw all assets for confirm quit window on screen
credits_window	Curr_bg, pad_x, pad_y	Draw credits window on screen	Draw all assets for credits on window on screen
tile	X, y, material, ver, grid, scale	Draw tile ground assets	Draw tile assets on screen
convert_image	img	Pygame image	Converts image formats between PIL and Pygame
draw_text	X, y, text, font_type, colour, size	Text on screen	Draws text on screen
draw_asset	Img, pos, scale	Asset on screen	Draws given asset on screen
menu_background	Top, right, bottom, left	Background image	Stitches multiple assets into one background
menu_music	None	Background music	Plays and loops random background tracks

play_sound	event	Event sound	Plays sound effect according to the event type
scale_to_display	pos	scaled_pos	Scales window position to display position
scale_to_window	pos	scaled_pos	Scales display position to window position
scale_rect	rect, x, y	rect	Scales window rect to display rect
add_screen_update	rect, x, y	None	Scales and adds rect to screen_updates to be updated
update_screen	None	Updates screen	Compiles changed areas of screen and updates specific parts only
get_mouse_pos	None	pos	Scales mouse position to window
game	clock	Main Gameplay	Every gameplay process in within this function
main	None	Main Menus	Every menu process is within this function

Classes

Name	Functions	Explanation
MenuCar	__init__, rotate, move, draw	Allows creation, movement and updates of car sprite in menus
Car	__init__, set_move_speed, set_rotation_speed, set_controls, draw_name, check_laps, check_checkpoints, clear_checkpoints, check_track_collisions, check_car_collisions, move, rotate, power_up, check_inputs, draw, update	Pygame sprite that the player can control
NPCCar	__init__, set_move_speed, set_rotation_speed, draw_name, check_laps, check_checkpoints, clear_checkpoints, get_path, check_track_collisions, check_car_collision, move, rotate, reset_to_checkpoint, follow_path, follow_diversion, power_up, draw, update	Pygame sprite that is automatically controlled

Imports

Name	Functions	Explanation
math	Ceil, floor	Used for rounding up and down
image_loader	(every asset)	Used to find directory names for requested asset
font_loader	(all fonts)	Used to find directory path to requested font file
audio_loader	(every sound file)	Used to find directory path to requested audio file

random	randint	Used to generate a random integer
time	sleep	Used for idle processing
pygame	(entire class)	Game engine
json	(entire class)	Used to save and load settings.json file

Libraries

Audio_loader.py

Variable Names	Function Names
assets	menu_track(track_no)
effects	scream(humanoid, ver)
music	explosion(length, ver, hard=False)
menu_music	alarm(ver, low_health=False)
screams	bleep(ver)
explosions	button(ver)
exp_cluster	coin(ver, amount)
exp_double	damage(ver)
exp_long	fanfare(ver)
exp_medium	high_pitch(ver)
exp_odd	impact(ver)
exp_short	interaction(ver)
exp_shortest	menu(ver)
exp_various	misc(sound_name)
general	negative(category, ver)
alarms	neutral(ver)
bleeps	pause_sound(ver, out=False)
buttons	positive(ver)
coins	door(ver)

damages	falling(ver)
fanfares	footstep(ver, loop=False, fast=False)
high_pitches	jump(ver, landing=False)
impacts	ladder(ver, loop=False)
interactions	portal(ver)
menu_sounds	stair(ver, loop=False)
misc_sounds	vehicle(veh, ver)
negative_sounds	cannon(ver)
neutral_sounds	grenade(ver)
pause_sounds	missile_launch()
positive_sounds	laser(ver)
movement	machine_gun(ver)
doors	melee(weapon, ver)
falling_sounds	
footsteps	
jumps	
ladders	
portals	
stairs	
vehicles	
weapons	
cannons	
grenades	
lasers	
machine_guns	
melees	
out_of_ammo	
shotguns	
single_shots	

Font_loader.py

Variable Names	Function Names
assets	load(bold=False, italic=False, bar=False, outline=False, three_d=False)

Image_loader.py

Variable Names	Function Names
assets	icon()
cars	car(colour, ver, small=False)
characters	bike(colour)
motorcycles	char(colour, racer=False)
objects	arrow(colour)
powerups	barrel(colour, lying=False)
tiles	barrier(colour, race=False)
	cone(lying=False)
	light(ver)
	oil()
	rock(ver)
	skid_mark(ver, long=False)
	tent(colour, race=False)
	tire(colour, filled=False)
	tree(small=False)
	tribune(striped=False)
	power_up(ver, active=True)
	tile(material, ver)
	error()

Map_loader.py

Variable Names	Function Names
bg_layer	racetrack(layer)
obj_layer	
track_layer	
maps_dir	
racetrack_dir	

Development

Implementation & Improvements

Window Structure / Functions

The usability and appearance of the code is very important, as having all of the code laid out clearly makes it a lot easier to make changes or improvements later on. This is why each window has its own function that draws all of the assets needed to make up that window (this goes for every window) and all of these functions rest at the top of the code, in one place. The function draws the assets needed, then within the main loop each button check / action is carried out separately. The code to make the window buttons work are also all in the same place and grouped together within the main loop, to keep the code neat and organised so that if a developer needs to find a piece of code to fix something then they know where to look.

For window animations, the function to draw the assets can take a pad_x and pad_y argument which will offset the asset positions by the given amount based on their default position.

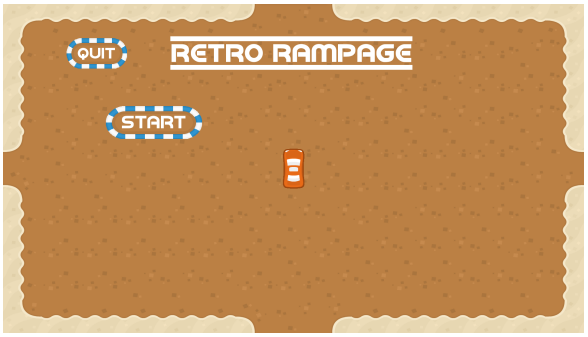
All animations for the windows are called from a single function too. animate_window(clock, window, new_window, bg, new_bg, car, direction) is called any time the programmer wants an animation to keep things simple and looking nice and clean. This also makes debugging and troubleshooting easier since all of the animation sequences are in one function. Within the animate_window() function there is another function within the MenuCar class used as car.animate(direction, bg, clock) and handles the rotation of the car separate from the movement of the window to keep things simple within the animate_window() function.

The windows themselves all contain different buttons and every button has a different action. Therefore, all of the button triggers and actions are separate from the window functions and are contained within the main loop to reduce the complexity as converting the button actions into functions would require all of the variables to be passed through or global. The main loop is also where the display updating and other tasks are carried out such as music.

