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

Computer Science Programming Project

Retro Rampage

https://github.com/bungee-boy/Retro-Rampage

Developed by Anthony Guy
2021 - 2023

Contents

Analysis **3**
- Description of the problem 3
- Stakeholders 3
- Why use a computer? 4
- Owner Interview 4
- Questionnaire 6
- Market Research 17
- Client Proposal 21
- Stakeholder Requirements 22
- Software and Hardware requirements 22
- Success Criteria 22

Design **25**
- Hand-drawn designs 25
- Flow Charts 34
- Alpha + Beta Tests 48
- Variables (Global + Classes) 53
- Functions 66
- Classes 72
- Imports 73
- Libraries 74

Development **76**
- Implementation & Improvements 76
- Alpha + Beta Testing 103
- User Feedback 111

Evaluation **113**
- Success Criteria 113
- Usability Questionnaire 115
- Future Developments 118
- Additional Features 118
- Maintenance 119

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 use a computer?

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 it 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...

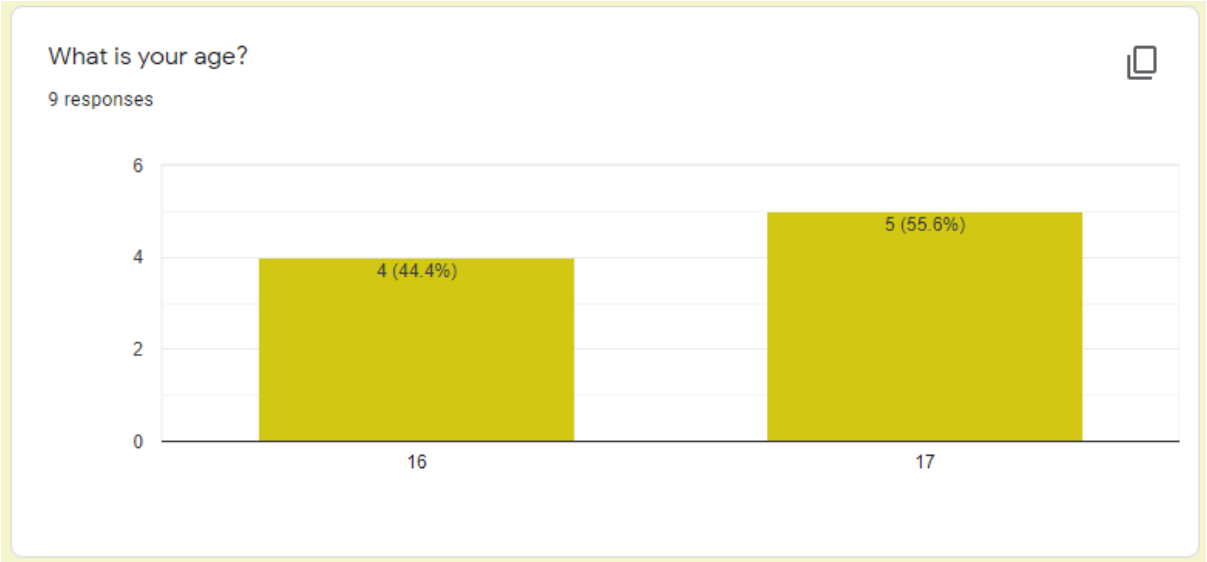
How many levels / maps would you like to see?

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

How many AI cars would you like to race against?

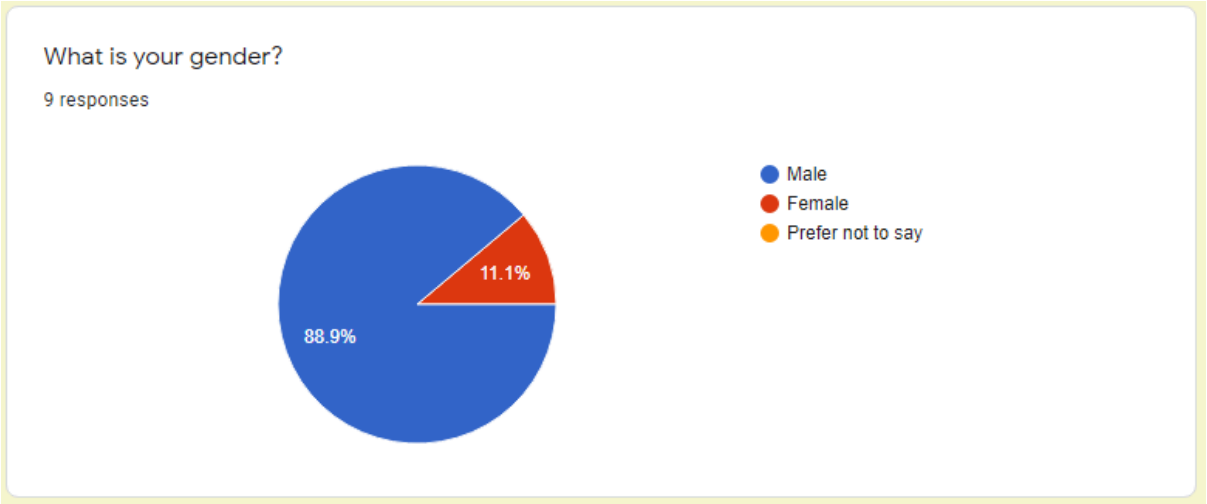
1	2	3	4	5
○	○	○	○	○

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

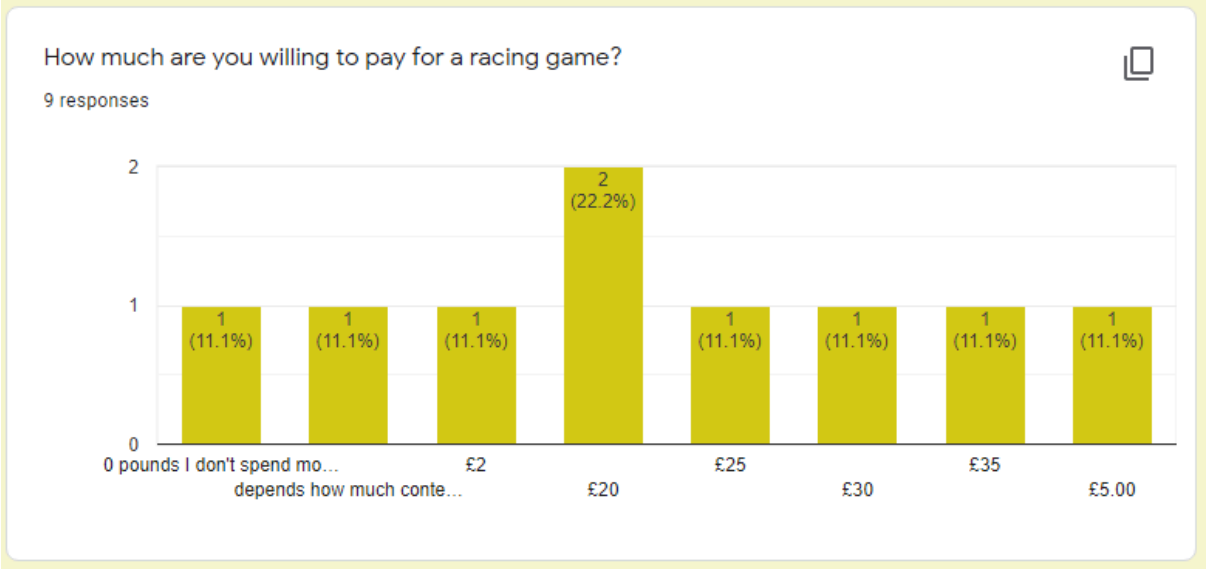


The responses show that 89% of people are males, this means that we should look to implement things that are more appealing to men - such as motorbikes as well as cars.

Name Suggestions

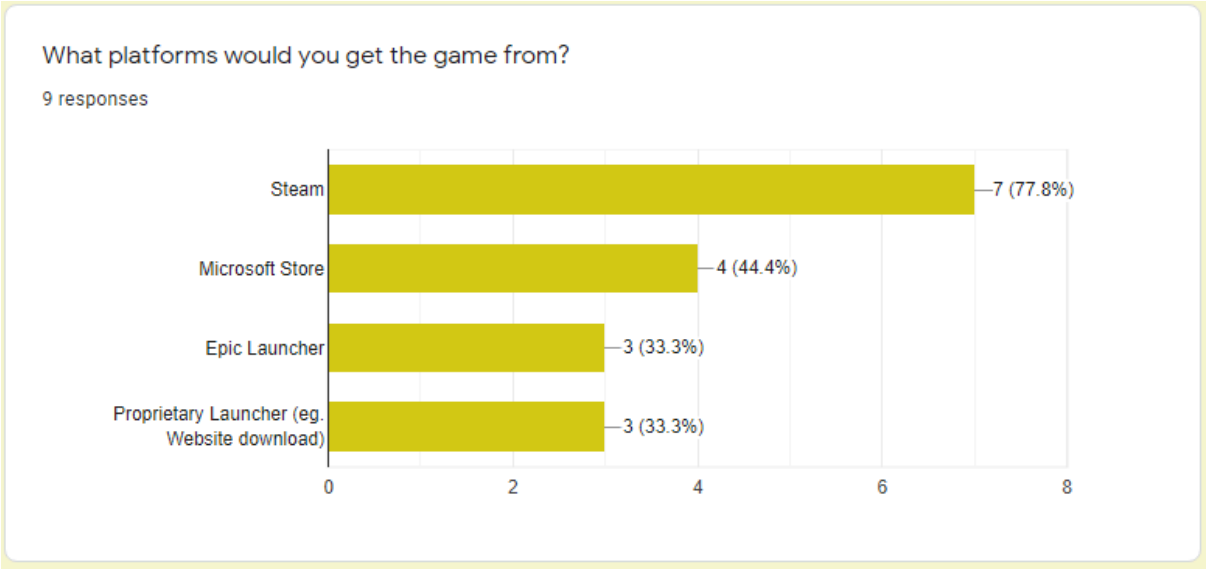
Out of the responses to this question, I think that the most appealing name for this game is ‘Retro Rampage’. This will be implemented into the advertising and game design.

Suggested price range



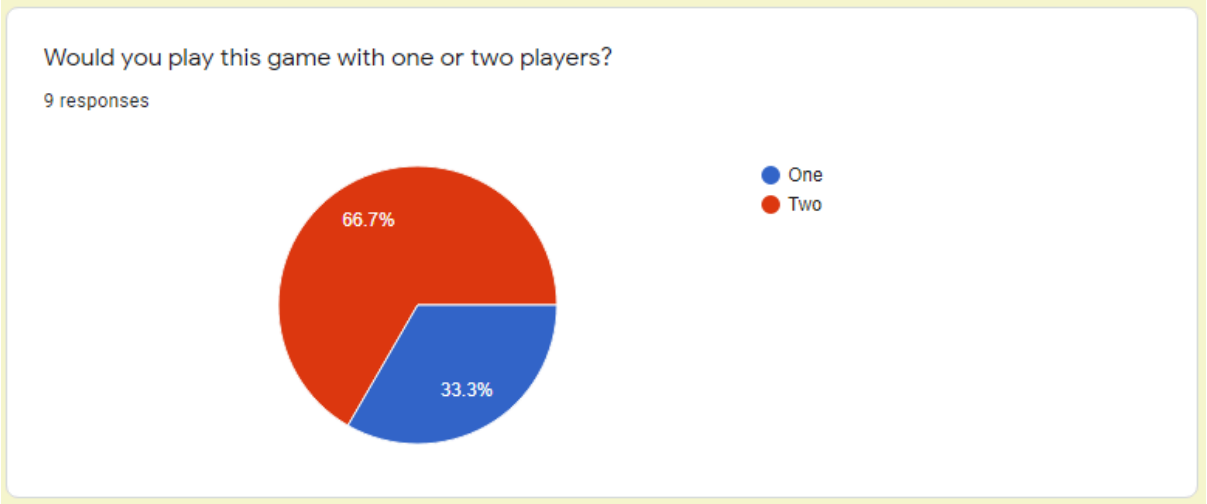
The responses show that most people would be willing to pay around £20 for this game, however they did not know that it would be in the style of an indie game. With this in mind, I believe that the appropriate price per game would be around £3 - £5.

Platforms



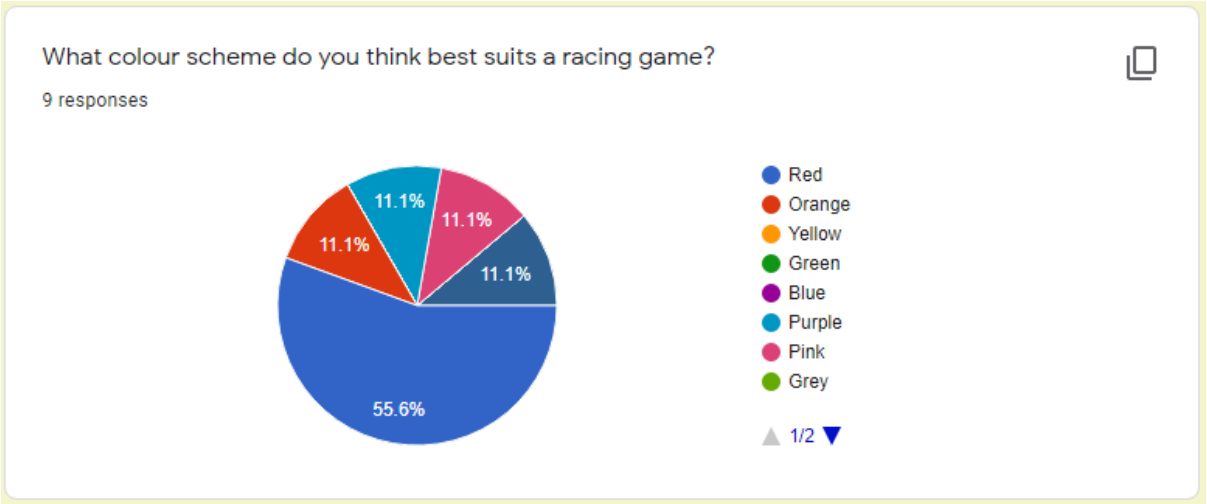
According to the responses, it would be most convenient for advanced gamers that the game is available on steam, but it should also be available on the microsoft store for people who are not as familiar with dedicated game launchers.

Amount of players



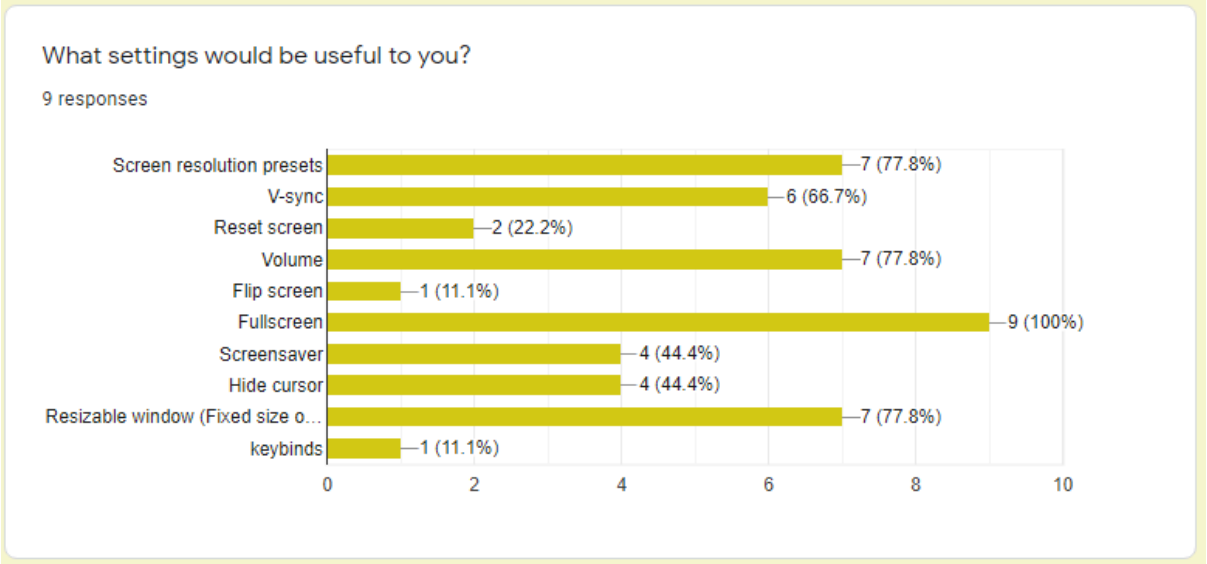
The responses show that if a two player option was implemented into the game, then 67% of the players would use it to play with either a friend or family member. Therefore, this option will be implemented into the game.

Colour scheme



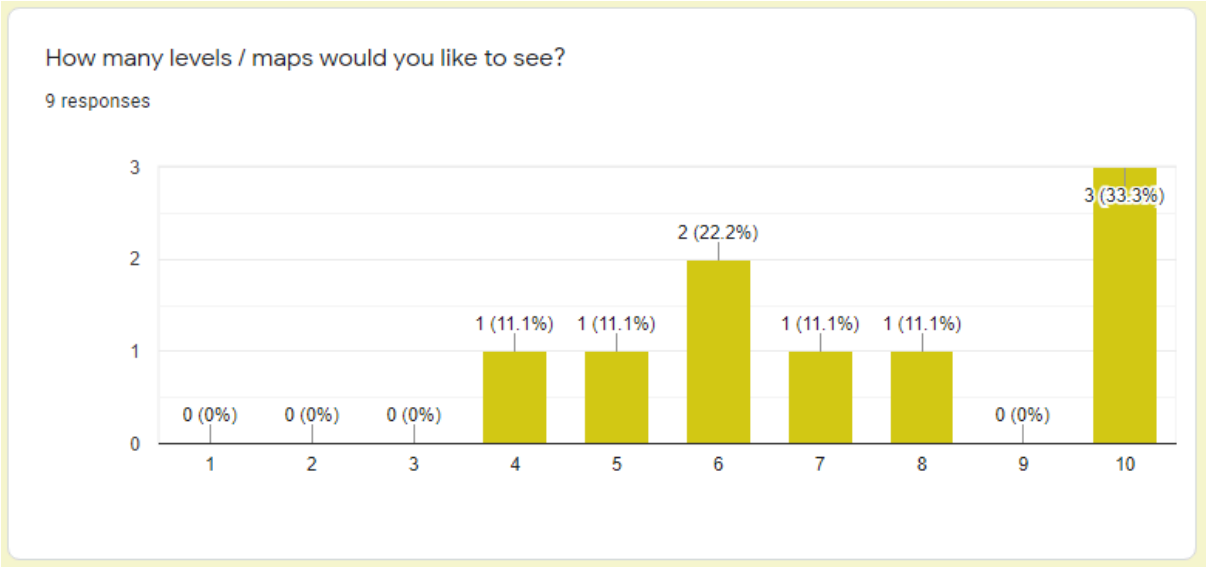
The data above shows that people would be more attracted to a red colour scheme for this game… However it may be more appropriate to use assets for the UI design as well as the gameplay itself.

Video settings



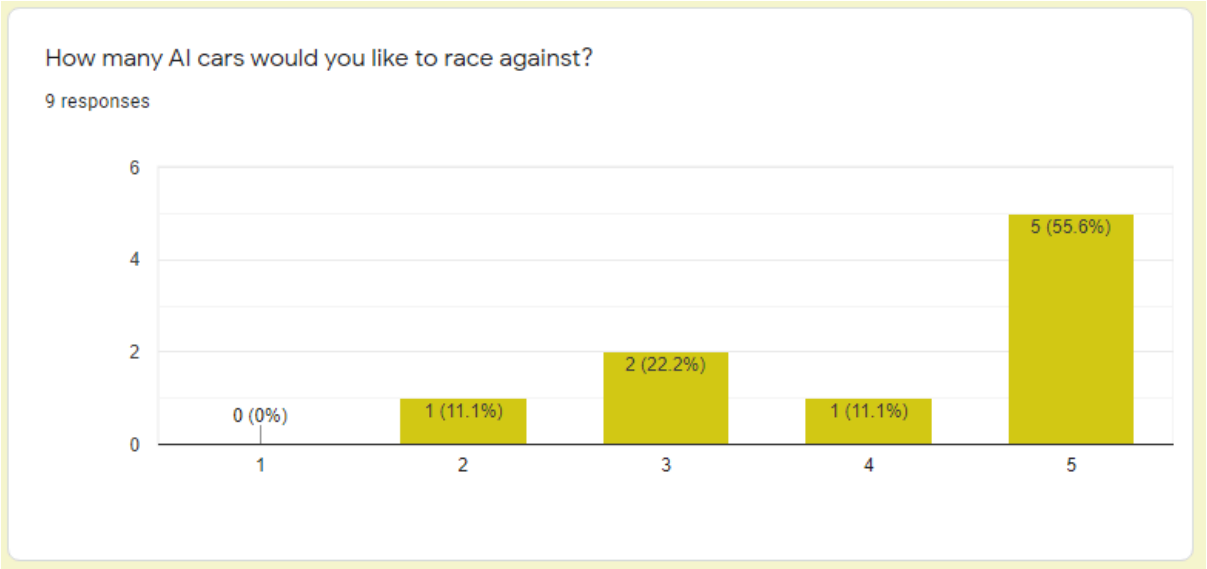
People would find having a fullscreen option very useful for a setting, with a three way tied second of volumes, resolution presets and resizable window option. Having a resizable window could lead to a lot of errors and bugs in the game, so that will not be added but resolution presets and fullscreen 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



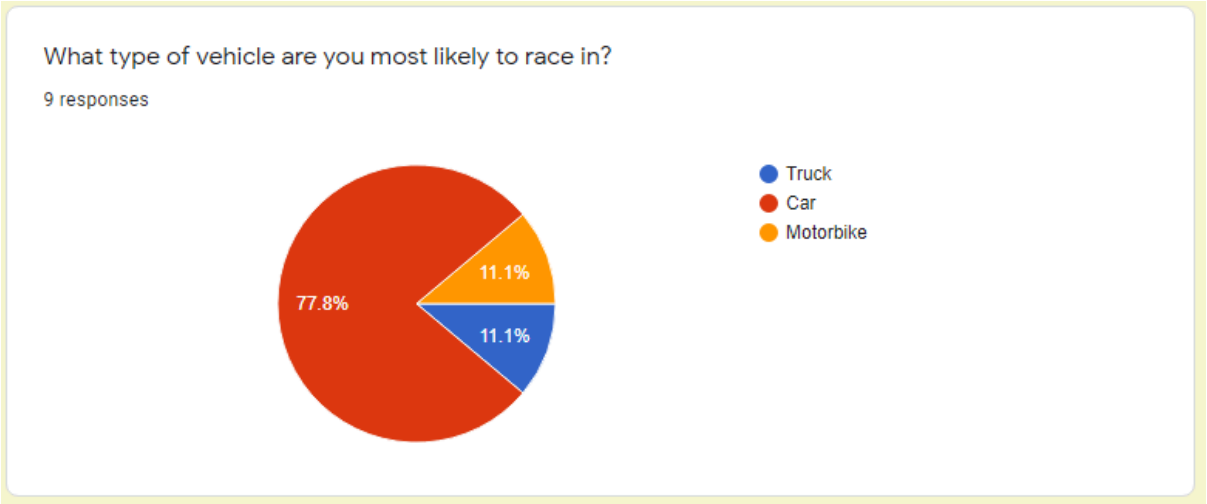
The amount of maps that are implemented into the game will heavily depend on the speed and progress of development, but as people would like to see at least 10 levels implemented… The development team will aim for more than 10 levels to be implemented.

Amount of AI



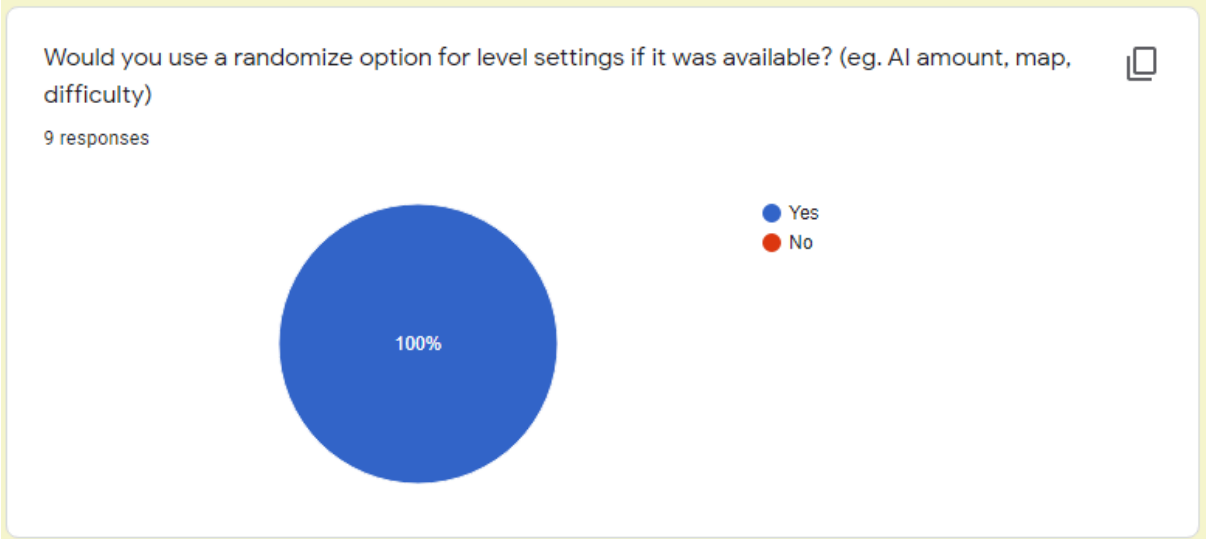
The survey data shows that people would find the game more fun if there is a maximum number of 5 AI cars that the player can race against, therefore there will be an option of up to 5 AI cars to race against in the main menu.

Type of vehicle



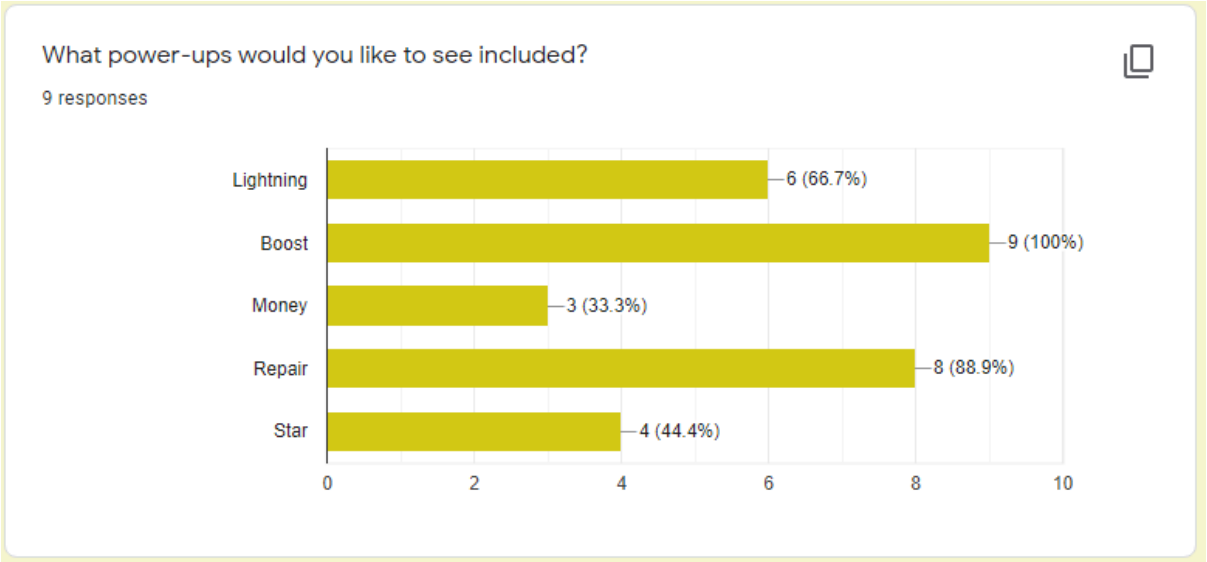
The data above shows that the majority of people would just use cars for their vehicle, but as shown by the previous gender question, some males picked motorbikes and trucks for their vehicles. This is important as it proves that we should implement features that give more interest to men.

Randomisation



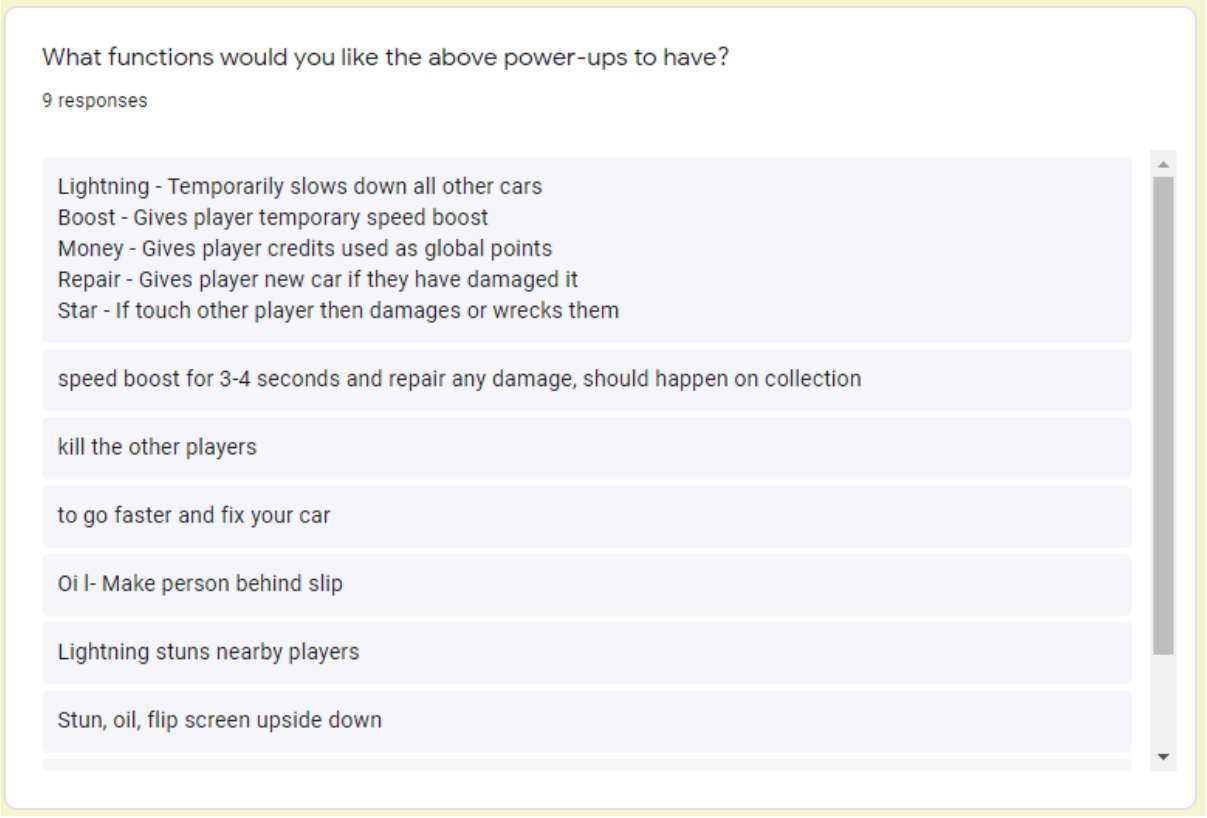
Out of the 9 people surveyed, all of them said that they would like an option to play a race where all of the options are randomised. Therefore to make more people play and buy the game we will implement this option so that players can have an infinite amount of campaign levels to race.

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

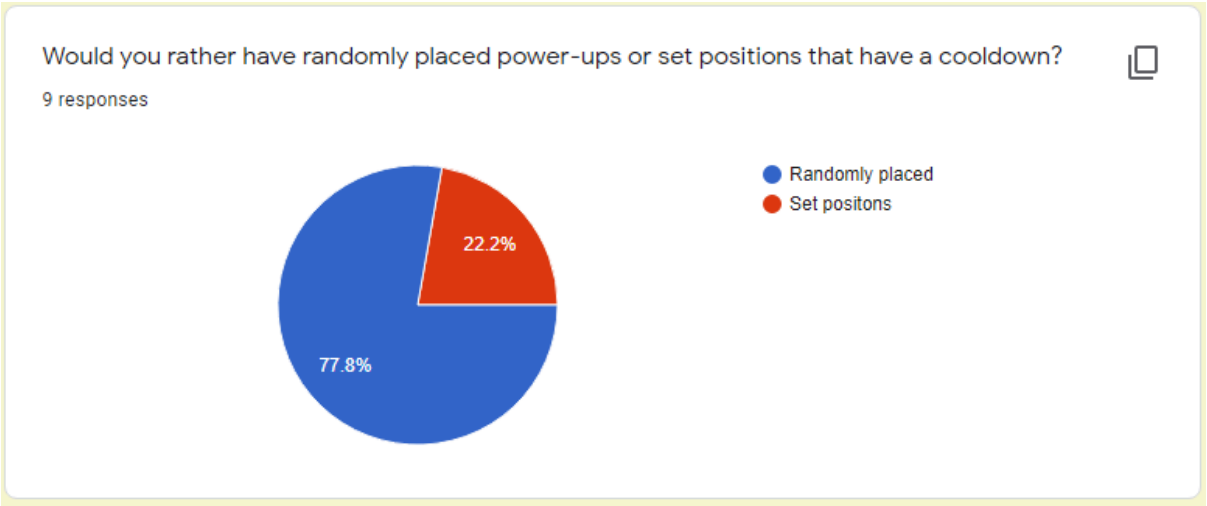


stun-Lighting, temporary speed - boost, repair-fix damage

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.

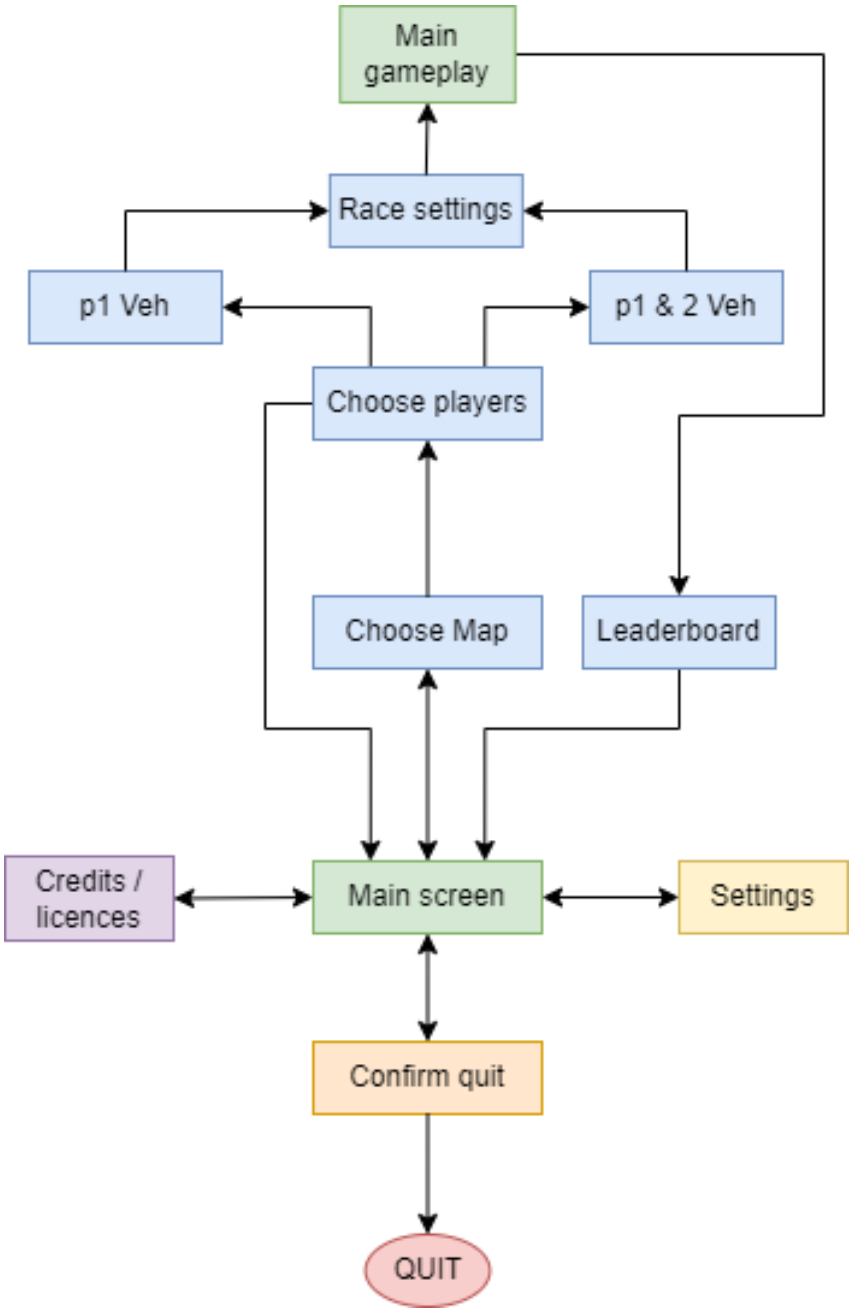
The limitations of Retro Rampage will mainly be that it is not a 3D game, so visualising and finding a good way to view what is happening in the game without any depth will be hard as it is much easier for the player to see what is close and what is far in a 3D game, however it is something that a 2D 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 such as sound and music volume
- 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 minimum
200mb free storage space
2Ghz CPU +
(No GPU required)

Success Criteria

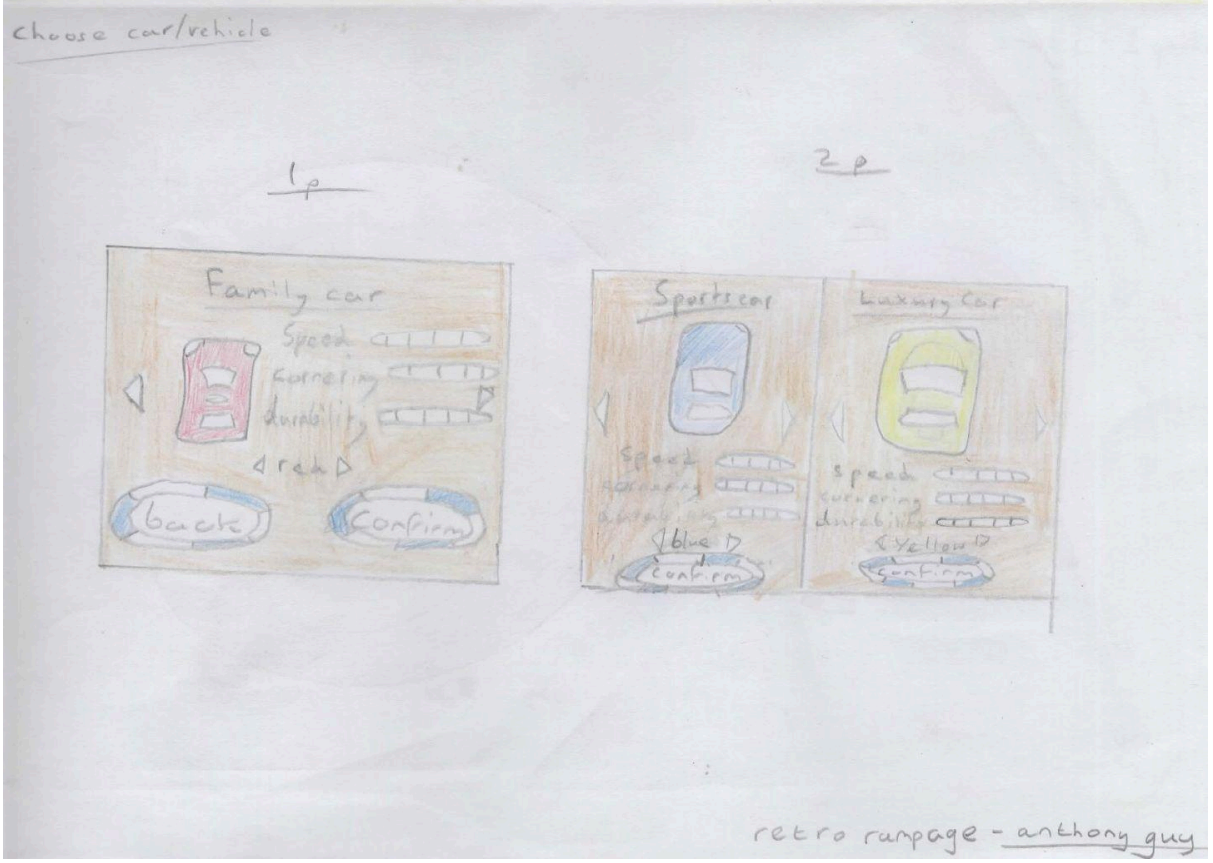
Number	Criteria	How to implement	Alternative
1	Main menu to start and change settings	Home screen with multiple buttons to move to other screen	Keyboard shortcuts to other windows like settings
2	Confirm quit screen	Yes or no button in centre	Quits without asking or waits before quitting
3	Travel between 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 amount of players	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 colour	After map and amount of players has been chosen, one by one player	Split-screen player control independent at same time
9	Amount of NPCs	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	Names above cars	Centred above player's car	Always vertical above car
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	Separate control of audio and sounds	In settings menu	Upon first time launch
18	Option to randomise race settings	On race settings window	On any window before the race
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