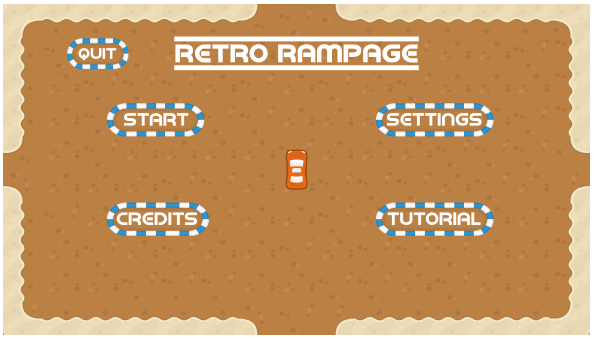
All windows follow this general rule, however all are slightly different and include specific modifications for them to work (eg. settings_window() and confirm_quit_window() can both take a surf argument for the pause menu during gameplay).

Main window

Old version



New version



Code

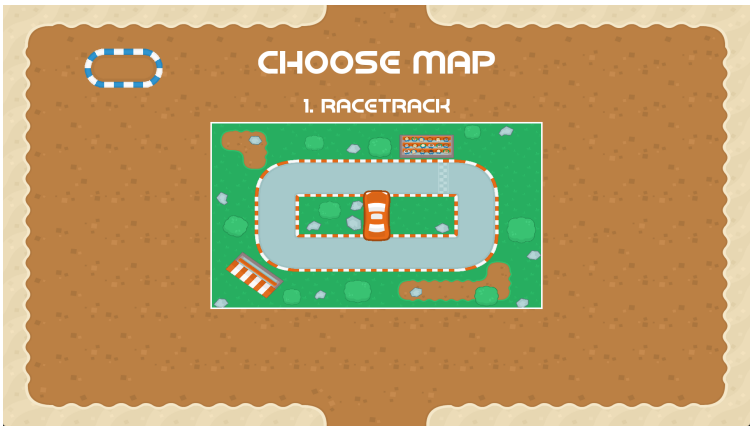
```
def main_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))  # Display the current bg

    x = pad_x + CENTRE[0]
    y = pad_y + 100
    draw_text(x, y, 'Retro Rampage', WHITE, 100, bold=True, bar=True)  # Title

    x = pad_x + 210
    y = pad_y + 112
    tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100))  # Quit button
    tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100))
    draw_text(x + 98, y + 21, 'Quit', WHITE, 60)

    x = pad_x + 340
    y = pad_y + 324
    tile(x, y, 'dirt road', 76, grid=False)  # Start button
    tile(x + 65, y, 'dirt road', 1, grid=False)
    tile(x + 190, y, 'dirt road', 60, grid=False)
    draw_text(x + 160, y + 20, 'Start', WHITE, 70)

    x = pad_x + 338
    y = pad_y + 648
    tile(x, y, 'dirt road', 76, grid=False)  # Credits button
    tile(x + 128, y, 'dirt road', 1, grid=False)
    tile(x + 205, y, 'dirt road', 60, grid=False)
    draw_text(x + 163, y + 20, 'Credits', WHITE, 70)

    x = pad_x + 1220
    y = pad_y + 324
    tile(x, y, 'dirt road', 76, grid=False)  # Settings button
    tile(x + 128, y, 'dirt road', 1, grid=False)
    tile(x + 256, y, 'dirt road', 60, grid=False)
    draw_text(x + 190, y + 20, 'Settings', WHITE, 70)
```

Description

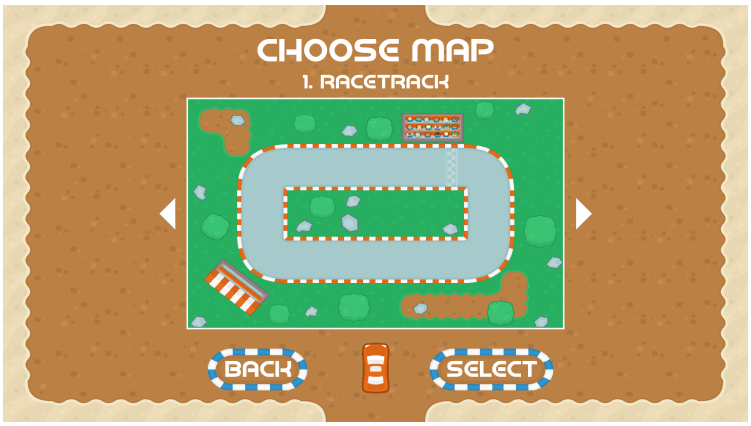
The main window does not have any special modifications from the explanation above for all the windows, and therefore follows the general rule exactly with its own function and button triggers separated but grouped together with the other windows.

Choose map window

Old version



New version



Code

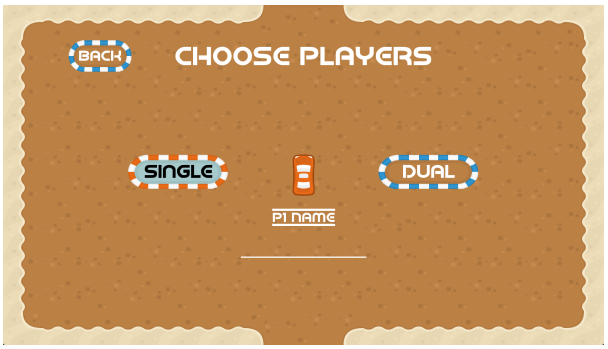
```
def choose_map_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))  # Display the current bg

    x = pad_x + CENTRE[0]
    y = pad_y + 70
    draw_text(x, y, 'Choose Map', WHITE, 100)  # Title

    x = pad_x + 528
    y = pad_y + 890
    tile(x, y, 'dirt road', 76, grid=False)  # Back button
    tile(x + 128, y, 'dirt road', 60, grid=False)
    draw_text(x + 130, y + 20, 'Back', WHITE, 70)

    x = pad_x + 1100
    y = pad_y + 890
    tile(x, y, 'dirt road', 76, grid=False)  # Select button
    tile(x + 65, y, 'dirt road', 1, grid=False)
    tile(x + 190, y, 'dirt road', 60, grid=False)
    draw_text(x + 160, y + 20, 'Select', WHITE, 70)

    # Draw map preview with border
    Window.blit(map_preview[1], (pad_x + map_preview_pos[0], pad_y + map_preview_pos[1]))
    pygame.draw.rect(Window, WHITE, (pad_x + map_preview_pos[0], pad_y + map_preview_pos[1], *map_preview_size), 4)

    x = pad_x + CENTRE[0]
    y = pad_y + (map_preview_pos[1] - 70)
    text = str(maps.map_index.index(Map) + 1) + '. ' + Map
    draw_text(x, y, text, WHITE, 60)  # Title

    draw_triangle((pad_x + (map_preview_pos[0] - 50),  # Map arrows
                   pad_y + (map_preview_pos[1] + map_preview_size[1] // 2)), 'left', width=40, height=80)
    draw_triangle((pad_x + (map_preview_pos[0] + map_preview_size[0] + 50),
                   pad_y + (map_preview_pos[1] + map_preview_size[1] // 2)), 'right', width=40, height=80)
```