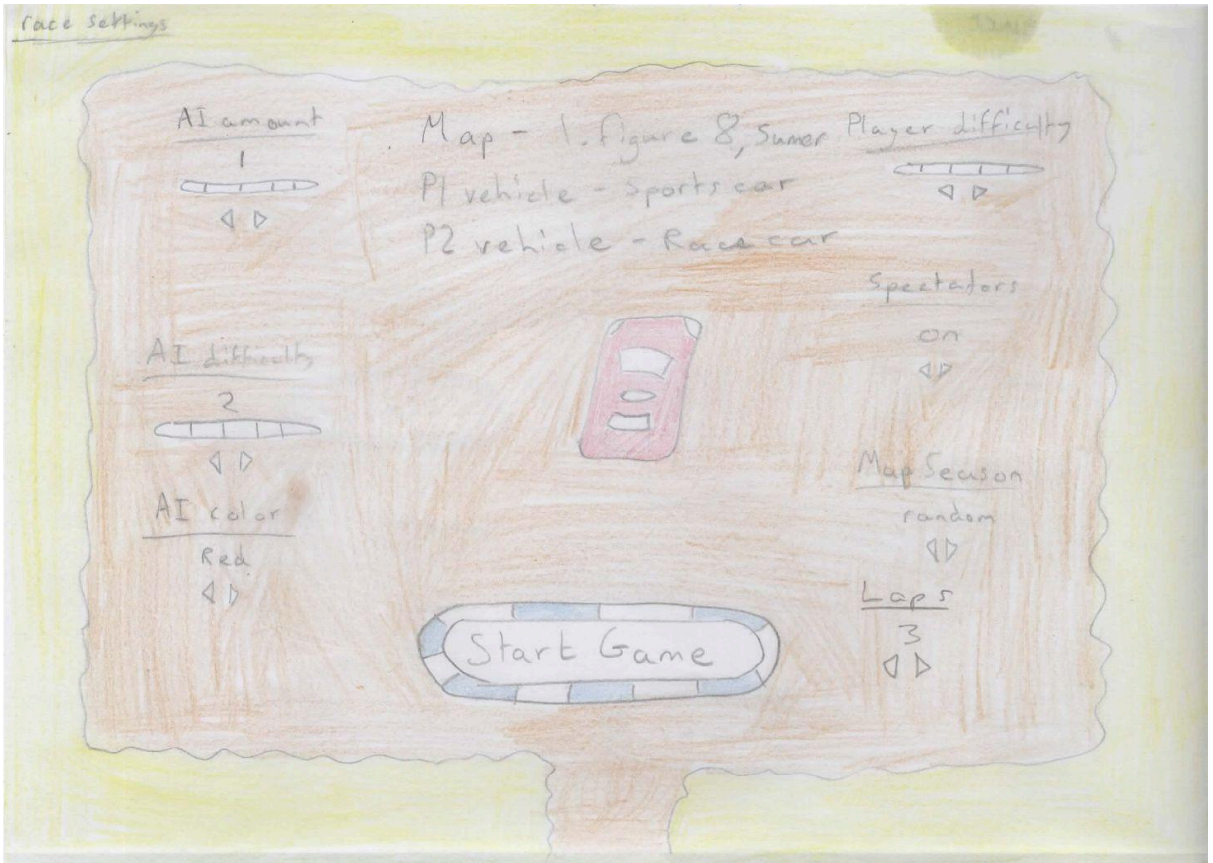
21	Performance settings for low-end computers	In settings window	Together with other settings
22	Change NPC difficulty	When choosing NPC options	When choosing the map
23	Toggle menu scrolling animation	Through an option in settings	Together with other settings
24	Change the window resolution	On settings window	Together with other settings
25	Have an auto option for resolution	In settings window	Together with other settings
26	Be able to specify a display	In settings window	Together with other settings

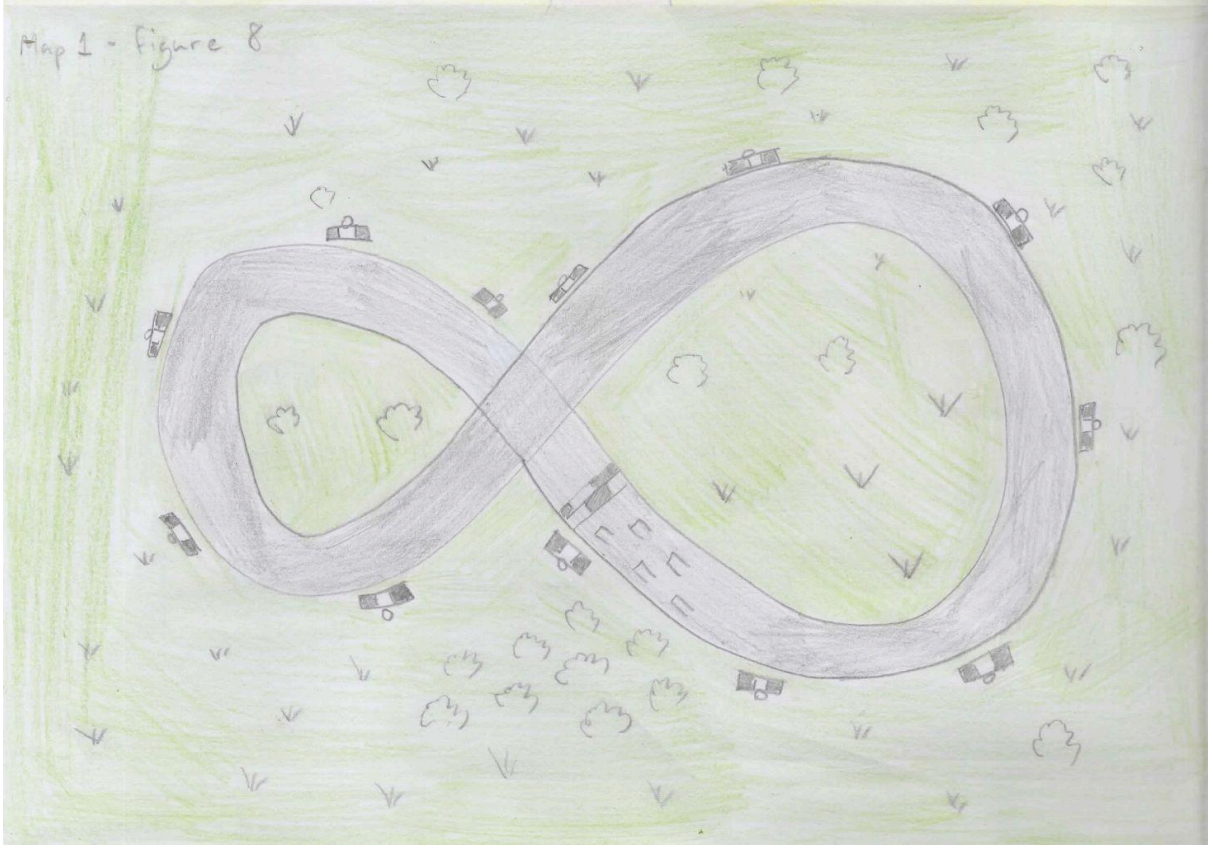
Design

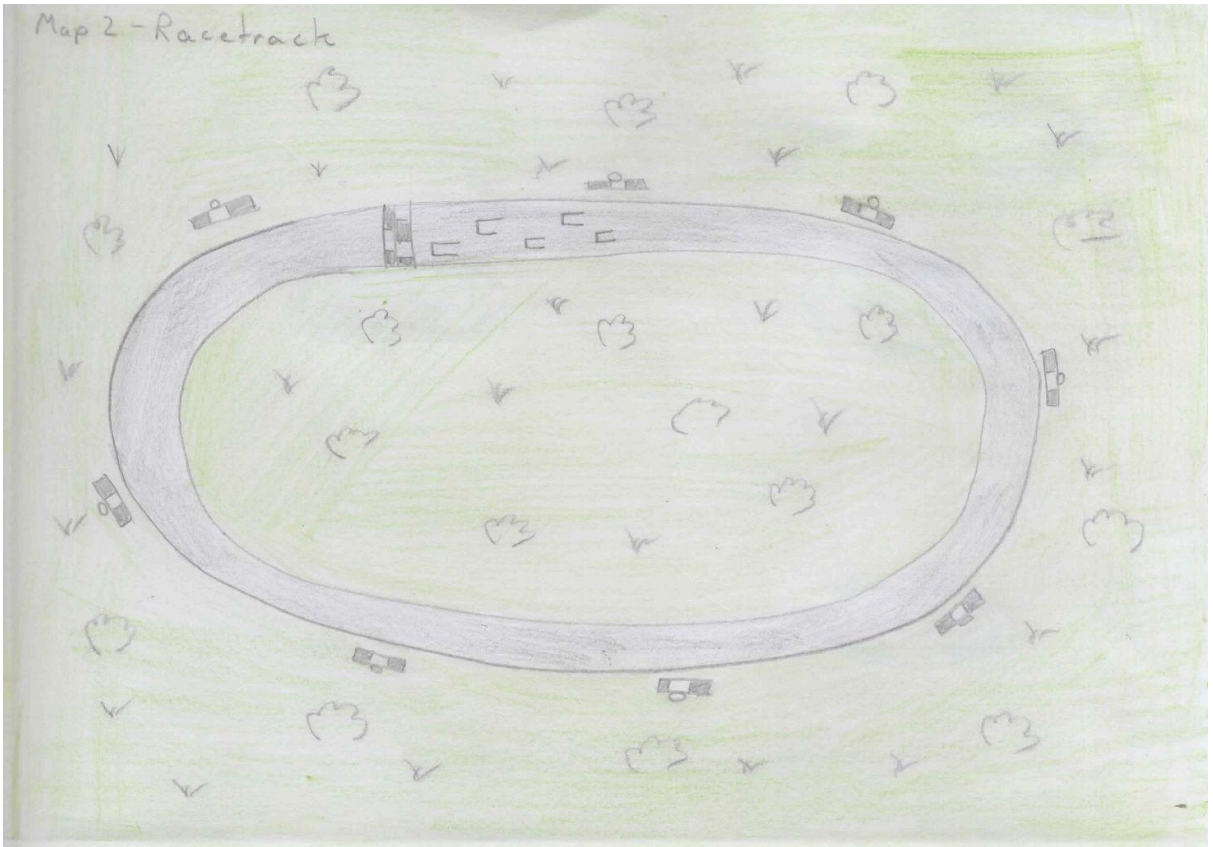
Hand-drawn designs



retro rampage - anthony guy

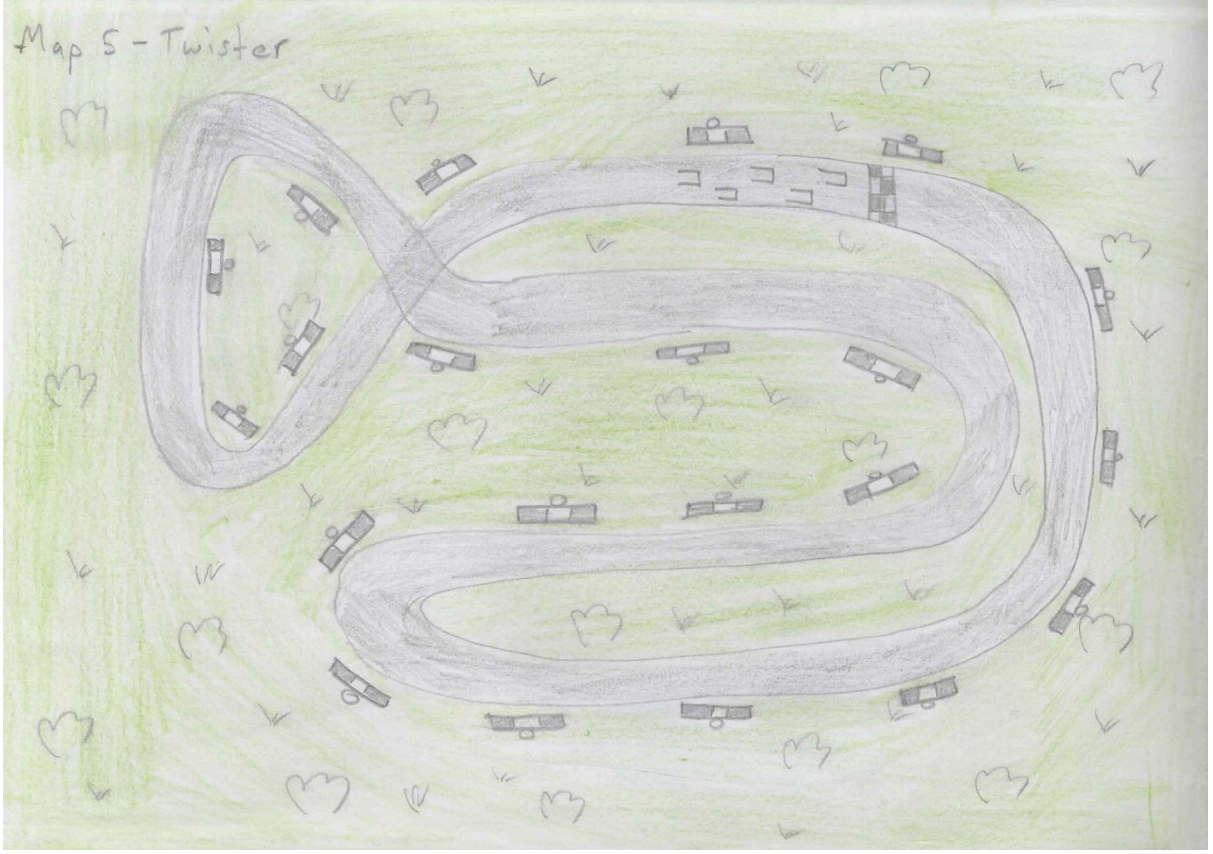

race settings
AI amount
1
Map - 1. figure 8, Sumer
P1 vehicle - Sports car
P2 vehicle - Race car
Player difficulty
Spectators
on
AI difficulty
2
AI color
Red
Map Season
random
Laps
3
Start Game