Description

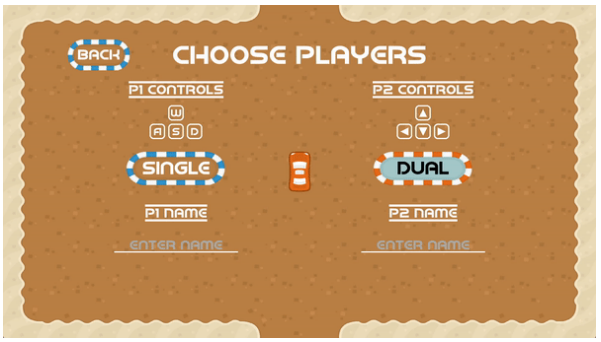
In order for the choose_map_window() function to display the map for gameplay, the function would've had to build each layer of the map every frame, making the game very resource intensive and possibly too hard to run on lower-end systems. Therefore, this window has its own global variable that it builds the selected map once upon loading it, then saves it as a single surface using the variable. This means that the window can instead just show the already built surface and is a lot more computational efficient.

Choose players window

Old version



New version



Code

```
def choose_players_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Choose Players', WHITE, 100)  # Title

    x = pad_x + 210
    y = pad_y + 112
    tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100))  # Back button
    tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100))
    draw_text(x + 100, y + 23, 'Back', WHITE, 55)

    if Player_amount == 1:
        # Controls
        x = pad_x + CENTRE[0]
        y = pad_y + 240
        draw_text(x, y, 'P1 controls', WHITE, 50, bar=True)
        draw_controls(x, y + 140, 'wasd')

        x = pad_x + 400
        y = pad_y + 476
        tile(x, y, 'road', 77, grid=False)  # Single Player button
        tile(x + 65, y, 'road', 2, grid=False)
        tile(x + 190, y, 'road', 61, grid=False)
        draw_text(x + 160, y + 20, 'Single', BLACK, 70)
```

```
        x = pad_x + CENTRE[0]
        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P1 Name', WHITE, 50, bar=True)
        # P1 name underline
        pygame.draw.line(Window, WHITE, (x - 200, y + 160), (x + 200, y + 160), 4)
        # P1 name
        rect = draw_text(x, y + 115, P1_name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 1:
            pygame.draw.line(Window, WHITE, (x + 5 + rect.width // 2, y + 124), (x + 5 + rect.width // 2, y + 152), 3)
        # Enter name prompt
        if selected_text_entry != 1 and not P1_name:
            if (pygame.time.get_ticks() // 1060) % 2:
                draw_text(x, y + 115, 'Enter name', V_LIGHT_GREY, 50)
            else:
                draw_text(x, y + 115, 'Enter name', LIGHT_GREY, 50)

    else:
        x = pad_x + 400
        y = pad_y + 476
        tile(x, y, 'dirt road', 76, grid=False)  # Single Player button
        tile(x + 65, y, 'dirt road', 1, grid=False)
        tile(x + 190, y, 'dirt road', 60, grid=False)
        draw_text(x + 160, y + 20, 'Single', WHITE, 70)


    if Player_amount == 2:
        # Controls
        x = pad_x + 560
        y = pad_y + 240
        draw_text(x, y, 'P1 controls', WHITE, 50, bar=True)
        draw_controls(x, y + 140, 'wasd')

        x = pad_x + 1360
        draw_text(x, y, 'P2 controls', WHITE, 50, bar=True)
        draw_controls(x, y + 140, 'arrows')

        x = pad_x + 1200
        y = pad_y + 476
        tile(x, y, 'road', 77, grid=False)  # Dual Player button
        tile(x + 65, y, 'road', 2, grid=False)
        tile(x + 190, y, 'road', 61, grid=False)
        draw_text(x + 160, y + 20, 'Dual', BLACK, 70)

        x = pad_x + 560
        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P1 Name', WHITE, 50, bar=True)
        # P1 name underline
        pygame.draw.line(Window, WHITE, (x - 200, y + 160), (x + 200, y + 160), 4)
        # P1 name
        rect = draw_text(x, y + 115, P1_name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 1:
            pygame.draw.line(Window, WHITE, (x + 5 + rect.width // 2, y + 124), (x + 5 + rect.width // 2, y + 152), 3)
        # Enter name prompt
        if selected_text_entry != 1 and not P1_name:
            if (pygame.time.get_ticks() // 1060) % 2:
                draw_text(x, y + 115, 'Enter name', V_LIGHT_GREY, 50)
            else:
                draw_text(x, y + 115, 'Enter name', LIGHT_GREY, 50)

        x = pad_x + 1360
        # P2 name title
        draw_text(x, y, 'P2 Name', WHITE, 50, bar=True)
        # P2 name underline
        pygame.draw.line(Window, WHITE, (x - 200, y + 160), (x + 200, y + 160), 4)
        # P2 name
        rect = draw_text(x, y + 115, P2_name, WHITE, 50, return_rect=True)
        # P2 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 2:
            pygame.draw.line(Window, WHITE, (x + 5 + rect.width // 2, y + 124), (x + 5 + rect.width // 2, y + 152), 3)
        # Enter name prompt
        if selected_text_entry != 2 and not P2_name:
            if (pygame.time.get_ticks() // 1060) % 2:
                draw_text(x, y + 115, 'Enter name', V_LIGHT_GREY, 50)
            else:
                draw_text(x, y + 115, 'Enter name', LIGHT_GREY, 50)

    else:
        x = pad_x + 1200
        y = pad_y + 476
        tile(x, y, 'dirt road', 76, grid=False)  # Dual Player button
        tile(x + 65, y, 'dirt road', 1, grid=False)
        tile(x + 190, y, 'dirt road', 60, grid=False)
        draw_text(x + 160, y + 20, 'Dual', WHITE, 70)

    if Player_amount == 1 and P1_name and P1_name.strip() or \
            Player_amount == 2 and P1_name.strip() and P2_name.strip():
        x = pad_x + 800
        y = pad_y + 850
        tile(x, y, 'dirt road', 76, grid=False)  # Start button
        tile(x + 65, y, 'dirt road', 1, grid=False)
        tile(x + 190, y, 'dirt road', 60, grid=False)
        draw_text(x + 160, y + 20, 'Start', WHITE, 70)
```