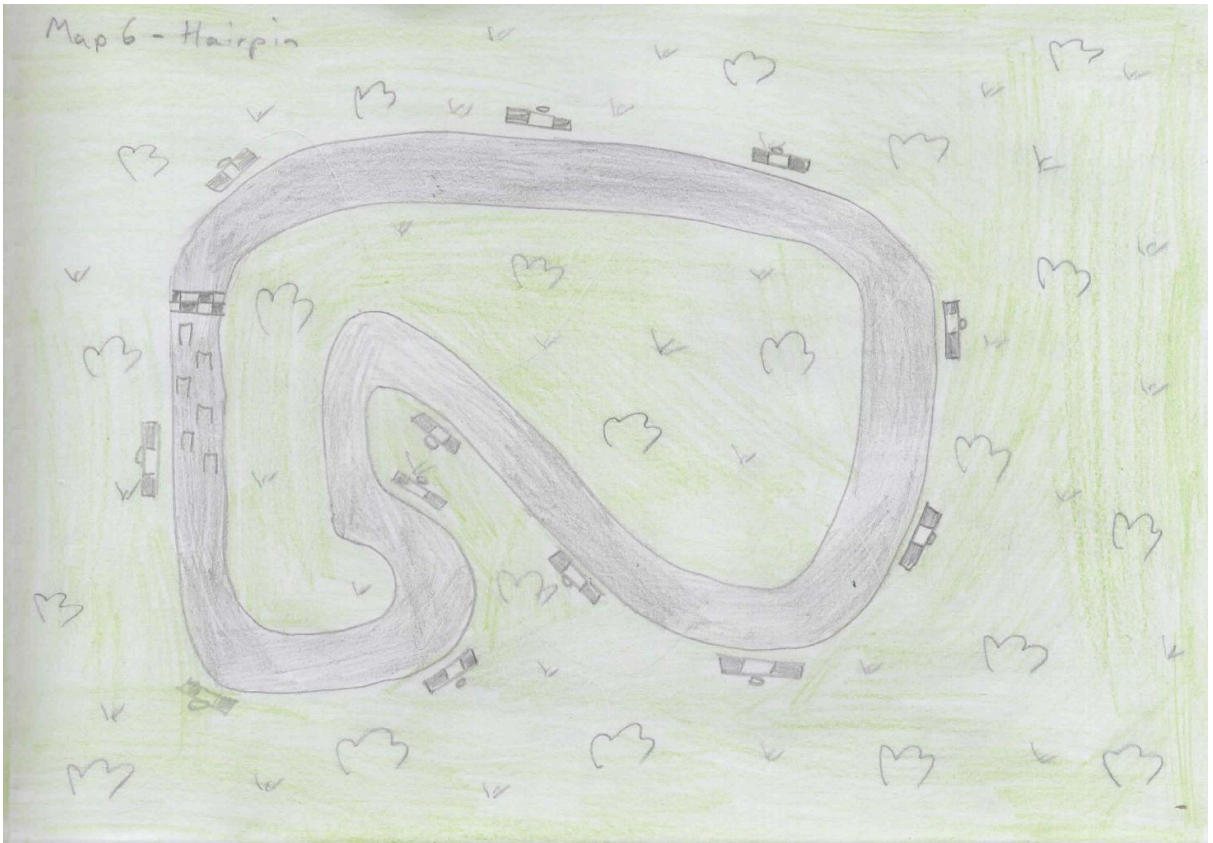

Map 1 - figure 8


Map 2 - Racetrack


Map 3 - Clover leaf


Map 4 - Mountain Hills

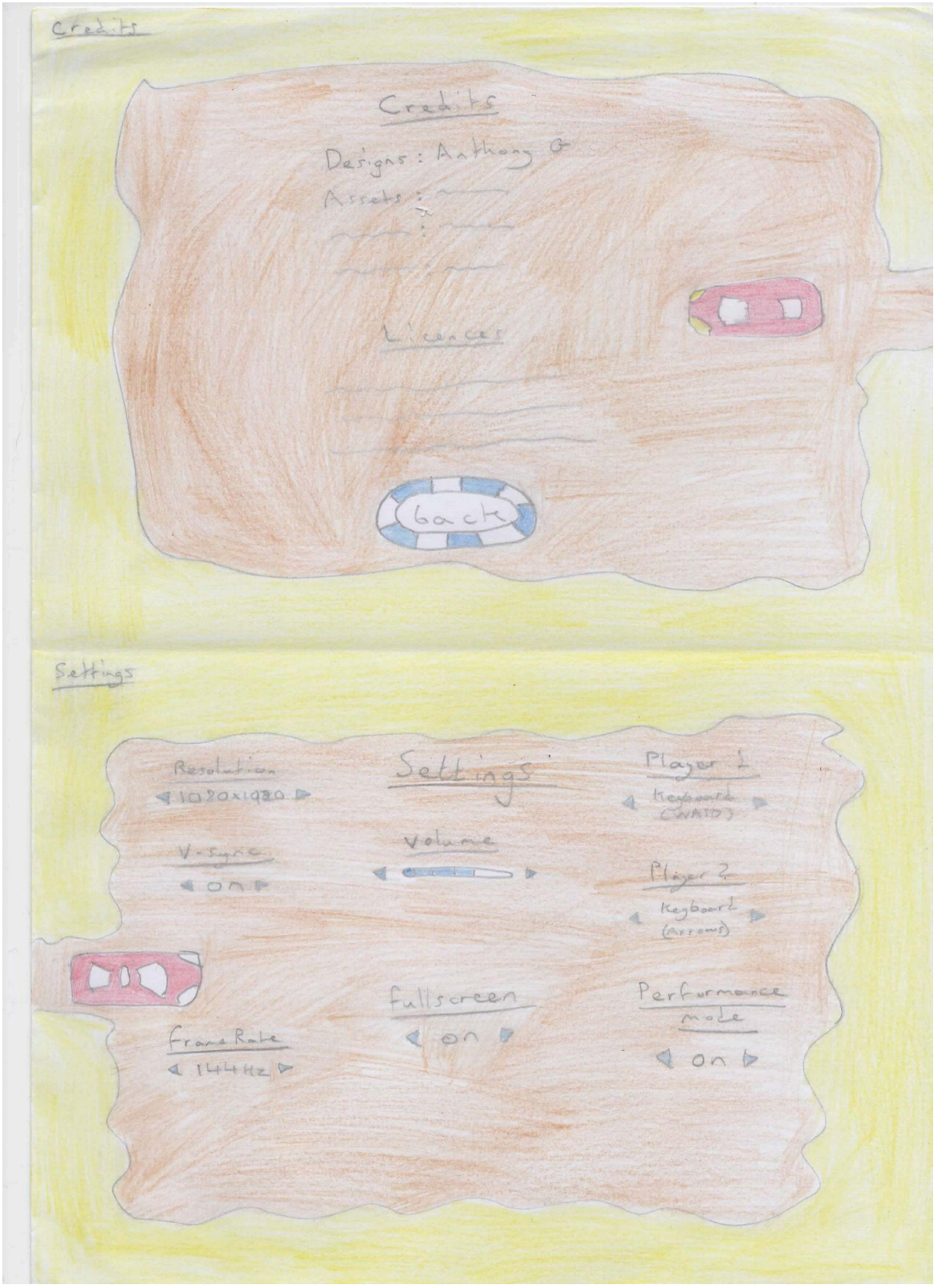

Map 5 - Twister


Map 6 - Hairpin

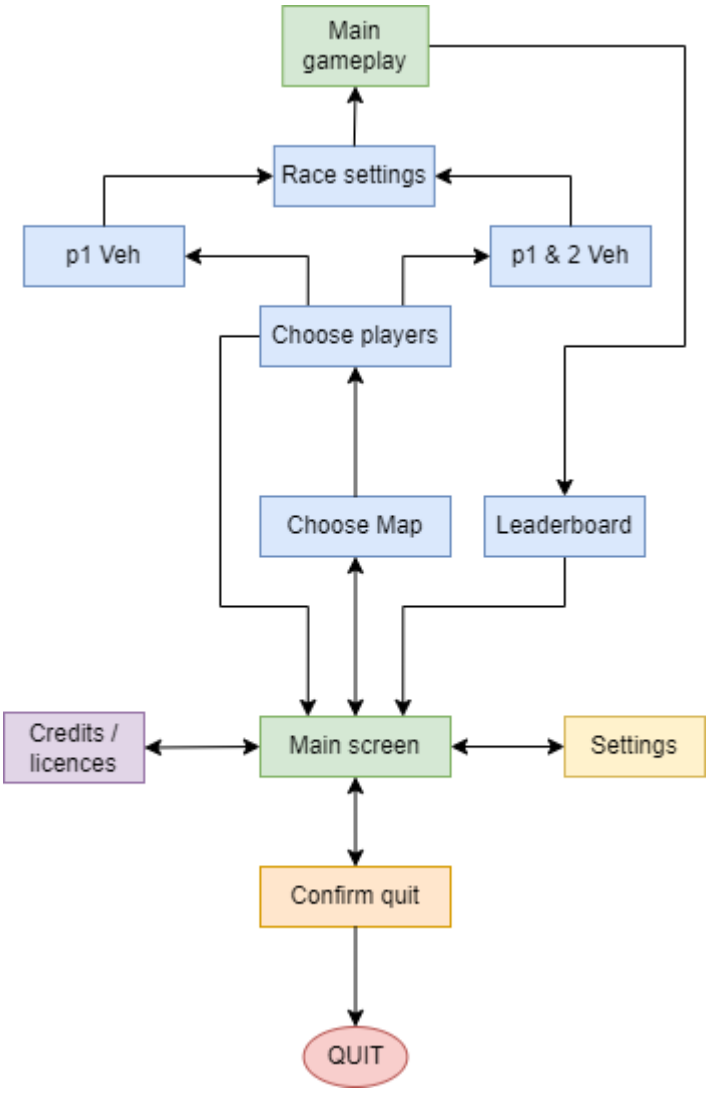

Map 7 - aerodrome

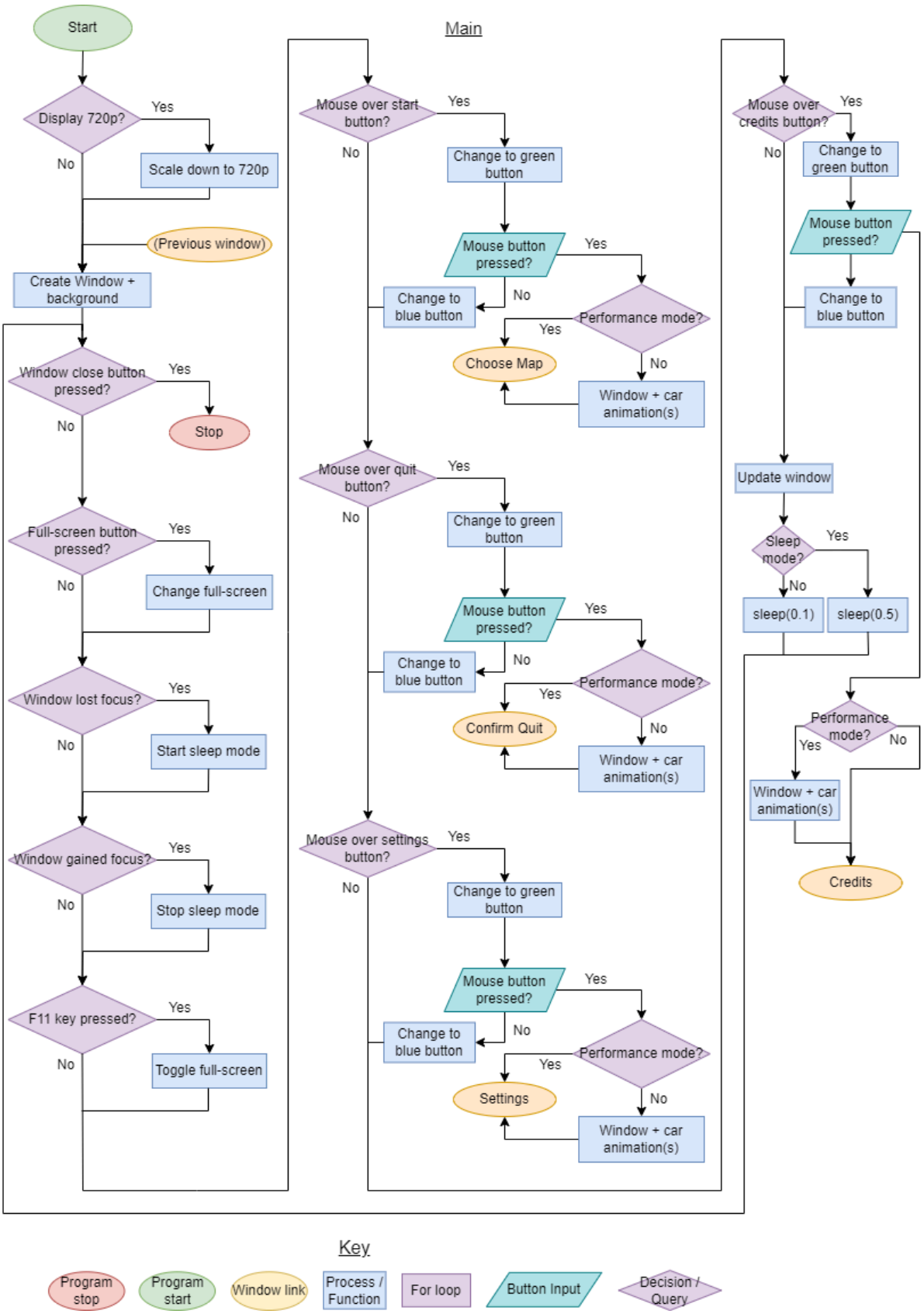

Map 8 - Hatchet
Map 9 - Ripple

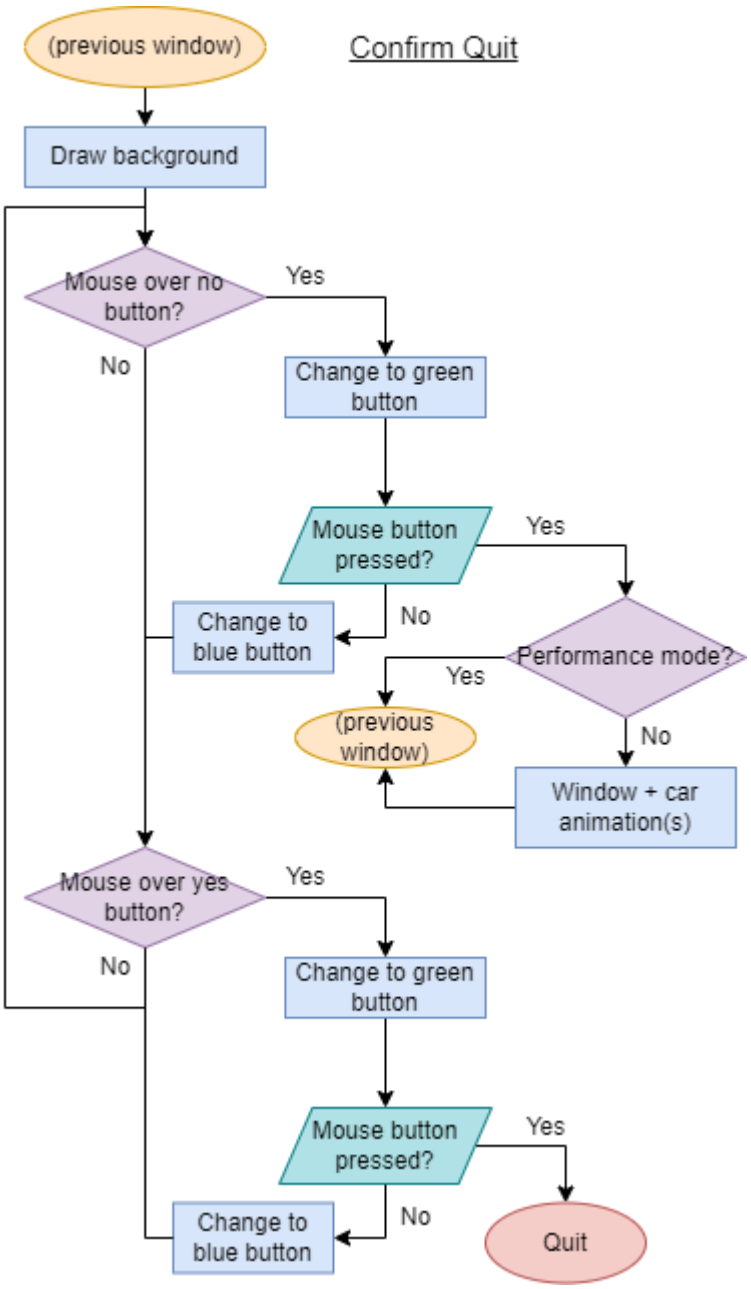

Map 10 - Teardrop
Leaderboard
Quit
Finished!
Settings
1st - [Player 1 name]
2nd - [Player 2 name]
3rd - Dave
4th - Bob
5th - Jimmy
6th - Andrew
7th - Thomas
8th - Sam
Map - 1. figure 8
AI difficulty -
Player difficulty -
Season - summer
AI amount - 5
Laps - 3
Main menu
retro rampage - anthony Guy

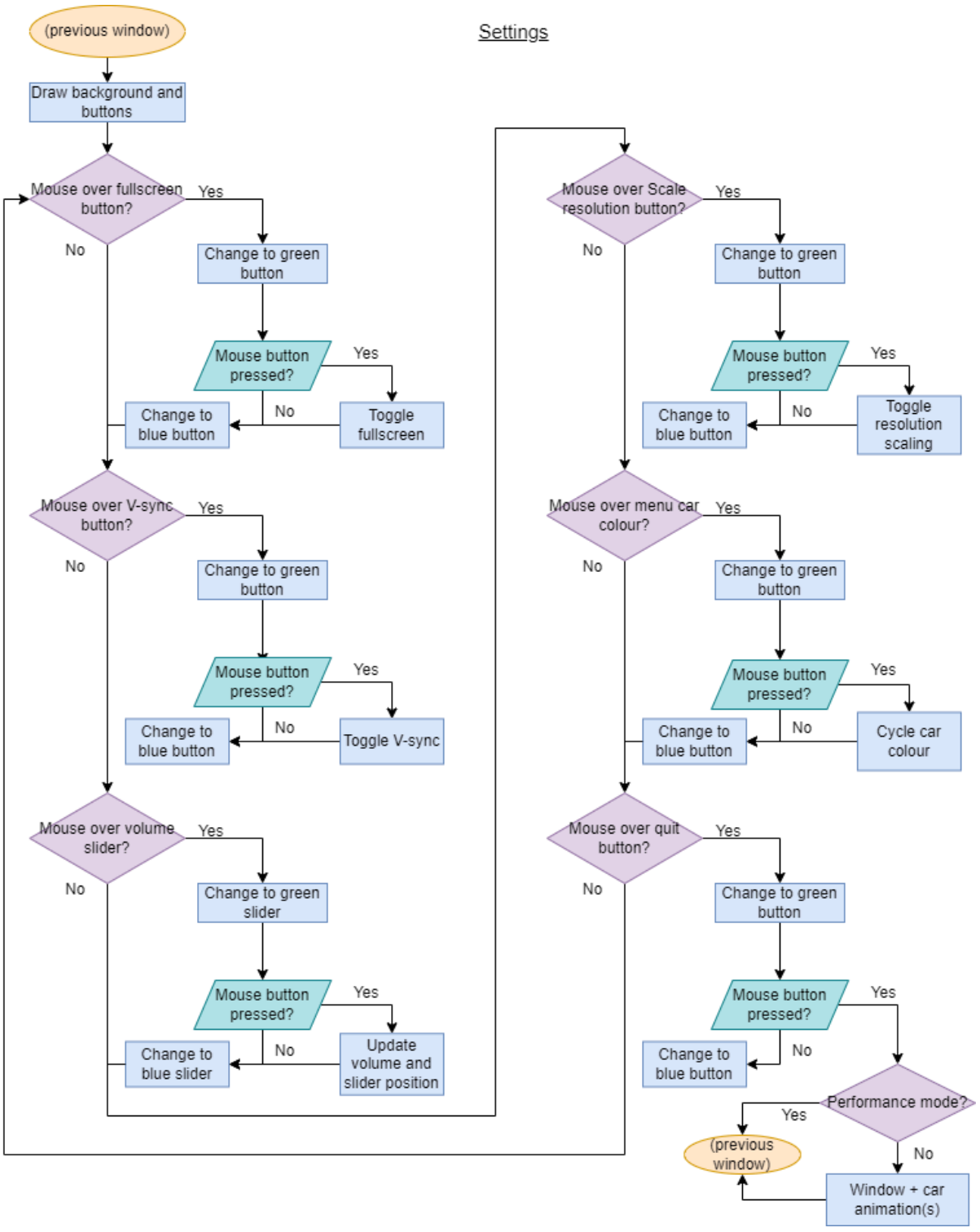

Credits
Credits
Designs : Anthony
Assets :
Licences
back
Settings
Settings
Resolution
1080x1920
V-Sync
on
Volume
Player 1
Keyboard
(WASD)
Player 2
Keyboard
(Arrows)
Fullscreen
on
Performance
mode
on
Frame Rate
144Hz

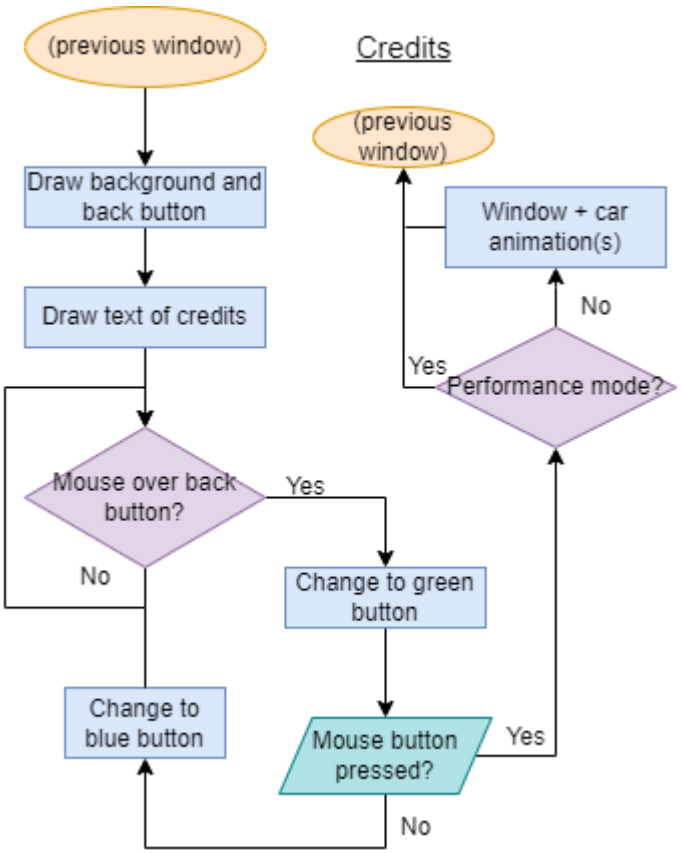
Flow Charts

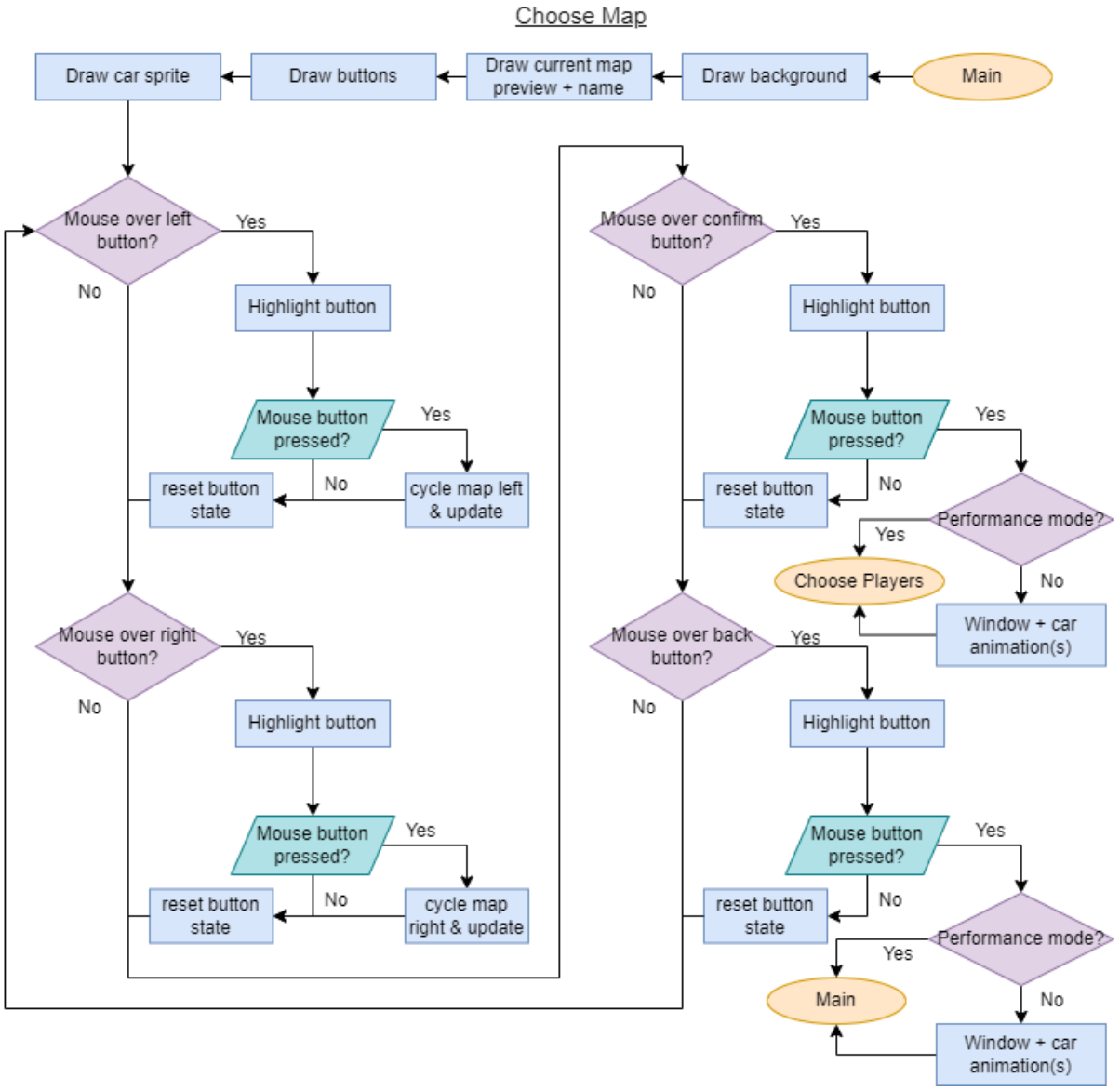

Main gameplay
Race settings
p1 Veh
p1 & 2 Veh
Choose players
Choose Map
Leaderboard
Credits / licences
Main screen
Settings
Confirm quit
QUIT

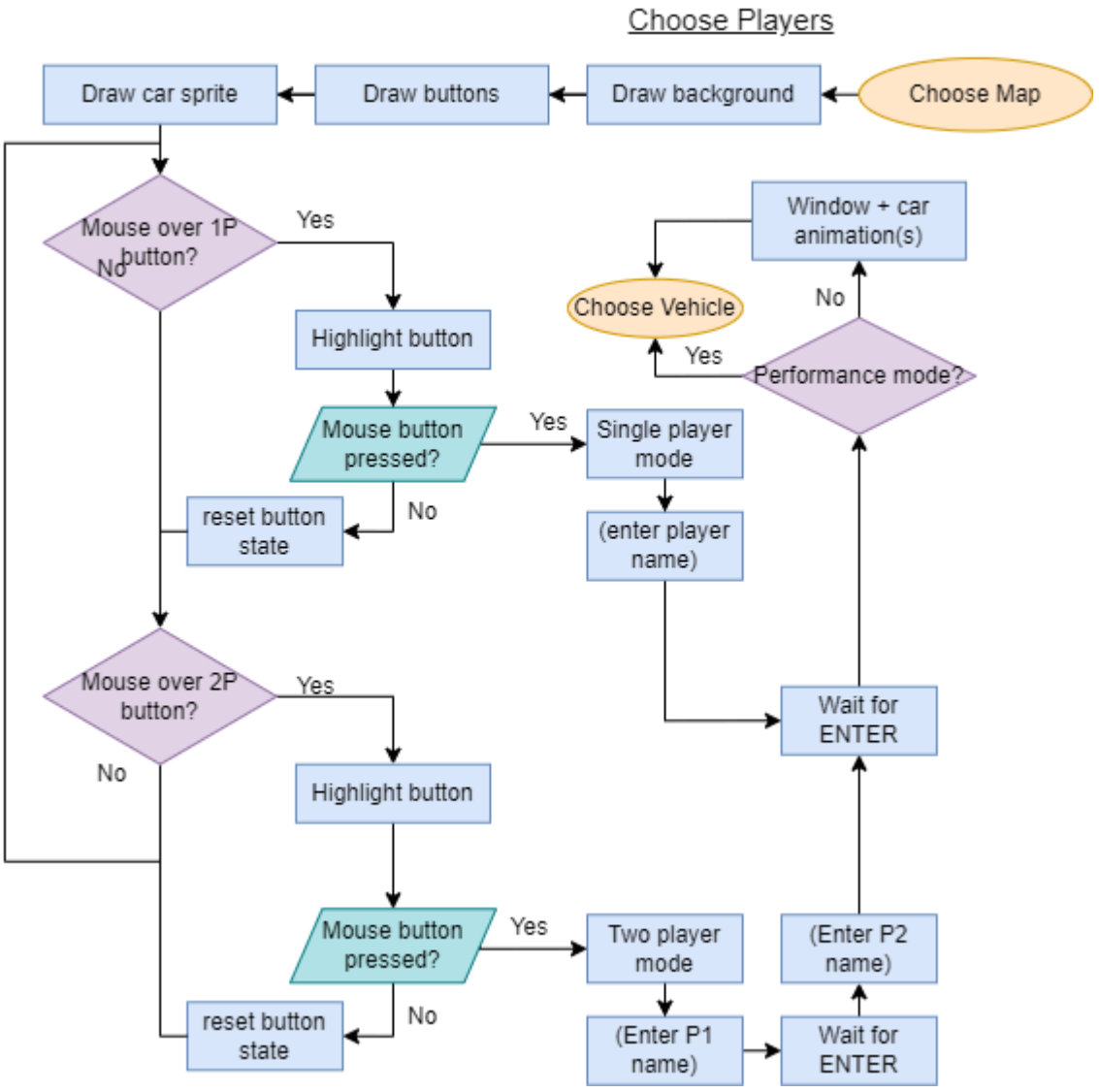

Main
Start
Display 720p?
Yes
No
Scale down to 720p
(Previous window)
Create Window + background
Window close button pressed?
Yes
No
Stop
Full-screen button pressed?
Yes
No
Change full-screen
Window lost focus?
Yes
No
Start sleep mode
Window gained focus?
Yes
No
Stop sleep mode
F11 key pressed?
Yes
No
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