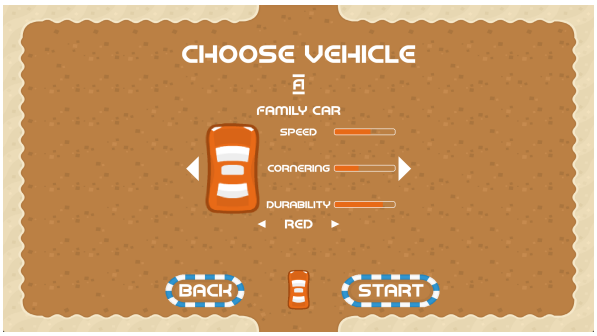
Description

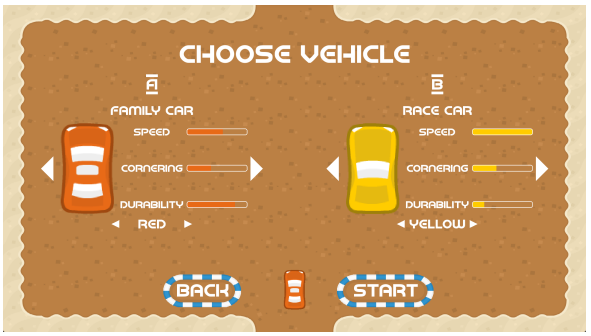
This window has 2 modes since it has to handle single player and dual player options, and although the game could just not show half of the window, it wouldn't look very good. Therefore this window can either show a single set of assets for single player mode and centre the assets, or can show 2 sets of assets (one for each player) and have one on each side of the screen. This window can also handle text input for the player names and has animated 'Choose name' and text cursor prompts. The window also uses render_key() to generate a little preview of the player controls and show the user(s) which inputs they have to use for their car.

Choose vehicle window

Old version

New version





Code

```
def choose_vehicle_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Choose Vehicle', WHITE, 100)  # Title

    x = pad_x + 528
    y = pad_y + 890
    tile(x, y, 'dirt road', 76, grid=False)  # Back button
    tile(x + 128, y, 'dirt road', 60, grid=False)
    draw_text(x + 130, y + 20, 'Back', WHITE, 70)

    x = pad_x + 1100
    y = pad_y + 890
    tile(x, y, 'dirt road', 76, grid=False)  # Start button
    tile(x + 65, y, 'dirt road', 1, grid=False)
    tile(x + 190, y, 'dirt road', 60, grid=False)
    draw_text(x + 160, y + 20, 'Start', WHITE, 70)
```

```
if Player_amount == 1:
    draw_text(pad_x + CENTRE[0], pad_y + 220, Pl_name, WHITE, 60, bar=True)
    draw_text(pad_x + CENTRE[0], pad_y + 325, Pl_veh[0], WHITE, 50)

    pl_veh_rect = Pl_veh[1].get_rect()
    draw_triangle((pad_x + CENTRE[0] - 170 - pl_veh_rect.width, pad_y + CENTRE[1]), 'left', width=40, height=80)
    Window.blit(Pl_veh[1], (pad_x + CENTRE[0] - 215 - pl_veh_rect.width // 2,
                            pad_y + CENTRE[1] - pl_veh_rect.height // 2))
    draw_triangle((pad_x + CENTRE[0] + 170 + pl_veh_rect.width, pad_y + CENTRE[1]), 'right', width=40, height=80)

    draw_text(pad_x + CENTRE[0], pad_y + CENTRE[1] + 160, Pl_veh[2], WHITE, 50)
    draw_triangle((pad_x + CENTRE[0] - 120, pad_y + CENTRE[1] + 183), 'left', width=25, height=25)
    draw_triangle((pad_x + CENTRE[0] + 120, pad_y + CENTRE[1] + 183), 'right', width=25, height=25)

    draw_text(pad_x + CENTRE[0], pad_y + 400, 'Speed', WHITE, 40)
    draw_text(pad_x + CENTRE[0], pad_y + 520, 'Cornering', WHITE, 40)
    draw_text(pad_x + CENTRE[0], pad_y + 640, 'Durability', WHITE, 40)

    if Pl_veh[0] == 'Family Car':
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Sports Car':
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Luxury Car':
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Truck':
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Race Car':
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                    (200, 22), 1, 0, 5, center_x=False, fill_color=Pl_veh[3])

elif Player_amount == 2:
    # 1P VEHICLE
    pos_x = CENTRE[0] // 2 + 10
    draw_text(pad_x + pos_x, pad_y + 220, Pl_name, WHITE, 60, bar=True)
    draw_text(pad_x + pos_x, pad_y + 325, Pl_veh[0], WHITE, 50)

    draw_triangle((pad_x + pos_x - 345, pad_y + CENTRE[1]), 'left', width=40, height=80)
    Window.blit(Pl_veh[1], (pad_x + pos_x - 300, pad_y + CENTRE[1] - 150))
    draw_triangle((pad_x + pos_x + 345, pad_y + CENTRE[1]), 'right', width=40, height=80)

    draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, Pl_veh[2], WHITE, 50)
    draw_triangle((pad_x + pos_x - 120, pad_y + CENTRE[1] + 183), 'left', width=25, height=25)
    draw_triangle((pad_x + pos_x + 120, pad_y + CENTRE[1] + 183), 'right', width=25, height=25)

    draw_text(pad_x + pos_x, pad_y + 400, 'Speed', WHITE, 40)
    draw_text(pad_x + pos_x, pad_y + 520, 'Cornering', WHITE, 40)
    draw_text(pad_x + pos_x, pad_y + 640, 'Durability', WHITE, 40)

    if Pl_veh[0] == 'Family Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Sports Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Luxury Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Truck':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 1, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=Pl_veh[3])
    elif Pl_veh[0] == 'Race Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=Pl_veh[3])
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 1, 0, 5, center_x=False, fill_color=Pl_veh[3])
```

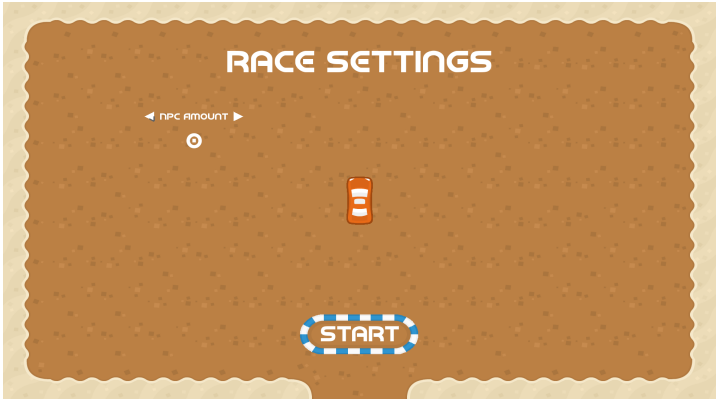
```
# 2P VEHICLE
pos_x = CENTRE[0] + CENTRE[0] // 2 - 10
draw_text(pad_x + pos_x, pad_y + 220, P2_name, WHITE, 60, bar=True)
draw_text(pad_x + pos_x, pad_y + 325, P2_veh[0], WHITE, 50)

draw_triangle((pad_x + pos_x - 345, pad_y + CENTRE[1]), 'left', width=40, height=80)
Window.blit(P2_veh[1], (pad_x + pos_x - 300, pad_y + CENTRE[1] - 150))
draw_triangle((pad_x + pos_x + 345, pad_y + CENTRE[1]), 'right', width=40, height=80)

draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, P2_veh[2], WHITE, 50)
draw_triangle((pad_x + pos_x - 120, pad_y + CENTRE[1] + 183), 'left', width=25, height=25)
draw_triangle((pad_x + pos_x + 120, pad_y + CENTRE[1] + 183), 'right', width=25, height=25)

draw_text(pad_x + pos_x, pad_y + 400, 'Speed', WHITE, 40)
draw_text(pad_x + pos_x, pad_y + 520, 'Cornering', WHITE, 40)
draw_text(pad_x + pos_x, pad_y + 640, 'Durability', WHITE, 40)

if P2_veh[0] == 'Family Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 4, 0, 5, center_x=False, fill_color=P2_veh[3])
elif P2_veh[0] == 'Sports Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 4, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 2, 0, 5, center_x=False, fill_color=P2_veh[3])
elif P2_veh[0] == 'Luxury Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 3, 0, 5, center_x=False, fill_color=P2_veh[3])
elif P2_veh[0] == 'Truck':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 2, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 1, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 5, 0, 5, center_x=False, fill_color=P2_veh[3])
elif P2_veh[0] == 'Race Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 5, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=P2_veh[3])
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 1, 0, 5, center_x=False, fill_color=P2_veh[3])
```

Description

Much like the window where the users can choose the amount of people, this window also has to have 2 separate modes for either single player or dual player. This window also has a lot of assets, and shows the user the individual damage, speed and turning speed of each car and allows them to choose the colour they have picked.

Race settings window

Old version



New version

Code

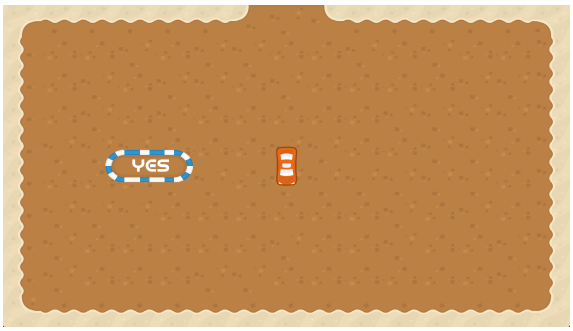
```
def race_settings_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Race Settings', WHITE, 100)  # Title

    x = pad_x + 210
    y = pad_y + 112
    tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100))  # Back button
    tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100))
    draw_text(x + 100, y + 23, 'Back', WHITE, 55)

    x = pad_x + 800
    y = pad_y + 850
    tile(x, y, 'dirt road', 76, grid=False)  # Start button
    tile(x + 65, y, 'dirt road', 1, grid=False)
    tile(x + 190, y, 'dirt road', 60, grid=False)
    draw_text(x + 160, y + 20, 'Start', WHITE, 70)

    x = pad_x + 511
    y = pad_y + 345
    draw_text(x, y, str(Npc_amount), WHITE, 70)  # NPC Amount option
    draw_text(x, y - 48, 'NPC Amount', WHITE, 30)
    draw_triangle((x - 120, y - 34), 'left', width=25, height=25)
    draw_triangle((x + 120, y - 34), 'right', width=25, height=25)

    x = pad_x + 1408
    y = pad_y + 345
    draw_text(x, y, str(Total_laps), WHITE, 70)  # Laps option
    draw_text(x, y - 48, 'Laps', WHITE, 30)
    draw_triangle((x - 60, y - 34), 'left', width=25, height=25)
    draw_triangle((x + 60, y - 34), 'right', width=25, height=25)
```

Description

This window allows the user(s) to change options related to the map they are going to play, such as the difficulty of the AI and the amount of laps they will race for to make the gameplay more versatile and tailored to what the user would prefer. This means that users are generally more likely to enjoy the game and will continue to play it for a longer period of time.

Confirm quit window

Old version



New version

Code

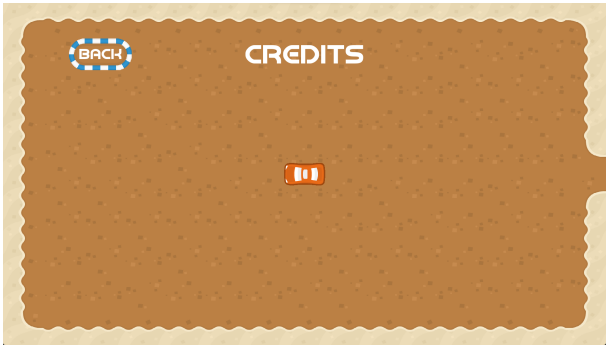
```
def confirm_quit_window(curr_bg, pad_x=0, pad_y=0, surf=Window):
    surf.blit(curr_bg, (pad_x, pad_y))  # Display the current bg

    x = pad_x + CENTRE[0]
    y = pad_y + 128
    draw_text(x, y, 'Are you sure?', WHITE, 100, surf=surf)  # Are you sure? title

    x = pad_x + 347
    y = pad_y + CENTRE[1] - (tile_scale[1] // 2)
    tile(x, y, 'dirt road', 76, grid=False, surf=surf)  # Yes button
    tile(x + 85, y, 'dirt road', 1, grid=False, surf=surf)
    tile(x + 168, y, 'dirt road', 60, grid=False, surf=surf)
    draw_text(x + 153, y + 20, 'Yes', WHITE, 70, surf=surf)

    x = pad_x + CENTRE[0] + 347
    y = pad_y + CENTRE[1] - (tile_scale[1] // 2)
    tile(x, y, 'dirt road', 76, grid=False, surf=surf)  # No button
    tile(x + 85, y, 'dirt road', 1, grid=False, surf=surf)
    tile(x + 168, y, 'dirt road', 60, grid=False, surf=surf)
    draw_text(x + 153, y + 20, 'No', WHITE, 70, surf=surf)
```

Description

This allows users to confirm if they want to quit the game to stop people accidentally pressing the quit button and exiting the game when they did not intend to. This window is the simplest of all and only has 2 buttons, however is used differently than the other windows the same as the settings window, as both these windows can be accessed from the pause menu when in a race the windows need to be able to use the Secondary_window instead of Window as the game is using Window already. Therefore these window functions can take an extra variable: surf, which defines where all of the assets will be drawn to so that they can be overlaid on top of the game screenshot and create a faded effect for the background.