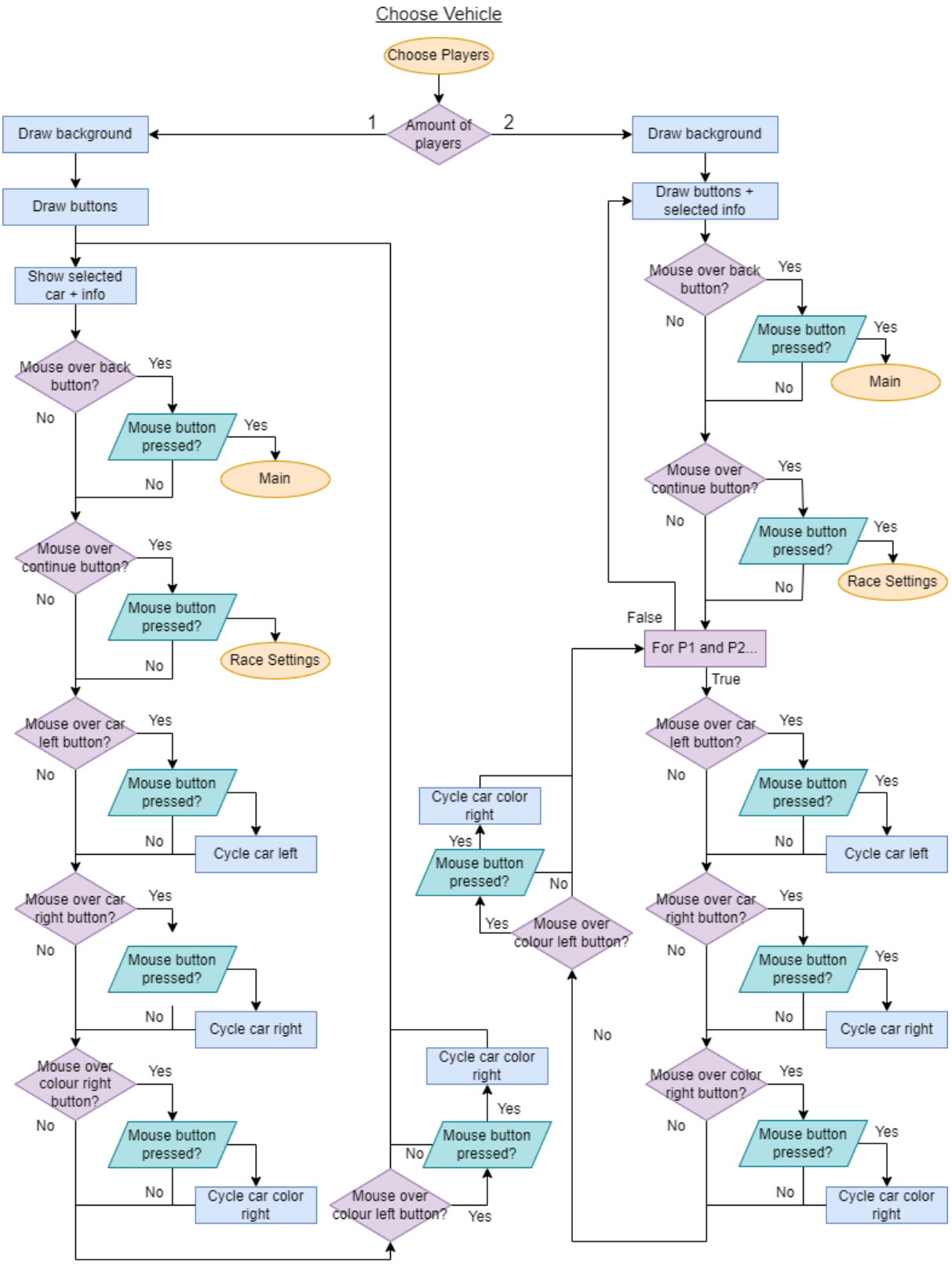

Confirm Quit
(previous window)
Draw background
Mouse over no button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Change to blue button
Performance mode?
Yes
No
(previous window)
Window + car animation(s)
Mouse over yes button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No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

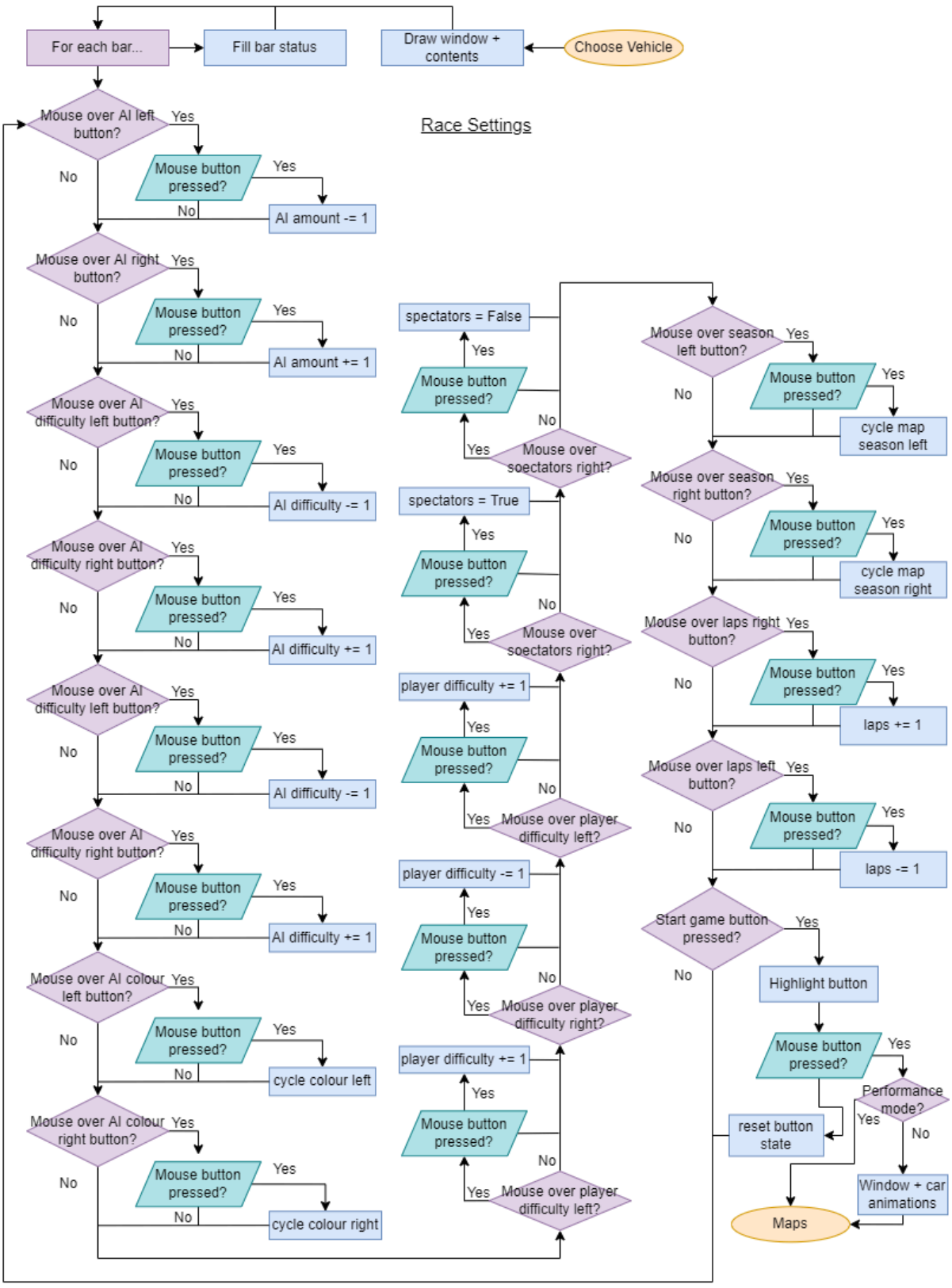

Settings
(previous window)
Draw background and buttons
Mouse over fullscreen button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Toggle fullscreen
Change to blue button
Mouse over V-sync button?
Yes
No
Change to green button
Mouse button pressed?
Yes
No
Toggle V-sync
Change to blue button
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

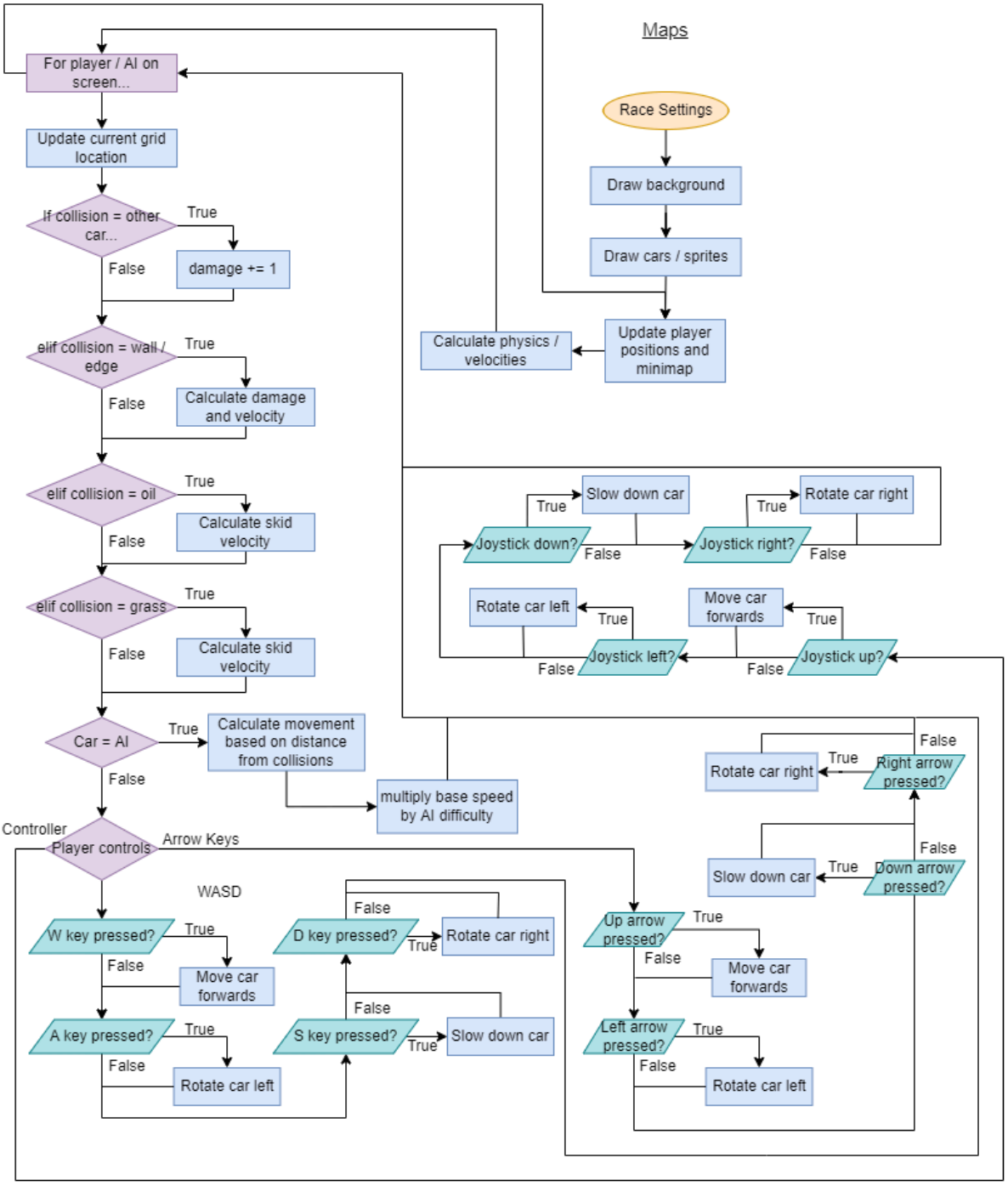

Credits
(previous window)
Draw background and back button
Draw text of credits
Mouse over back button?
Yes
No
Change to green button
Change to blue button
Mouse button pressed?
Yes
No
(previous window)
Window + car animation(s)
No
Yes
Performance mode?

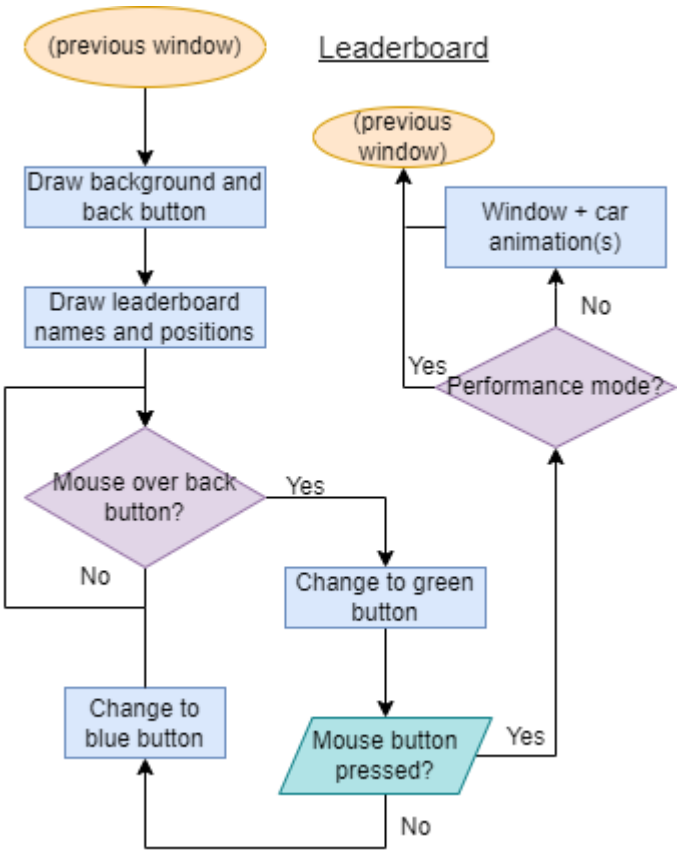

Choose Map
Draw car sprite
Draw buttons
Draw current map preview + name
Draw background
Main
Mouse over left button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button state
cycle map left & update
Mouse over right button?
Yes
No
Highlight button
Mouse button pressed?
Yes
No
reset button 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


For each bar...
Fill bar status
Draw window + contents
Choose Vehicle
Race Settings
Mouse over AI left button?
Yes
No
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
spectators = True
Mouse over soectators right?
player difficulty += 1
player difficulty -= 1
Mouse over player difficulty left?
Mouse over player difficulty right?
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


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
False
multiply base speed by AI difficulty
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, 4k, 1440p, 1080p, 720p, 480p. However, users will only be able to access resolutions that their screen can support. For example, if a user is running the game on a 1080p, they will not have the option for 4k or 1440p. Alternatively, if the game is running on a screen capable of 4k the game will allow all of the options. Manual resolutions are only in standard resolutions however auto mode can support any resolution.

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 1 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 colour, AI vehicle, power ups, laps, player starting positions (based on how many are playing). 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

Each map has its own class within map_loader.py. The game takes each layer of the map and overlays them onto one surface so that the game can run efficiently. The first thing to happen in every frame is the background is drawn, then each car is drawn over the map. 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 minimise 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 that would be quite noticeable due to the fade animation showing the map before the game starts.

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 Tests

For testing my program, I wrote several automated tests that check for things like incorrect variables and functions that don't work correctly. For example, a test that is testing the Car movement will call Car.move(x, y) and tell it to move to 100, 100… If the car's position is not 100, 100 the test will know that Car.move(x, y) does not work, meaning that a lot of coverage can be automated using these tests, and even though it took more time to make these tests it will have saved a lot more time overall by manually checking the same parameters as many times as the automated tests were used. The tests use nose2 to run and also report back a coverage percent of the entire project showing where the gaps in testing are.

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

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	NPC vehicles and avoidance / track traversal	Ensure that the NPCs work properly and manoeuvres the track(s) efficiently. Also check that the NPC colour and vehicle settings match
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 etc…

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 (Global + Classes)

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
FPS	Global	Int	60	Frames per second
Intro_screen	Global	Bool	True	Enables the intro screen
Countdown	Global	Bool	True	Enables countdown on race start
Load_settings	Global	Bool	True	Enables loading settings
Game_end	Global	Bool	False	Used as a flag if a player quits a race
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
Window_screenshot	Global	Pygame surface	(pygame surface)	Holds screenshot of screen for pause effect
Window_sleep	Global	Boolean	False	True if tabbed out
Secondary_window	Global	Pygame surface	(pygame surface)	Used for fade animations
current_window	Global	String	(empty)	Stores current menu window
Display_scaling	Global	Boolean	False	Set scaling between render and screen
Players	Global	List	[ ]	Stores player data
Selected_player	Global	List	[ ]	Stores selected player data
Player_amount	Global	Int	0	Amount of players
Npc_amount	Global	Int	0	Amount of NPCs
Map	Global	String	snake	Holds selected map
Total_laps	Global	Int	3	Total amount of laps
Current_lap	Global	Int	0	Holds current lap number
Race_time	Global	Int	0	Time of race in ms
Music_loop	Global	Bool	True	Enables music threading

Player_positions	Global	List	[ ]	Holds current player positions
Npc_names	Global	List	[[(name), (bool)], …]	Stores Npc names and if they are used
controllers	Global	List	[ ]	Stores connected controllers
controls	Global	List	[ ]	Stores used controls
controller_prompts	Global	List	[ ]	Stores controllers for visual prompts
Npc_force_veh	Global	Int	0	Vehicle to force Npcs into
Npc_force_colour	Global	None	None	Colour to force Npcs into
lightning_frames	Global	List	[ ]	Stores lightning animation frames
smoke_frames	Global	List	[ ]	Stores smoke animation frames
repair_frames	Global	List	[ ]	Stores repair animation frames
map_preview_size	Global	Tuple	974, 600	Stores size for map preview
map_preview_pos	Global	Tuple	Int, int	Stores position for map preview
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
button_trigger	Global	Bool	False	Used to only trigger a button per press
selected_text_entry	Global	Int	0	Used for entering player names
current_song	Global	String	(empty)	Stores current song
clock	Global	pygame.time.Clock()	(pygame clock)	Used to manage the speed of updates
music_thread	Global	Thread	thread	Threads music loops
powerups	Global	Bool	True	Enables powerups
Game_paused	Global	Bool	False	Stores if game is paused
global_car_rotation_speed	Global	Int	1	Speed of car rotations
global_car_move_speed	Global	Int	4	Speed of car movement
checkpoint_triggers	Global	List	[ ]	Stores triggers for checkpoints
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

recorded_keys	Global	List	[ ]	Sotres keys pressed for making Npc paths
controls	Player	String	None	Selected controls
default_controls	Player	String	None	Default controls
id	Player	Int	(player number)	Stores player id number
name	Player	String	(empty)	Player name
start_pos	Player	Int	None	Player start position
veh_colour	Player	tuple [int, int, int]	None	Player vehicle colour
veh_image	Player	Pygame surface	None	Player vehicle image
veh_name	Player	String	None	Player vehicle name
image	MenuCar	Pygame surface	(pygame surface)	Menu vehicle surface
pos_x	MenuCar	Int	CENTRE[0]	Menu car start pos x
pos_y	MenuCar	Int	CENTRE[1]	Menu car start pos y
prev_rect	MenuCar	Pygame rect	Pygame rect	Menu car prev rect
rect	MenuCar	Pygame rect	Pygame rect	Menu car rect
rotation	MenuCar	Int	0	Menu car rotation

scale	MenuCar	Tuple [int, int]	71, 131	Menu car size
scaled_rect	MenuCar	Pygame rect	None	Menu car scaled rect
size	MenuCar	Tuple [int, int]	Image size	Menu car image size
speed	MenuCar	Int	menu_car_speed	Menu car move speed
player	Car	Player	player	Contains Player
is_player	Car	Bool	player.is_player	If player then True
id	Car	Int	player.id	Id of car
start_pos	Car	Int	player.start_pos	Starting position
_origin_pos	Car	Tuple [int, int]	(map start pos)	Original start pos
_origin_rotation	Car	Int	(map start rotation)	Original start rotation
vehicle	Car	Str	player.veh_name	Vehicle
_move_speed	Car	Int	0	Car move speed
_rotation_speed	Car	Int	0	Car rotation speed
max_speed	Car	Int	0	Car maximum speed
max_rotation_speed	Car	Int	0	Car maximum rotation speed
durability	Car	Int	0	Car durability
colour	Car	Tuple [int, int, int]	player.veh_colour	Car colour
image_dir	Car	String	(image dir)	Car image directory
image	Car	Pygame surface	(image)	Car image
_origin_img	Car	Pygame surface	(image)	Original car image
_dmg_img	Car	Pygame surface	None	Car damage

damage	Car	Int	0	Car damage
size	Car	Tuple [int, int]	0, 0	Car image size
rect	Car	Pygame rect	Pygame rect	Car image rect
pos_x	Car	Int	0	Car pos x
pos_y	Car	Int	0	Car pos y
rotation	Car	Int	0	Car rotation
mask	Car	Pygame mask	None	Car collision mask
mask_overlap	Car	Bool	False	Car mask collision
mask_area	Car	Tuple [int, int]	0, 0	Car mask area
mask_size	Car	Tuple [int, int]	0, 0	Car mask size
collision	Car	String	(empty)	Car collision
_up	Car	Pygame key	None	Car up controls
_down	Car	Pygame key	None	Car down control
_left	Car	Pygame key	None	Car left control
_right	Car	Pygame key	None	Car right controls
input_type	Car	String	(empty)	Car input
controller	Car	Pygame controller	None	Car controller
_allow_forwards	Car	Bool	True	Allows forward
_allow_reverse	Car	Bool	True	Allows reverse
_pressed_keys	Car	List	[ ]	List of inputs to move car
_boost_timeout	Car	Int	0	Boost duration timeout
bullet_penalty	Car	Int	0	Bullet duration
_bullet_damage	Car	Int	0	Damage at time of bullet
_current_speed	Car	Int	0	Car speed