Credits window

Old version



New version

Code

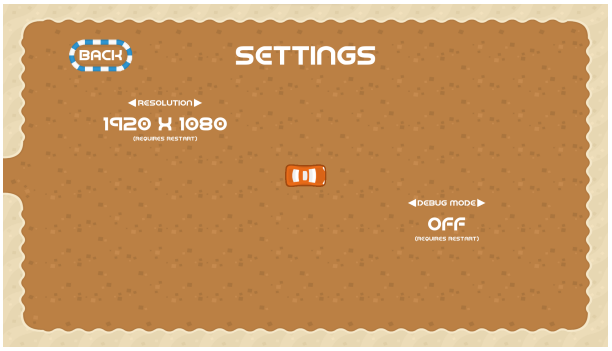
```
def credits_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Credits', WHITE, 100)  # Title

    y = pad_y + 215
    text = 'Developer - Anthony Guy'
    draw_text(x, y, text, WHITE, 50)  # Credits #1

    y += 70
    text = 'Graphics - Kenney Vleugels'
    draw_text(x, y, text, WHITE, 50)  # Credits #2

    y += 70
    text = 'Menu Music - Trevor Lentz'
    draw_text(x, y, text, WHITE, 50)  # Credits #3

    y += 70
    text = 'Game Music & SFX - Juhani Junkala'
    draw_text(x, y, text, WHITE, 50)  # Credits #4

    y = CENTRE[1] + 80
    text = 'Special Thanks'
    draw_text(x, y, text, WHITE, 50, bar=True)  # Credits #1

    y += 80
    text = 'Keith Brown'
    draw_text(x, y, text, WHITE, 50)  # Credits #2

    y += 70
    text = 'Jonny Farmer'
    draw_text(x, y, text, WHITE, 50)  # Credits #3

    x = pad_x + 210
    y = pad_y + 112
    tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100))  # Back button
    tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100))
    draw_text(x + 100, y + 23, 'Back', WHITE, 55)
```

Description

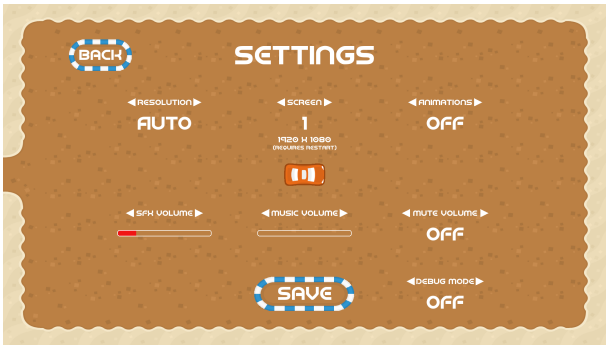
This window gives credit to the creators and contributors that make up the game. In order to have each contributor centred on the screen, the draw_text() function needs to be called separately for every line, as it only centres based on the width of the entire surface that the text is on however this method has an added benefit of customisable amount of space between the line which makes the window look neater.

Settings window

Old version



New version



Code

```
def settings_window(curr_bg, pad_x=0, pad_y=0, surf=Window):
    surf.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Settings', WHITE, 100, surf=surf)  # Title

    x = pad_x + CENTRE[0]
    y = pad_y + 220
    draw_text(x, y, 'Make sure you save your changes!', WHITE, 35, surf=surf)  # Tip

    x = pad_x + 210
    y = pad_y + 112
    tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100), surf=surf)  # Back button
    tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100), surf=surf)
    draw_text(x + 100, y + 23, 'Back', WHITE, 55, surf=surf)
```

```
x = pad_x + 800
y = pad_y + 864
tile(x, y, 'dirt road', 76, grid=False, surf=surf)  # Save button
tile(x + 65, y, 'dirt road', 1, grid=False, surf=surf)
tile(x + 190, y, 'dirt road', 60, grid=False, surf=surf)
draw_text(x + 160, y + 20, 'Save', WHITE, 70, surf=surf)

x = pad_x + 514
y = pad_y + 345
if Display_resolution == Desktop_info[Screen]:
    text = 'Auto'
else:
    text = str(Display_resolution[0]) + ' x ' + str(Display_resolution[1])  # Resolution option
draw_text(x, y, text, WHITE, 70, surf=surf)
draw_text(x, y - 48, 'Resolution', WHITE, 30, surf=surf)
draw_triangle((x - 105, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 105, y - 34), 'right', width=25, height=25, surface=surf)

x = pad_x + 960
y = pad_y + 345
text = str(Desktop_info[Screen][0]) + ' x ' + str(Desktop_info[Screen][1])  # Screen option
draw_text(x, y, str(Screen + 1), WHITE, 70, surf=surf)
draw_text(x, y - 48, 'Screen', WHITE, 30, surf=surf)
draw_triangle((x - 75, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 75, y - 34), 'right', width=25, height=25, surface=surf)
draw_text(x, y + 70, text, WHITE, 30, surf=surf)
draw_text(x, y + 100, '(Requires restart)', WHITE, 20, surf=surf)

x = pad_x + 1408
y = pad_y + 345
if Menu_animation:
    text = 'On'
else:
    text = 'Off'
draw_text(x, y, text, WHITE, 70, surf=surf)  # Animation option
draw_text(x, y - 48, 'Animations', WHITE, 30, surf=surf)
draw_triangle((x - 105, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 105, y - 34), 'right', width=25, height=25, surface=surf)

x = pad_x + 1408
y = pad_y + 696
if Mute_volume:
    text = 'On'
else:
    text = 'Off'
draw_text(x, y, text, WHITE, 70, surf=surf)  # Mute volume option
draw_text(x, y - 48, 'Mute volume', WHITE, 30, surf=surf)
draw_triangle((x - 125, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 125, y - 34), 'right', width=25, height=25, surface=surf)

x = pad_x + 1408
y = pad_y + 912
if Debug:
    text = 'On'
else:
    text = 'Off'
draw_text(x, y, text, WHITE, 70, surf=surf)  # Debug option
draw_text(x, y - 48, 'Debug mode', WHITE, 30, surf=surf)
draw_triangle((x - 110, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 110, y - 34), 'right', width=25, height=25, surface=surf)

x = pad_x + 960
y = pad_y + 696
draw_slider((x, y + 30), (300, 20), Music_volume, 0, 1, surface=surf)  # Music volume option
draw_text(x, y - 48, 'Music Volume', WHITE, 30, surf=surf)
draw_triangle((x - 125, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 125, y - 34), 'right', width=25, height=25, surface=surf)

x = pad_x + 514
y = pad_y + 696
draw_slider((x, y + 30), (300, 20), Sfx_volume, 0, 1, surface=surf)  # Sfx volume option
draw_text(x, y - 48, 'Sfx Volume', WHITE, 30, surf=surf)
draw_triangle((x - 110, y - 34), 'left', width=25, height=25, surface=surf)
draw_triangle((x + 110, y - 34), 'right', width=25, height=25, surface=surf)
```

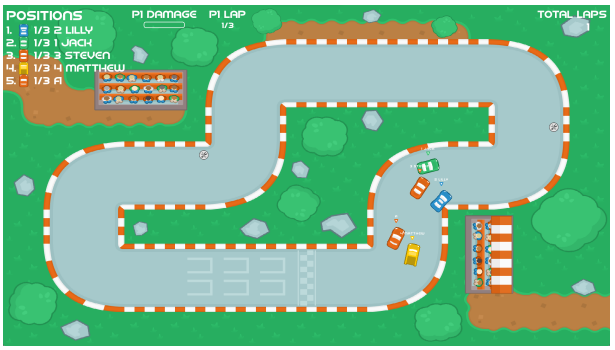
Description

This window allows users to change several options about the game such as the volumes for music and sound effects, as well as debug mode and screen resolution. This means that there are a lot of assets on this window as each text, description and button are all separate assets that have to be added in turn.

Gameplay screen

Old version

New version



Code

```
def gameplay_gui(player_list, game_countdown_timer, lap_timer):
    if Player_amount >= 1:
        draw_text(CENTRE[0] - 250, 10, 'P1 Lap', WHITE, 40)
        draw_text(CENTRE[0] - 250, 50, str(player_list[0].laps) + '/' + str(Total_laps), WHITE, 30)
        draw_text(CENTRE[0] - 450, 10, 'P1 Damage', WHITE, 40)
        draw_slider((CENTRE[0] - 450, 63), (130, 20), player_list[0].damage, 0,
                    player_list[0].durability, fill_color=player_list[0].colour)
    if Player_amount == 2:
        draw_text(CENTRE[0] + 250, 10, 'P2 Lap', WHITE, 40)
        draw_text(CENTRE[0] + 250, 50, str(player_list[1].laps) + '/' + str(Total_laps), WHITE, 30)
        draw_text(CENTRE[0] + 450, 10, 'P2 Damage', WHITE, 40)
        draw_slider((CENTRE[0] + 450, 63), (130, 20), player_list[1].damage, 0,
                    player_list[1].durability, fill_color=player_list[1].colour)

    draw_text(10, 10, 'Positions', WHITE, 50, center_x=False)  # Leaderboard positions
    for car_no in range(0, len(Player_positions)):
        draw_text(10, 40 * car_no + 60, str(car_no + 1) + '.', WHITE, 40, center_x=False)  # Car position
        Window.blit(pygame.transform.scale(pygame.image.load(Player_positions[car_no][2]),
                                           (30, 40)), (50, 40 * car_no + 60))  # Small car image
        draw_text(85, 40 * car_no + 60, ' ' + str(Player_positions[car_no][1]) + '/' + str(Total_laps) + ' ' +
                  Player_positions[car_no][0], WHITE, 40, center_x=False)  # Car name and current lap

    draw_text(1800, 10, 'Total Laps', WHITE, 40)
    if Player_positions:
        draw_text(1850, 50, Player_positions[0][1], WHITE, 40)
        if lap_timer > pygame.time.get_ticks():
            draw_text(CENTRE[0], CENTRE[1], 'Lap ' + str(Player_positions[0][1]), WHITE, 70, bar=True)
    else:
        draw_text(1850, 50, 0, WHITE, 40)

    if game_countdown_timer:
        draw_text(CENTRE[0], CENTRE[1] - 50, Player_positions[0][0] + ' has finished!', WHITE, 50)
        draw_text(CENTRE[0], CENTRE[1], 'Game ends in ' + str(game_countdown_timer // 1000), WHITE, 50)
```

Description

The gameplay GUI function is the only function that doesn't really follow the general rule of the other windows since there are no buttons or interactions and it is purely informational. The function takes the player_list and the countdown timer and fetches then displays all of the required information from the car classes for the players such as their current lap and how much damage they have taken, then shows the user at the top of the screen

including a progress bar to show their current damage. Every time the lap advances it is displayed in the centre of the screen to better inform the user of which lap they are on and will stay for 5 seconds before disappearing before the next lap advances and the cycle repeats.

Powerups can randomly spawn during the race and are dependent on the amount of cars on the track at once (if there are few cars then there will be less power ups). There are 4 types of power ups:

1. Repair (spanner & screwdriver cross) - Sets player's damage to 0
2. Boost (purple circle) - Sets player's move & rotation speed to 10
3. Lightning (lightning bolt) - Randomly makes an NPC crash
4. Bullet (bullets) - Stops the player moving for a few seconds

For every powerup, the influences are scaled based on how good the player's car is eg. If the player picks up a boost in a speed 1 car then it will last longer than a player in a speed 5 car. This also means that the bullet power up will penalise a faster car for longer ect. Once a power up spawns, there can only be 5 (times the amount of players) on the screen at the same time, so no more power ups will spawn after that. Furthermore, to keep the amount and type of power ups constantly changing, after 15 seconds a powerup will despawn if it hasn't already been picked up by the player at that point to allow another powerup to spawn. This means that mechanics like the bullet powerup where the player is not supposed to obtain the powerup can be achieved whilst still having all other powerups and removing the possibility of having the max amount of power ups all being bullet and the player not being able to get a boost or other type of aid during that race.

When an NPC crashes they will temporarily stop themselves based on how fast their car is (faster = longer) before continuing their set randomised path. Once a car has finished, a prompt will appear on the screen with the following: "[player name] has finished!" underneath will say "Game finishes in [5 second countdown]" before fading to black, which gives other players a small window in which they are also able to finish the game before it times out.

During the race a music loop will be played and every lap the music will play a sound then advance to the next 'stage' before looping back around to the first type to give a seamless effect of ever changing background music and adding more suspense to the game. For example, if the race lasted 6 laps then the music would wrap back around on lap 4 to the first stage of music. Each track loops seamlessly and has no pause or gap in between thanks to a dedicated thread that is independent of the main race loop.