_boost_frames	Car	List	(boost frames)	Boost animation frames
_boost_ani_frame	Car	Int	-1	Boost animation frame
_smoke_ani_frame	Car	Int	-1	Smoke animation frame
_repair_ani_frame	Car	Int	-1	Repair animation frame
_ani_frame	Car	Int	None	Animation frame num
_ani_frame_rect	Car	Pygame rect	None	Animation rect
lightning_animation	Car	Bool	False	True if lightning strike
lightning_target	Car	Bool	False	True if target
lightning_indicator	Car	Pygame surface	(lightning powerup)	Small indicator next to name
_lightning_frame	Car	Int	None	Frame of ani
name	Car	String	player.name	Player name
_name_rect	Car	Pygame rect	(pygame rect)	Car name rect
_lap_halfway	Car	Bool	False	Lap halfway collision trigger
laps	Car	Int	0	Car laps
checkpoint_count	Car	Int	-1	Amount of checkpoints
checkpoint_time	Car	Int	0	Time hit point
_collision_sound	Car	Bool	False	Only allows single collision sound
player	NpcCar	Player	player	Contains Player
is_player	NpcCar	Bool	player.is_player	If player then True
id	NpcCar	Int	player.id	Id of car
start_pos	NpcCar	Int	player.start_pos	Starting position

_origin_pos	NpcCar	Tuple [int, int]	(map start pos)	Original start pos
_origin_rotation	NpcCar	Int	(map start rotation)	Original start rotation
_move_speed	NpcCar	Int	0	Car move speed
_rotation_speed	NpcCar	Int	0	Car rotation speed
vehicle	NpcCar	Str	player.veh_name	Vehicle
max_speed	NpcCar	Int	0	Car maximum speed
max_rotation_speed	NpcCar	Int	0	Car maximum rotation speed
durability	NpcCar	Int	0	Car durability
colour	NpcCar	Tuple [int, int, int]	player.veh_colour	Car colour
image_dir	NpcCar	String	(image dir)	Car image directory
image	NpcCar	Pygame surface	(image)	Car image
_origin_img	NpcCar	Pygame surface	(image)	Original car image
_dmg_img	NpcCar	Pygame surface	None	Car damage
damage	NpcCar	Int	0	Car damage
size	NpcCar	Tuple [int, int]	0, 0	Car image size
rect	NpcCar	Pygame rect	Pygame rect	Car image rect
pos_x	NpcCar	Int	0	Car pos x
pos_y	NpcCar	Int	0	Car pos y
rotation	NpcCar	Int	0	Car rotation
mask	NpcCar	Pygame mask	None	Car collision mask
mask_overlap	NpcCar	Bool	False	Car mask collision

mask_area	NpcCar	Tuple [int, int]	0, 0	Car mask area
mask_size	NpcCar	Tuple [int, int]	0, 0	Car mask size
collision	NpcCar	String	(empty)	Car collision
collision_time	NpcCar	Int	0	Time of collision
_boost_timeout	NpcCar	Int	0	Boost duration timeout
bullet_penalty	NpcCar	Int	0	Bullet duration
_bullet_damage	NpcCar	Int	0	Damage at time of bullet
_current_speed	NpcCar	Int	0	Car speed
_boost_frames	NpcCar	List	(boost frames)	Boost animation frames
_boost_ani_frame	NpcCar	Int	-1	Boost animation frame
_smoke_ani_frame	NpcCar	Int	-1	Smoke animation frame
_repair_ani_frame	NpcCar	Int	-1	Repair animation frame
_ani_frame	NpcCar	Int	None	Animation frame num
_ani_frame_rect	NpcCar	Pygame rect	None	Animation rect
lightning_animation	NpcCar	Bool	False	True if lightning strike
lightning_target	NpcCar	Bool	False	True if target
lightning_indicator	NpcCar	Pygame surface	(lightning powerup)	Small indicator next to name
_lightning_frame	NpcCar	Int	None	Frame of ani
allow_forward	NpcCar	Bool	True	Allow forward
allow_back	NpcCar	Bool	True	Allow reverse
allow_left	NpcCar	Bool	True	Allow left
allow_right	NpcCar	Bool	True	Allow right
move_forward	NpcCar	Bool	False	Move forward
move_back	NpcCar	Bool	False	Move back

move_left	NpcCar	Bool	False	Move left
move_right	NpcCar	Bool	False	Move right
reverse	NpcCar	Bool	False	Reverse
reverse_time	NpcCar	Int	0	Time of reverse
_move_rect_radius	NpcCar	Int	90	Rect positioning
_move_rect_offset	NpcCar	Int	28	Rect positioning
_move_layer_offset	NpcCar	Int	35	Rect positioning
_avoid_rect_radius	NpcCar	Int	60	Rect positioning
_avoid_rect_offset	NpcCar	Int	15	Rect positioning
_avoid_layer_offset	NpcCar	Int	32	Rect positioning
movements_obj	Car	Tuple	(rects)	Holds ‘feelers’
avoidance_obj	Car	Tuple	(rects)	Holds ‘feelers’
name	Car	String	player.name	Player name
_name_rect	Car	Pygame rect	(pygame rect)	Car name rect
_lap_halfway	Car	Bool	False	Lap halfway collision trigger
laps	Car	Int	0	Car laps
checkpoint_count	Car	Int	-1	Amount of checkpoints
checkpoint_time	Car	Int	0	Time hit point
prev_checkpoint_position	Car	Tuple [Int, Int]	_origin_pos	Used to reset to checkpoint
prev_checkpoint_rotation	Car	Int	_origin_rotation	Used to reset to checkpoint
collision_sound	Car	Bool	False	Only allows single collision sound
prev_window	main	String	None	Stores previous window

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

Functions

Name	Inputs	Outputs	Explanation
save_settings	None	Sets settings	Saves / sets settings
load_settings	None	Loads settings	Loads settings
(Object) __init__	None	Creates class	Initialises Object class
(Player) __init__	player	Creates variables for class	Initialises variables for Player
(Player) load_player	None	Sets variables	Sets defaults for player
(Player) load_npc	None	Sets variables	Sets defaults for NPC
(Player) update_image	None	Updates image	Updates car image
(MenuCar) __init__	None	Creates variables for class	Initialises variables for MenuCar
(MenuCar) rotate	degree	Rotates car	Rotates menu car
(MenuCar) move	x, y	Moves car	Moves menu car
(MenuCar) draw	update=False	Draws car	Draws car on screen
(MenuCar) animate	direction, bg	Animates movement	Moves car to direction and scrolls screen
(Car) __init__	None	Creates variables for class	Used to initialise variables for Car class
(Car) set_move_speed	speed	Sets move speed	Sets move speed
(Car) set_rotation_speed	speed	Sets rotation speed	Sets rotation speed
(Car) set_controls	control	Sets controls	Sets controls
(Car) check_checkpoints	checkpoint_rectangles	Check checkpoint collisions	Check checkpoint collisions

(Car) check_track_collisions	track_mask	Checks track collisions	Checks track collisions
(Car) check_car_collision	sprite	Checks car collisions	Checks car collisions
(Car) move	x, y	Movement of car asset	Used to move car around and off screen
(Car) rotate	degree	Rotation of car asset	Used to rotate car asset
(Car) power_up	ver	Powerup action	Powerup actions
(Car) check_inputs	None	Check inputs	Checks inputs
(Car) draw	update	Draw car on screen	Draw car on screen
(Car) update	None	Updates	Updates
(NpcCar) __init__	None	Sets variables for npc	Sets variables for npc
(NpcCar) set_move_speed	speed	Sets move speed	Sets move speed
(NpcCar) set_rotation_speed	speed	Sets rotation speed	Sets rotation speed
(NpcCar) check_checkpoints	checkpoint_rectangles	Checks checkpoint collisions	Checks checkpoint collisions
(NpcCar) check_track_collisions	track_mask	Checks track collisions	Checks track collisions
(NpcCar) check_car_collision	sprite	Checks car collisions	Checks car collisions
(NpcCar) move	x, y	Moves	Moves car
(NpcCar) rotate	degree	Rotates	Rotates car
(NpcCar) rotate_rect	rect, angle, radius	Rotates around point	Rotates rects for ‘feelers’

(NpcCar) reset_to_checkpoint	None	Resets position to checkpoint	Resets position to checkpoint
(NpcCar) decide_movement	None	Sets variables	Decides movement NPC will take
(NpcCar) take_movement	None	Moves NPC	Moves NPC based on decided action
(NpcCar) power_up	ver	Powerup action	Powerup action
(NpcCar) draw	surf=Window	Draws car	Draws car
(NpcCar) update	None	Updates car	Updates car
get_car_positions	None	Gets car positions	Gets car positions
get_map_preview	None	Gets map preview	Gets map preview
main_window	curr_bg, pad_x, pad_y	Draw main window on screen	Draw all assets for main window on screen
choose_map_window	curr_bg, pad_x, pad_y	Draw choose map window on screen	Draw all assets for main window on screen
choose_players_window	curr_bg, pad_x, pad_y	Draw choose players window on screen	Draw all assets for main window on screen
choose_players_window_2	curr_bg, pad_x, pad_y	Draw choose players 2 window on screen	Draw all assets for main window on screen
choose_players_window_3	curr_bg, pad_x, pad_y	Draw choose players 3 window on screen	Draw all assets for main window on screen
choose_vehicle_window	curr_bg, pad_x, pad_y	Draw choose vehicle window on screen	Draw all assets for main window on screen
choose_vehicle_window_2	curr_bg, pad_x, pad_y	Draw choose vehicle 2 window on screen	Draw all assets for main window on screen
choose_vehicle_window_3	curr_bg, pad_x, pad_y	Draw choose vehicle 3 window on screen	Draw all assets for main window on screen
race_settings_window	curr_bg, pad_x, pad_y	Draw race settings window on screen	Draw all assets for main window on screen

confirm_quit_window	curr_bg, pad_x, pad_y, surf=Window	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
tutorial_window	curr_bg, pad_x, pad_y	Draw tutorial window on screen	Draw all assets for main window on screen
settings_window	curr_bg, pad_x, pad_y, surf=Window	Draw settings window on screen	Draw all assets for main window on screen
leaderboard_window	curr_bg, pad_x, pad_y	Draw leaderboard window on screen	Draw all assets for main window on screen
gameplay_gui	Player_list, game_countdown_timer, lap_timer	Draw gameplay gui window on screen	Draw all assets for main window on screen
paused_window	None	Draw paused window on screen	Draw all assets for main window on screen
menu_background	Top=False, right=False, bottom=False, left=False	Background image	Stitches multiple assets into one background
animate_window	curr_bg, pad_x, pad_y	Animates window animation	Moves window in direction
controller_popup	surf=Window	Draws controller popups	Draws controller names
tile	X, y, material, ver, grid=True, scale=tile_scale, surf=Window, update=True	Draw tile ground assets	Draw tile assets on screen

draw_triangle	Pos, direction, width=18, height=18, colour=WHITE, border=None, border_width=4, surface=Window	Draws triangle	Draws triangle
draw_slider	Pos, size, value, min_value, max_value, center_x=True, fill_colour=RED_CAR, surface=Window, border_width=2, border_radius=7	Draws slider	Draws slider
draw_text	X, y, text, font_type, colour, size	Text on screen	Draws text on screen
render_key	size=50	Renders key	Renders key
draw_controls	X, y, surface=Window, return_rect=False	Draws controls	Draws controls
draw_asset	asset, pos, rotation, scale, return_rect=False	Asset on screen	Draws given asset on screen
fade_to_black	speed=12, show_loading=False, paused=False, car=None	Fades window to black	Fades window to black
fade_from_black	speed=12, show_loading=False, paused=False, window=” ”	Fades to window from black	Fades window from black
cycle_veh_right	player	Cycles vehicle right	Cycles vehicle right
cycle_veh_left	player	Cycles vehicle left	Cycles vehicle left
cycle_veh_colour_right	player	Cycles vehicle colour right	Cycles vehicle colour right

cycle_veh_colour_left	player	Cycles vehicle colour left	Cycles vehicle colour left
cycle_controls_left	control	Cycles controls left	Cycles controls left
cycle_controls_right	control	Cycles controls right	Cycles controls right
increase_position	Player, amount=1, reset=False	Increases player position	Increases player position
decrease_position	Player, amount=1, reset=False	Decreases player position	Decreases player position
rand_vehicle	None	Randomises vehicle	Randomises vehicle
rand_colour	None	Randomises colour	Randomises colour
menu_music	None	Background music	Plays and loops random background tracks
game_music	Stage, leaderboard=False	Game music	Plays and loops game music based on lap number
play_sound	event	Event sound	Plays sound effect according to the event type
game_music_loop	None	Game music loop	Used by thread to check music
menu_music_loop	None	Menu music loop	Used by thread to check music
increase_resolution	None	Increases resolution	Increases resolution
decrease_resolution	None	Decreases resolution	Decreases resolution
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
controller_added	None	Adds controller	Adds a controller
controller_removed	instance_id	Removes controller	Removes a controller by id
short_controller_name	name	Shortens / normalises controller name	Shortens a controller's name
get_mouse_pos	None	pos	Scales mouse position to window
game	None	Main Gameplay	Every gameplay process is within this function
main	None	Main Menus	Every menu process is within this function

Classes

Name	Functions	Explanation
Object	__init__	Used to group surfs, rects and masks that relate
Player	__init__, load_defaults, update_image	Holds player data
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, 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
json	(all)	Used to save and load settings.json file
math	Cos, sin, radians, ceil, floor	Used for rounding up and down
random	randint	Used to generate a random integer between x and y
threading	Thread	Used to create threads eg. for animations and music
time	sleep	Used for idle processing
audio_loader	(all sounds)	Used to find directory path to requested audio file
font_loader	(all fonts)	Used to find directory path to requested font file
image_loader	(all assets)	Used to find directory names for requested asset
map_loader	Map objects	Holds class for each map with layer, checkpoint, start positions ect.
pygame	(all)	Game engine

Libraries

Audio_loader.py

Variable Names	Function Names
assets	menu_track(track_no)
effects	game_track(track_no)
music	explosion(length, ver)
menu_music	alarm(ver)
gameplay_music	bleep(ver)
screams	impact(ver)
explosions	interaction(ver)
exp_cluster	menu(ver)
exp_double	negative(category, ver)
exp_long	pause_sound(ver, out=False)
exp_medium	positive(ver)
exp_odd	machine_gun(ver)
exp_short	
exp_shortest	
exp_various	
general	
alarms	
bleeps	
impacts	
interactions	
menu_sounds	
negative_sounds	
pause_sounds	
positive_sounds	

weapons	
machine_guns	

Font_loader.py

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

Image_loader.py

Variable Names	Function Names
assets	animation(ver, frame, car_num='' or int)
animations	controller()
cars	controller_button(ver)
objects	car(colour, ver, small=False)
powerups	car_damage(ver, dmg)
tiles	traffic_light(ver)
	power_up(ver, active=True)
	tile(material, ver)

Map_loader.py

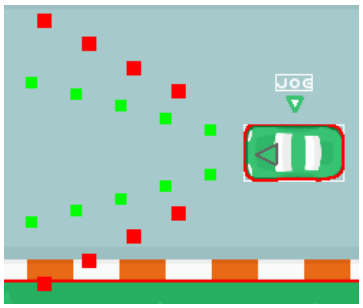
Variable Names	Classes
index	Racetrack
objs	Snake
	DogBone
	Hairpin

Development

Implementation & Improvements

NPC Implementation

Originally the NPCs were implemented by repeating inputs from a previously recorded lap. While this did work in the terms that an NPC could easily make multiple laps, it proved very constricting and limiting to the functionality of the NPCs as well as very time consuming to create new maps since 90+ lap recordings had to be made for each map. Because of this, I decided to implement a ‘self aware’ NPC that does not need any recordings and can traverse any map, use power-ups and actively avoid other vehicles.



To improve the functionality, realism and make adding new maps much much easier, the NPCs were later updated to feature real time object avoidance and track traversal using several rects as ‘feelers’ to know if the car needs to turn or not. As shown on the left, the red squares are used by the NPC to detect the track edges and know if the car needs to turn, and the green squares are used to detect any vehicles in front of the NPC that the car can then avoid. These are only shown when Debug is True.

To make debugging easier, when Debug is True, the desired actions and allowed movements of each NPC can be seen by the triangles showing the direction the NPC is facing in green, if the NPC decides it cannot move in a direction then a red arrow is shown, and any allowed movements are shown with a grey arrow. This means that the movements of each NPC can be clearly seen at any time. For example, we can tell that the car on the left is trying to move forward and back based on the grey arrows, but is not allowed to move left. Even though the red arrow is facing up, we know that the NPC counts it as left as the green arrow is the direction of the NPC.

Using this updated method of NPC movements, the NPCs can appear much more realistic to the point that they are almost the exact same class as Car but with the avoidance and movement automation. By not having a fixed movement speed and path, the NPCs can have the same damage implementation that they players have using varying speeds based on the amount of damage and durability the vehicle has left. Furthermore, the NPCs are also now able to trigger power ups that will repair their damage or penalise them based on the type of powerup they hit as well as animations for each powerup like the players do.

Not only does the newer implementation of the NPCs boast more accuracy to the players, damage implementation, powerups, animations and much easier map implementation but when implementing these features more code was removed than added. This is because the new implementation also allowed multiple features to be heavily optimised in the process, such as vehicle and track collisions and map handling as well as reducing overall project size and removing several files.
The position of all ‘feelers’ for each NPC can also be heavily customised and tuned to give different avoidance characteristics such as turning earlier for tracks or not seeing other vehicles until they are closer, which could prove useful in the future.

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.

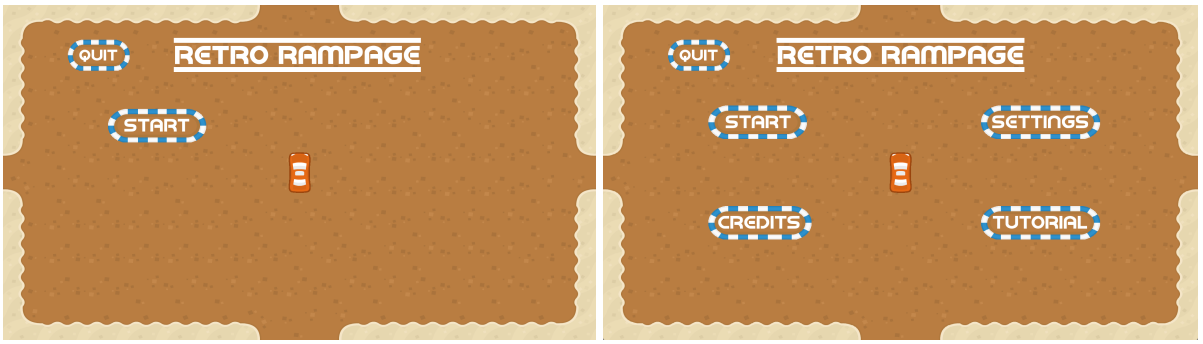
For window animations, the function to draw the assets can take a pad_x and pad_y argument which will offset the asset positions by the given amount based on their default position.

All animations for the windows are called from a single function; animate_window(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 such as having a surf input for effects.

Main window



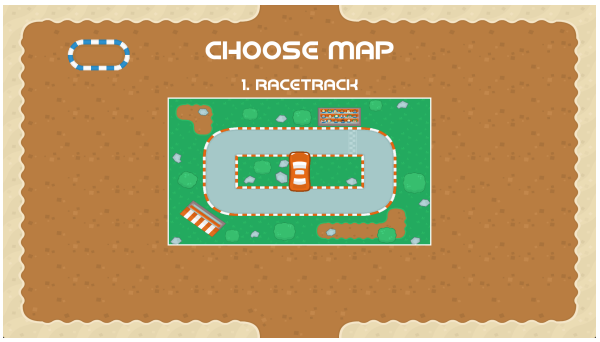
Code

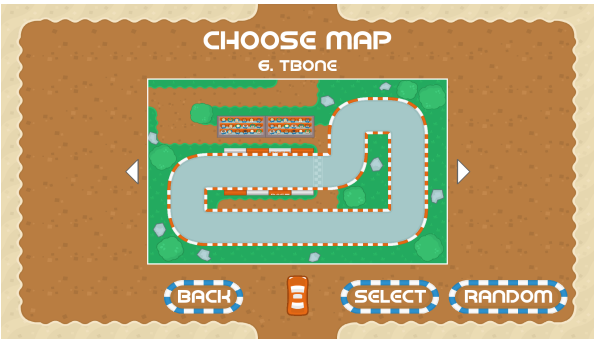
```
def main_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))  # Display the current bg

    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 + 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)

    x = pad_x + 1220
    y = pad_y + 648
    tile(x, y, 'dirt road', 76, grid=False)  # Tutorial 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, 'Tutorial', WHITE, 70)

    x = pad_x + 340
    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)
```

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. The window has 5 buttons for the player to use and each button apart from the tutorial window has an animated transition. The tutorial window has a fade transition instead because all 4 directions for the car were already used.

Choose map window



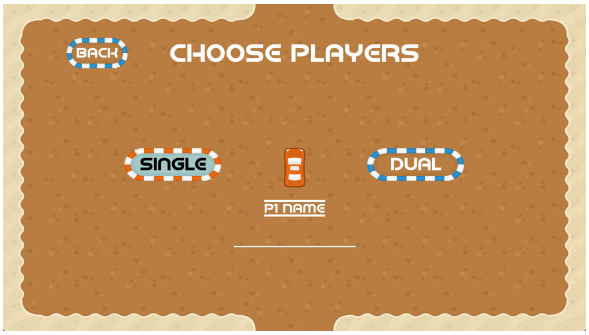


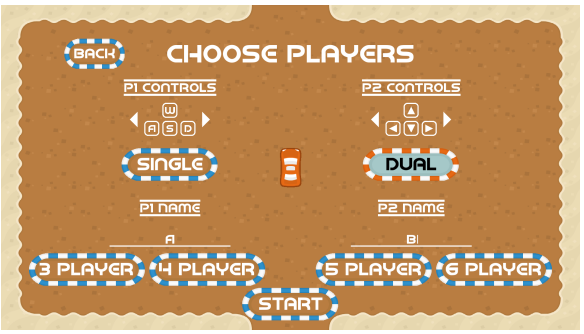
Code & Description

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.

```python
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)
```

Choose players window





Code

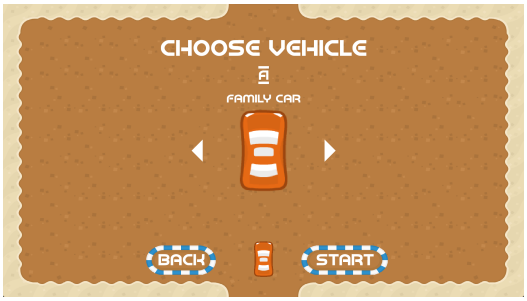
```
def choose_players_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    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:
        player = Players[0]
        # Controls
        x = pad_x + CENTRE[0]
        y = pad_y + 240
        draw_text(x, y, 'P1 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 840, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 1080, pad_y + 380), 'right', width=25, height=50)

        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, player.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 player.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)

    elif Player_amount >= 2:
        # P1
        player = Players[0]
        # Controls
        x = pad_x + 560
        y = pad_y + 240
        draw_text(x, y, 'P1 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 440, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 680, pad_y + 380), 'right', width=25, height=50)

        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, player.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)
        x = pad_x + WIDTH - 876
        y = pad_y + HEIGHT - 255
        tile(x, y, 'dirt road', 76, grid=False)  # 5 player 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, '5 Player', WHITE, 70)
    if len(controllers) >= 4:
        x = pad_x + WIDTH - 477
        y = pad_y + HEIGHT - 255
        tile(x, y, 'dirt road', 76, grid=False)  # 6 player 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, '6 Player', WHITE, 70)

    if Player_amount != 1:
        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:
        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:
        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)
    elif Player_amount == 2:
        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)
    elif Player_amount == 3:
        x = pad_x + 93
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 3 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '3 Player', BLACK, 70)
    elif Player_amount == 4:
        x = pad_x + 492
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 4 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '4 Player', BLACK, 70)
    elif Player_amount == 5:
        x = pad_x + WIDTH - 876
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 5 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '5 Player', BLACK, 70)
    elif Player_amount == 6:
        x = pad_x + WIDTH - 477
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 6 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '6 Player', BLACK, 70)
```

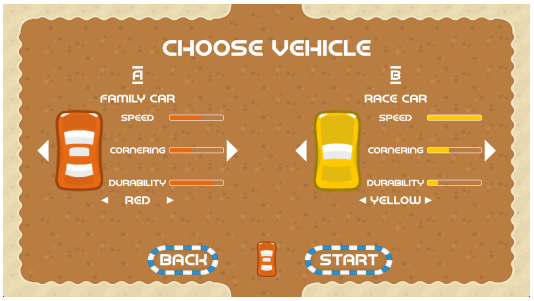
```
    # Enter name prompt
    if selected_text_entry != 1 and not player.name:
        if (pygame.time.get_ticks() // 1060) % 2:
            draw_text(x, y + 115, 'Enter name', V_LIGHT_GREY, 50)
        else:
            draw_text(x, y + 115, 'Enter name', LIGHT_GREY, 50)

    # P2
    player = Players[1]
    x = pad_x + 1360
    y = pad_y + 240
    # Controls
    draw_text(x, y, 'P2 controls', WHITE, 50, bar=True)
    rect = draw_controls(x, y + 140, player.controls, return_rect=True)
    if player.controls == 'controller':
        rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
        Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                    (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
    elif player.controls in controllers:
        draw_text(x, y + rect.height + rect.centerx + 42,
                  short_controller_name(player.controls.get_name()), WHITE, 32)
    if type(player.controls) == str:
        draw_triangle((pad_x + 1240, pad_y + 380), 'left', width=25, height=50)
        draw_triangle((pad_x + 1480, pad_y + 380), 'right', width=25, height=50)

    y = pad_y + CENTRE[1] + 100
    # 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, player.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 player.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)

for player in Players:
    if player.controls in controllers:
        rect = draw_text(100, HEIGHT-108, 'To unbind a controller, press ',
                         WHITE, 32, center_x=False, return_rect=True)
        Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('b')), (34, 34)),
                    (100 + rect.width, HEIGHT-109))

# START BUTTON
if Player_amount == 1 and Players[0].name.strip() and Players[0].controls != 'controller' or \
        Player_amount >= 2 and Players[0].name.strip() and Players[0].controls != 'controller' and \
        Players[1].name.strip() and Players[1].controls != 'controller':
    x = pad_x + 800
    y = pad_y + 940
    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 len(controllers) >= 1:
    x = pad_x + 93
    y = pad_y + HEIGHT - 255
    tile(x, y, 'dirt road', 76, grid=False)  # 3 player 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, '3 Player', WHITE, 70)
if len(controllers) >= 2:
    x = pad_x + 492
    y = pad_y + HEIGHT - 255
    tile(x, y, 'dirt road', 76, grid=False)  # 4 player 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, '4 Player', WHITE, 70)
if len(controllers) >= 3:
```

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 preview of the player controls and show the user(s) which inputs they have to use for their car. Depending on the amount of controllers connected to the game, the window will also show options for 3 - 6 player modes at the bottom of the screen.

Choose vehicle window





Code

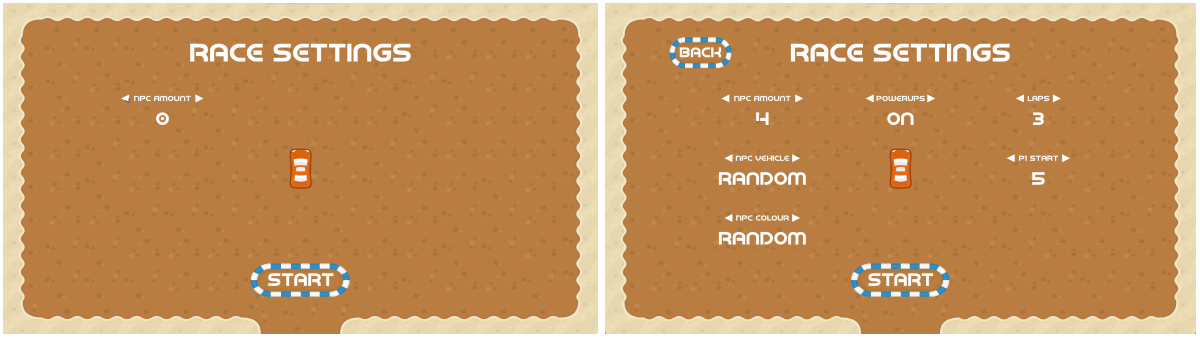
```
def choose_vehicle_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    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:
        player = Players[0]
        draw_text(pad_x + CENTRE[0], pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + CENTRE[0], pad_y + 325, player.veh_name, WHITE, 50)

        p1_veh_rect = player.veh_image.get_rect()
        draw_triangle((pad_x + CENTRE[0] - 170 - p1_veh_rect.width, pad_y + CENTRE[1]), 'left', width=40, height=80)
        Window.blit(player.veh_image, (pad_x + CENTRE[0] - 215 - p1_veh_rect.width // 2,
                                       pad_y + CENTRE[1] - p1_veh_rect.height // 2))
        draw_triangle((pad_x + CENTRE[0] + 170 + p1_veh_rect.width, pad_y + CENTRE[1]), 'right', width=40, height=80)

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + CENTRE[0], pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

    elif Player_amount >= 2:
        # 1P VEHICLE
        player = Players[0]
        pos_x = CENTRE[0] // 2 + 10
        draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

        # 2P VEHICLE
```

```
    player = Players[1]
    pos_x = CENTRE[0] + CENTRE[0] // 2 - 10
    draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
    draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

    if player.veh_colour == RED_CAR:
        text = 'Red'
    elif player.veh_colour == YELLOW_CAR:
        text = 'Yellow'
    elif player.veh_colour == GREEN_CAR:
        text = 'Green'
    elif player.veh_colour == BLUE_CAR:
        text = 'Blue'
    else:
        text = 'Black'

    draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
    elif player.veh_name == 'Sports Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    elif player.veh_name == 'Luxury Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    elif player.veh_name == 'Truck':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
    elif player.veh_name == 'Race Car':
        draw_slider((pad_x + pos_x + 115, pad_y + 408),
                    (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 528),
                    (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        draw_slider((pad_x + pos_x + 115, pad_y + 648),
                    (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
```

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



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)

    if Player_amount != 6:
        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 + 960
        y = pad_y + 345
        if powerups:
            text = 'On'
        else:
            text = 'Off'
        draw_text(x, y, text, WHITE, 70)  # Powerups option
        draw_text(x, y - 48, 'Powerups', WHITE, 30)
        draw_triangle((x - 100, y - 34), 'left', width=25, height=25)
        draw_triangle((x + 100, 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)

        if Npc_amount:
            x = pad_x + 511
            y = pad_y + 540
            if Npc_force_veh == 1:
                text = 'family car'
            elif Npc_force_veh == 2:
                text = 'sports car'
            elif Npc_force_veh == 3:
                text = 'luxury car'
            elif Npc_force_veh == 4:
                text = 'truck'
            elif Npc_force_veh == 5:
                text = 'race car'
            else:
                text = 'random'
            draw_text(x, y, str(text), WHITE, 70)  # NPC Vehicle option
            draw_text(x, y - 48, 'NPC Vehicle', WHITE, 30)
            draw_triangle((x - 110, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 110, y - 34), 'right', width=25, height=25)

            x = pad_x + 511
            y = pad_y + 735
            if Npc_force_colour == RED_CAR:
                text = 'red'
            elif Npc_force_colour == YELLOW_CAR:
                text = 'yellow'
            elif Npc_force_colour == GREEN_CAR:
                text = 'green'
            elif Npc_force_colour == BLUE_CAR:
                text = 'blue'
            elif Npc_force_colour == BLACK_CAR:
                text = 'black'
            else:
                text = 'random'
            draw_text(x, y, str(text), WHITE, 70)  # NPC Colour option
            draw_text(x, y - 48, 'NPC Colour', WHITE, 30)
            draw_triangle((x - 110, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 110, y - 34), 'right', width=25, height=25)

        x = pad_x + 1408
        y = pad_y + 540
        draw_text(x, y, str(Players[0].start_pos), WHITE, 70)  # P1 start option
        draw_text(x, y - 48, 'P1 Start', WHITE, 30)
        draw_triangle((x - 90, y - 34), 'left', width=25, height=25)
        draw_triangle((x + 90, y - 34), 'right', width=25, height=25)

        if Player_amount >= 2:
            x = pad_x + 1408
            y = pad_y + 735
            draw_text(x, y, str(Players[1].start_pos), WHITE, 70)  # P2 start option
            draw_text(x, y - 48, 'P2 Start', WHITE, 30)
            draw_triangle((x - 90, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 90, y - 34), 'right', width=25, height=25)

        if Player_amount >= 3:
            x = pad_x + 1408
            y = pad_y + 930
            draw_text(x, y, str(Players[2].start_pos), WHITE, 70)  # P3 start option
            draw_text(x, y - 48, 'P3 Start', WHITE, 30)
            draw_triangle((x - 90, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 90, y - 34), 'right', width=25, height=25)

        if Player_amount >= 4:
            x = pad_x + 1688
            y = pad_y + 540
            draw_text(x, y, str(Players[3].start_pos), WHITE, 70)  # P4 start option
            draw_text(x, y - 48, 'P4 Start', WHITE, 30)
            draw_triangle((x - 90, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 90, y - 34), 'right', width=25, height=25)

        if Player_amount == 5:
            x = pad_x + 1688
            y = pad_y + 735
            draw_text(x, y, str(Players[4].start_pos), WHITE, 70)  # P5 start option
            draw_text(x, y - 48, 'P5 Start', WHITE, 30)
            draw_triangle((x - 90, y - 34), 'left', width=25, height=25)
            draw_triangle((x + 90, y - 34), 'right', width=25, height=25)

    else:
        x = pad_x + 960
        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)

        x = pad_x + 960
        y = pad_y + 735
        if powerups:
            text = 'On'
        else:
            text = 'Off'
        draw_text(x, y, text, WHITE, 70)  # Powerups option
        draw_text(x, y - 48, 'Powerups', WHITE, 30)
        draw_triangle((x - 100, y - 34), 'left', width=25, height=25)
        draw_triangle((x + 100, y - 34), 'right', width=25, height=25)

        x = pad_x + 511
        y = pad_y + 345
        draw_text(x, y, str(Players[0].start_pos), WHITE, 70)  # P1 start option
        draw_text(x, y - 48, 'P1 Start', WHITE, 30)

        x = pad_x + 511
        y = pad_y + 540
        draw_text(x, y, str(Players[1].start_pos), WHITE, 70)  # P2 start option
```

```
    draw_text(x, y - 48, 'P2 Start', WHITE, 30)

    x = pad_x + 511
    y = pad_y + 735
    draw_text(x, y, str(Players[2].start_pos), WHITE, 70)  # P3 start option
    draw_text(x, y - 48, 'P3 Start', WHITE, 30)

    x = pad_x + 1408
    y = pad_y + 345
    draw_text(x, y, str(Players[3].start_pos), WHITE, 70)  # P4 start option
    draw_text(x, y - 48, 'P4 Start', WHITE, 30)

    x = pad_x + 1408
    y = pad_y + 540
    draw_text(x, y, str(Players[4].start_pos), WHITE, 70)  # P5 start option
    draw_text(x, y - 48, 'P5 Start', WHITE, 30)

    x = pad_x + 1408
    y = pad_y + 735
    draw_text(x, y, str(Players[5].start_pos), WHITE, 70)  # P6 start option
    draw_text(x, y - 48, 'P6 Start', WHITE, 30)
```

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

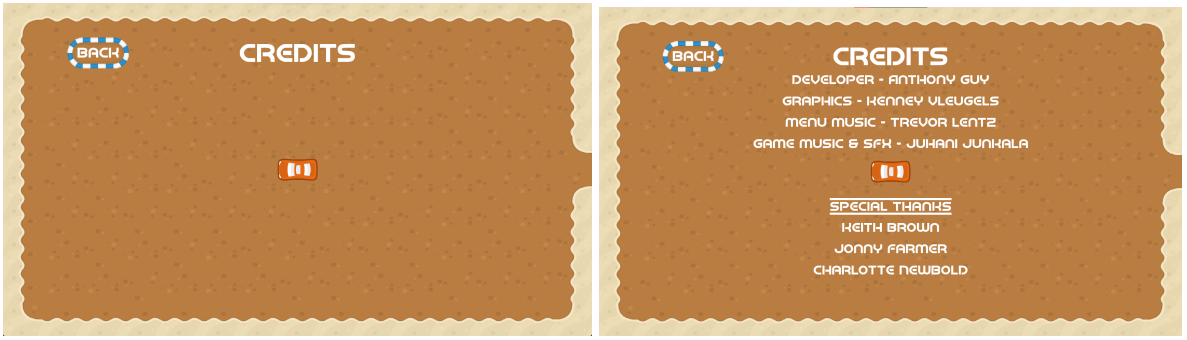
Code