Leaderboard window

Old version

New version

Code

```
def leaderboard_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    x = pad_x + CENTRE[0]
    y = pad_y + 115
    draw_text(x, y, 'Leaderboard', WHITE, 100)  # Title

    # x = pad_x + 210
    # y = pad_y + 112
    # tile(x, y, 'dirt road', 76, grid=False, scale=(100, 100))  # Back button
    # tile(x + 100, y, 'dirt road', 60, grid=False, scale=(100, 100))
    # draw_text(x + 100, y + 23, 'Quit', WHITE, 55)

    x = pad_x + 800
    y = pad_y + 764
    tile(x, y, 'dirt road', 76, grid=False)  # Finish button
    tile(x + 65, y, 'dirt road', 1, grid=False)
    tile(x + 190, y, 'dirt road', 60, grid=False)
    draw_text(x + 160, y + 20, 'Finish', WHITE, 70)

    for car_no in range(0, len(Player_positions)):
        rect = draw_text(CENTRE[0], 85 * car_no + 250, str(car_no + 1) + '. ' + Player_positions[car_no][0],
                         WHITE, 75, return_rect=True)  # Car position
        Window.blit(pygame.transform.scale(pygame.image.load(Player_positions[car_no][2]),
                                           (45, 68)), (CENTRE[0] + rect.width // 2 + 15, 85 * car_no + 250))

    # Draw race info at bottom of screen
    text = 'Map: ' + str(Map) + '  |  Laps: ' + str(Total_laps) + '  |  NPCs: ' + str(Npc_amount) + \
           '  |  Players: ' + str(Player_amount) + '  |  Time: ' + str(Race_time)
    draw_text(CENTRE[0], 926, text, WHITE, 50)
```

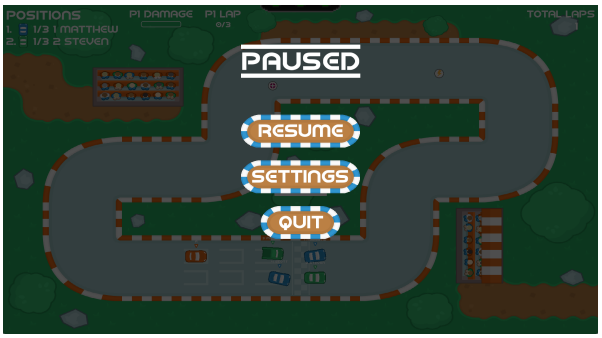
Description

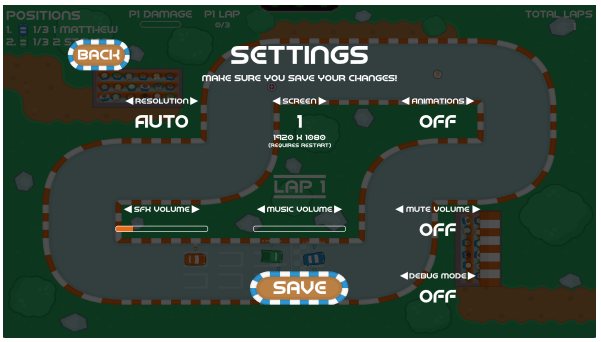
This window only shows up one time after every race and displays the leaderboard of who finished where as well as additional information about the race such as the map, amount of laps and how long the race took so that players can compare their scores with their friends and have more fun

competing with each other. This window also has its own dedicated relaxed version of the game music to smoothen the transition between the energetic race music and the calmer menu music. Having music that is similar to the race's also signifies that the player is not fully back into the menus and is still finishing the race.

Additional windows

Pause menu (during gameplay)

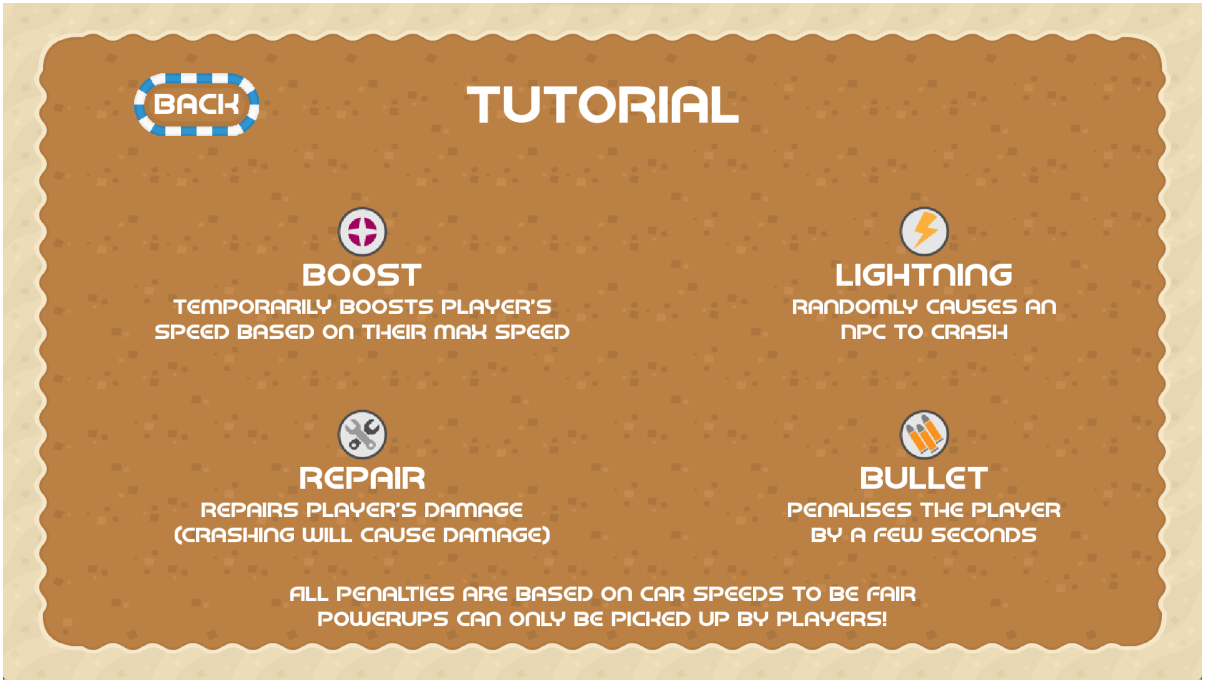




Description

This window can be accessed by the player at any time during a race apart from the countdown to start and finish. There are multiple ways of getting to and from the game and this window - The player can press the escape key at any time to enter and exit the pause menu, and if the player tabs out of the game to do other things on a different window the game will pause itself however when the player tabs back into the game the window will stay on the pause menu until the player is ready to unpause. There is also an unpause button that the player can use to exit the pause menu apart from the escape key. When transitioning to and from the pause menu a screenshot of the game will be displayed and the background will fade - out to give the impression that the game has been frozen in time. Once in the pause menu the player can choose to alter their settings or quit the game and all of the settings are identical to the ones on the main menu using the same function for a different purpose which shows how versatile and easy it is to implement more windows using the method that I have chosen. If the user presses the quit button on the pause menu, the game will ask them if they are sure and that they will lose their progress from that race allowing the user to go back if they did not intend to quit the race.

Tutorial window



Description

The tutorial window is purely to inform the user of what the different types of powerups do to and what they look like to help them decide if they want to go after them during a game - for example, the bullet power up penalises the player for a few seconds and therefore the player should stay away from them. The existence of this powerup is to make sure that the player is paying attention to the type of power up that they are going for and that they are not just going for every power up they see. It also makes sure that the players are not overpowered with the power ups and that the game is still fair to the NPCs. The player would only know this from either finding out by experimenting during a race or could visit this window anytime which is much easier for the user rather than trying to guess what each of the power ups do.