```
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



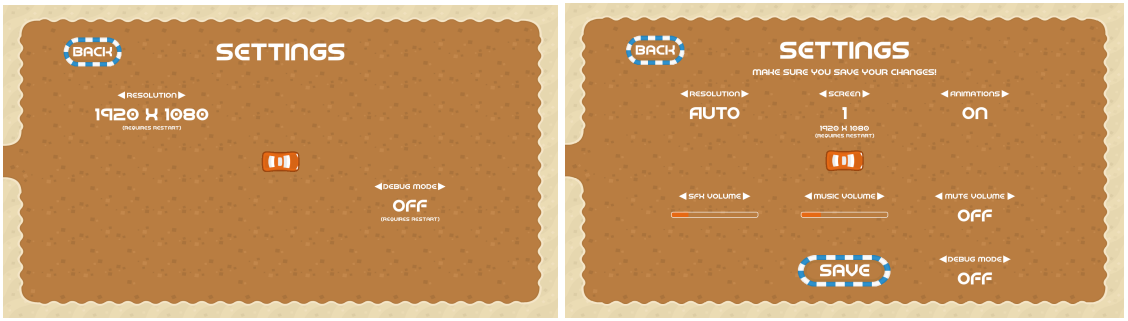
Code

```
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



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)
```

```
    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

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) - Causes the vehicle in first to get struck
4. Bullet (bullets) - Stops the player moving for a few seconds

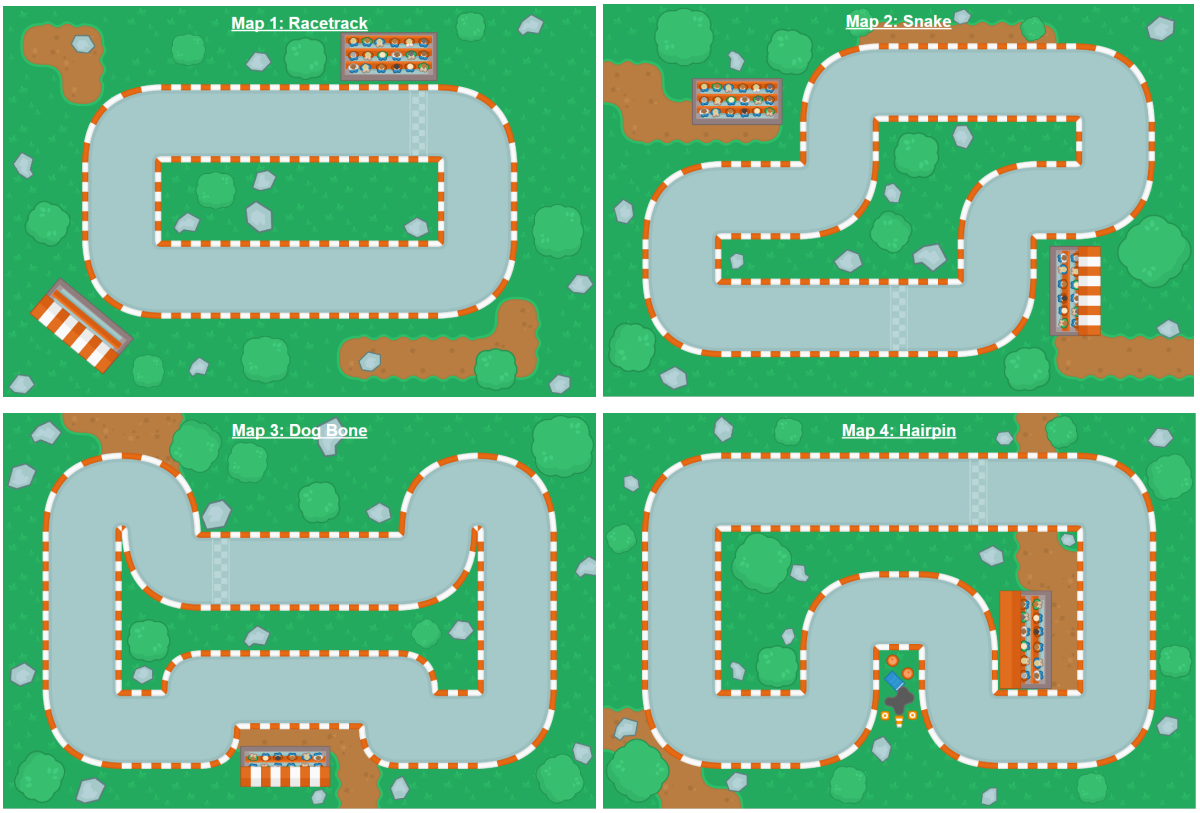
For every powerup, the influences are scaled based on how good the player's car is eg. If the player picks up a boost in a speed 1 car then it will last longer than a player in a speed 5 car. This also means that the bullet power up will penalise a faster car for longer etc. Once a power up spawns, there 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 since they constantly cycle.

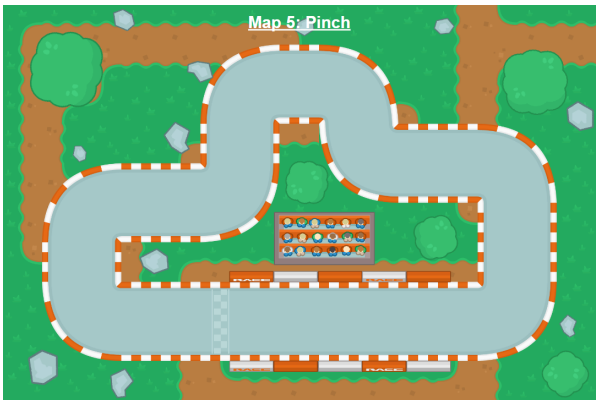
When an NPC crashes they will attempt to fix themselves by reversing after a random time. If they are stuck for more than 5s they will reset to the last checkpoint. Once a car has finished, a prompt will appear on the screen with the following: "[player name] has finished!" underneath will say "Game 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. In addition to the game showing a finishing prompt, for cars that have finished, once they have touched / crossed the finish line the vehicle will appear transparent, signalling that collisions for that vehicle have been turned off and the car is now 'ghosted' meaning they cannot block or hit other vehicles trying to finish. Furthermore when 'ghosted' a vehicle cannot advance their checkpoint status meaning if a vehicle carried on after the finish and hit the next checkpoint, without this feature the second car would finish first even

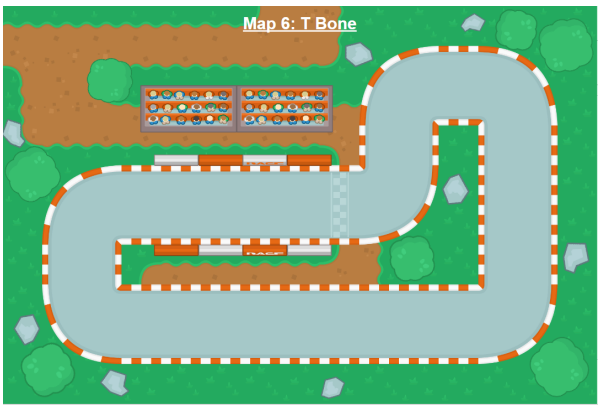
though the other vehicle hit the finish line first, ensuring that vehicles cannot ‘steal’ another’s win.

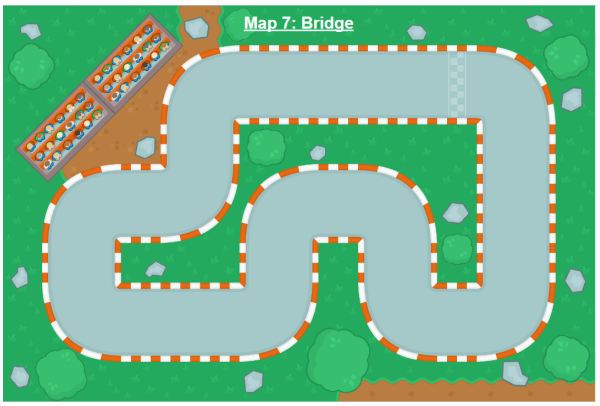
During the race a music loop will be played and every lap the music will play a sound then advance to the next ‘stage’ before looping back around to the first type to give a seamless effect of ever changing background music and adding more suspense to the game. For example, if the race lasted 6 laps 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. This loop is threaded.

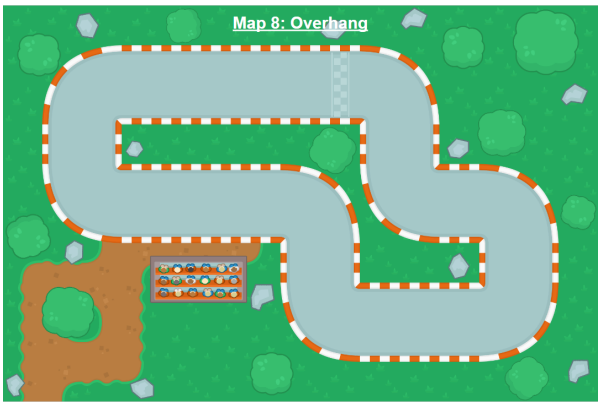
Maps

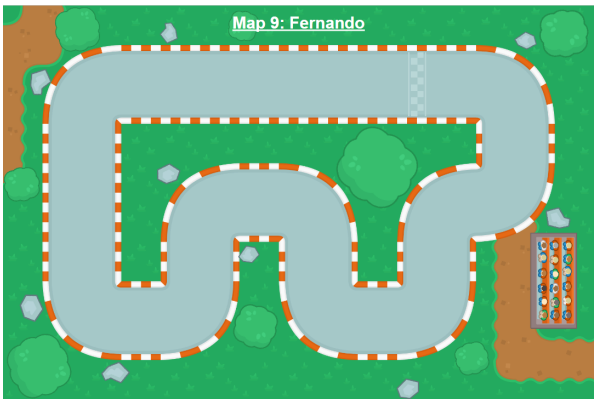



Map 5: Pinch


Map 6: T Bone

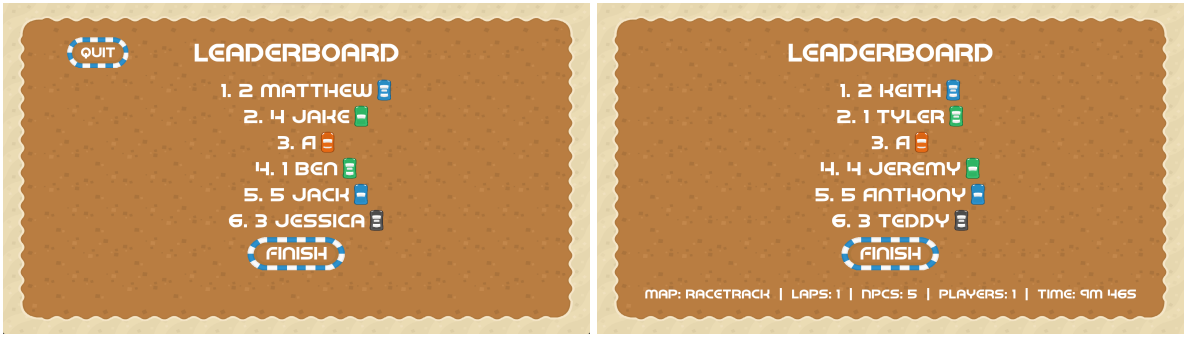

Map 7: Bridge


Map 8: Overhang


Map 9: Fernando


Map 10: Crumple

Leaderboard window



Code

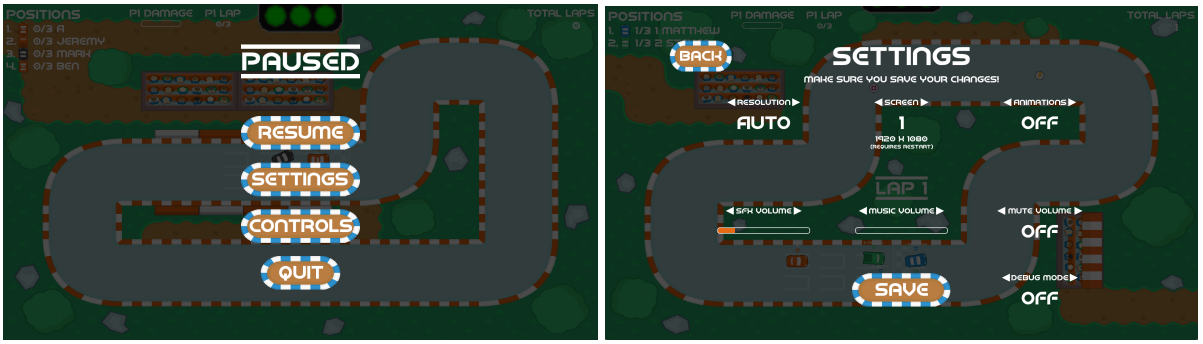
```
def leaderboard_window(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

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

    x = pad_x + 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 (only during gameplay)



Code

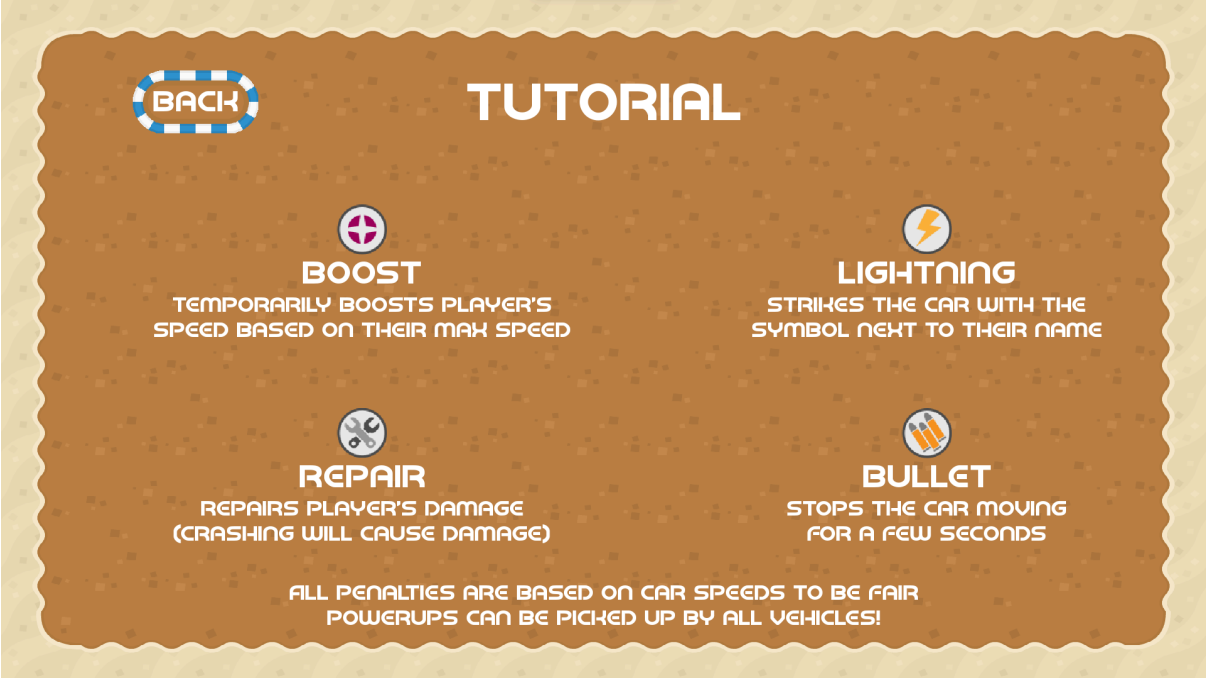
```
def paused_window():
    Secondary_window.fill(BLACK)
    Secondary_window.blit(Window_screenshot, (0, 0))

    draw_text(CENTRE[0], 115, 'Paused', WHITE, 100, bar=True, surf=Secondary_window)  # Title

    x = CENTRE[0] - 192
    y = CENTRE[1] - 180
    tile(x, y, 'dirt road', 76, grid=False, surf=Secondary_window)  # Resume button
    tile(x + 128, y, 'dirt road', 1, grid=False, surf=Secondary_window)
    tile(x + 256, y, 'dirt road', 60, grid=False, surf=Secondary_window)
    draw_text(x + 193, y + 20, 'Resume', WHITE, 70, surf=Secondary_window)

    x = CENTRE[0] - 192
    y = CENTRE[1] - 30
    tile(x, y, 'dirt road', 76, grid=False, surf=Secondary_window)  # Settings button
    tile(x + 128, y, 'dirt road', 1, grid=False, surf=Secondary_window)
    tile(x + 256, y, 'dirt road', 60, grid=False, surf=Secondary_window)
    draw_text(x + 190, y + 20, 'Settings', WHITE, 70, surf=Secondary_window)

    x = CENTRE[0] - 128
    y = CENTRE[1] + 120
    tile(x, y, 'dirt road', 76, grid=False, surf=Secondary_window)  # Quit button
    tile(x + 128, y, 'dirt road', 60, grid=False, surf=Secondary_window)
    draw_text(x + 130, y + 20, 'Quit', WHITE, 70, surf=Secondary_window)
```

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



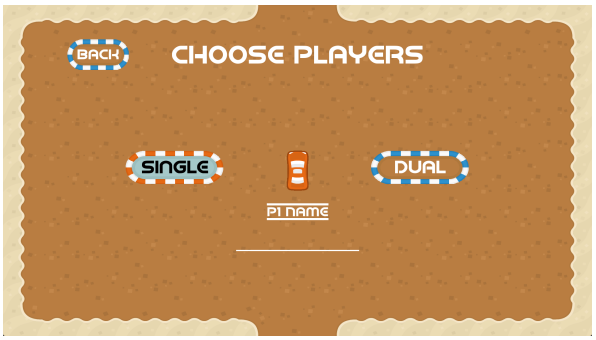
Description

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.

Code

```
def tutorial_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, 'Tutorial', 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 + 535
    y = pad_y + 325
    surf = pygame.transform.scale(pygame.image.load(assets.power_up('boost')), (80, 80))
    surf.set_colorkey(BLACK)
    Window.blit(surf, (x, y))
    draw_text(x + 40, y + 80, 'Boost', WHITE, 60)
    draw_text(x + 40, y + 140, "Temporarily boosts player's", WHITE, 40)
    draw_text(x + 40, y + 180, "speed based on their max speed", WHITE, 40)

    x = pad_x + 535
    y = pad_y + 649
    surf = pygame.transform.scale(pygame.image.load(assets.power_up('repair')), (80, 80))
    surf.set_colorkey(BLACK)
    Window.blit(surf, (x, y))
    draw_text(x + 40, y + 80, 'Repair', WHITE, 60)
    draw_text(x + 40, y + 140, "Repairs player's damage", WHITE, 40)
    draw_text(x + 40, y + 180, "(crashing will cause damage)", WHITE, 40)

    x = pad_x + 1435
    y = pad_y + 325
    surf = pygame.transform.scale(pygame.image.load(assets.power_up('lightning')), (80, 80))
    surf.set_colorkey(BLACK)
    Window.blit(surf, (x, y))
    draw_text(x + 40, y + 80, 'Lightning', WHITE, 60)
    draw_text(x + 40, y + 140, "Randomly causes an", WHITE, 40)
    draw_text(x + 40, y + 180, "NPC to crash", WHITE, 40)

    x = pad_x + 1435
    y = pad_y + 649
    surf = pygame.transform.scale(pygame.image.load(assets.power_up('bullet')), (80, 80))
    surf.set_colorkey(BLACK)
    Window.blit(surf, (x, y))
    draw_text(x + 40, y + 80, 'Bullet', WHITE, 60)
    draw_text(x + 40, y + 140, "Penalises the player", WHITE, 40)
    draw_text(x + 40, y + 180, "by a few seconds", WHITE, 40)

    draw_text(CENTRE[0], 923, 'All penalties are based on car speeds to be fair', WHITE, 40)
    draw_text(CENTRE[0], 963, 'Powerups can only be picked up by players!', WHITE, 40)
```

Choose players 2 window



Code

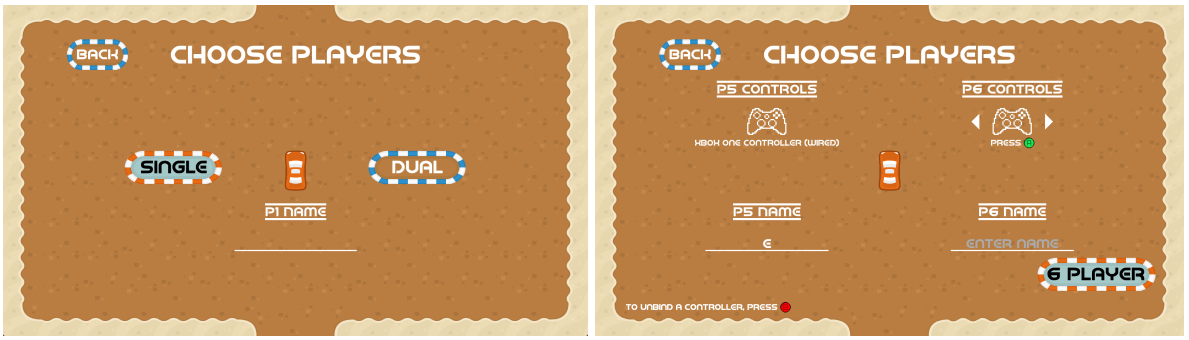
```
def choose_players_window_2(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    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 == 3:
        player = Players[2]
        # Controls
        x = pad_x + CENTRE[0]
        y = pad_y + 240
        draw_text(x, y, 'P3 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 840, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 1080, pad_y + 380), 'right', width=25, height=50)

        x = pad_x + CENTRE[0]
        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P3 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, player.name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 3:
            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 != 3 and not player.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)

    elif Player_amount >= 4:
        # P1
        player = Players[2]
        # Controls
        x = pad_x + 560
        y = pad_y + 240
        draw_text(x, y, 'P3 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 440, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 680, pad_y + 380), 'right', width=25, height=50)

        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P3 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, player.name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 3:
            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 != 3 and not player.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)

        # P2
        player = Players[3]
        x = pad_x + 1360
        y = pad_y + 240
        # Controls
        draw_text(x, y, 'P4 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 1240, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 1480, pad_y + 380), 'right', width=25, height=50)

        y = pad_y + CENTRE[1] + 100
        # P2 name title
        draw_text(x, y, 'P4 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, player.name, WHITE, 50, return_rect=True)
        # P2 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 4:
            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 != 4 and not player.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)

    for player in Players:
        if player.controls in controllers:
            rect = draw_text(100, HEIGHT - 110, 'To unbind a controller, press ',
                             WHITE, 32, center_x=False, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('b')), (34, 34)),
                        (100 + rect.width, HEIGHT - 111))

    # START BUTTON
    if Player_amount == 3 and Players[0].name.strip() and Players[0].controls != 'controller' and \
            Players[1].name.strip() and Players[1].controls != 'controller' and Players[2].name.strip() and \
            Players[2].name.strip() or Player_amount >= 4 and Players[0].name.strip() and \
            Players[0].controls != 'controller' and Players[1].name.strip() and Players[1].controls != 'controller' \
            and Players[2].name.strip() and Players[2].name.strip() and Players[3].name.strip() and \
            Players[3].controls != 'controller':
        x = pad_x + 800
        y = pad_y + 940
        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 == 3:
        x = pad_x + 93
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 3 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '3 Player', BLACK, 70)
    elif Player_amount == 4:
        x = pad_x + 492
        y = pad_y + HEIGHT - 255
```

```
    tile(x, y, 'road', 77, grid=False)  # 4 player button
    tile(x + 128, y, 'road', 2, grid=False)
    tile(x + 256, y, 'road', 61, grid=False)
    draw_text(x + 190, y + 20, '4 Player', BLACK, 70)
elif Player_amount == 5:
    x = pad_x + WIDTH - 876
    y = pad_y + HEIGHT - 255
    tile(x, y, 'road', 77, grid=False)  # 5 player button
    tile(x + 128, y, 'road', 2, grid=False)
    tile(x + 256, y, 'road', 61, grid=False)
    draw_text(x + 190, y + 20, '5 Player', BLACK, 70)
elif Player_amount == 6:
    x = pad_x + WIDTH - 477
    y = pad_y + HEIGHT - 255
    tile(x, y, 'road', 77, grid=False)  # 6 player button
    tile(x + 128, y, 'road', 2, grid=False)
    tile(x + 256, y, 'road', 61, grid=False)
    draw_text(x + 190, y + 20, '6 Player', BLACK, 70)
```

Description

This window is exactly the same as the choose players window, except it doesn't have the options to choose the amount of players at the bottom since to get to this window the user would have had to choose the amount of players anyways. This window allows players 3 or 3 and 4 to choose their name and continue to the next window.

Choose players 3 window



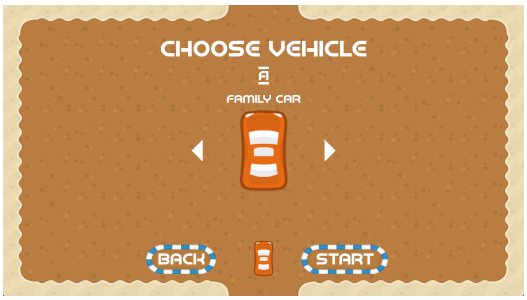
Code

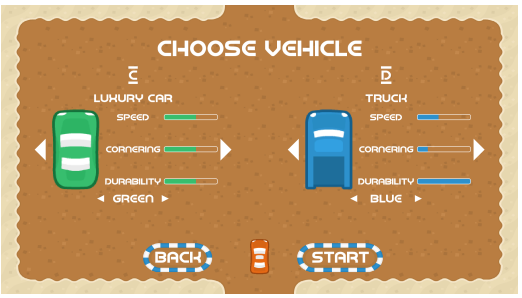
```
def choose_players_window_3(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    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 == 5:
        player = Players[4]
        # Controls
        x = pad_x + CENTRE[0]
        y = pad_y + 240
        draw_text(x, y, 'P5 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 840, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 1080, pad_y + 380), 'right', width=25, height=50)

        x = pad_x + CENTRE[0]
        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P5 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, player.name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 5:
            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 != 5 and not player.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 + WIDTH - 876
        y = pad_y + HEIGHT - 255
        tile(x, y, 'road', 77, grid=False)  # 5 player button
        tile(x + 128, y, 'road', 2, grid=False)
        tile(x + 256, y, 'road', 61, grid=False)
        draw_text(x + 190, y + 20, '5 Player', BLACK, 70)

    if PLayer_amount == 6:
        # P1
        player = Players[4]
        # Controls
        x = pad_x + 560
        y = pad_y + 240
        draw_text(x, y, 'P5 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 440, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 680, pad_y + 380), 'right', width=25, height=50)

        y = pad_y + CENTRE[1] + 100
        # P1 name title
        draw_text(x, y, 'P5 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, player.name, WHITE, 50, return_rect=True)
        # P1 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 5:
            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 != 5 and not player.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)

        # P2
        player = Players[5]
        x = pad_x + 1360
        y = pad_y + 240
        # Controls
        draw_text(x, y, 'P6 controls', WHITE, 50, bar=True)
        rect = draw_controls(x, y + 140, player.controls, return_rect=True)
        if player.controls == 'controller':
            rect_2 = draw_text(x - 16, y + rect.height + rect.centerx + 42, 'Press ', WHITE, 32, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('a')), (34, 34)),
                        (x + rect_2.centerx - 16, y + rect.height + rect.centerx + 41))
        elif player.controls in controllers:
            draw_text(x, y + rect.height + rect.centerx + 42,
                      short_controller_name(player.controls.get_name()), WHITE, 32)
        if type(player.controls) == str:
            draw_triangle((pad_x + 1240, pad_y + 380), 'left', width=25, height=50)
            draw_triangle((pad_x + 1480, pad_y + 380), 'right', width=25, height=50)

        y = pad_y + CENTRE[1] + 100
        # P2 name title
        draw_text(x, y, 'P6 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, player.name, WHITE, 50, return_rect=True)
        # P2 name cursor
        if (pygame.time.get_ticks() // 1060) % 2 and selected_text_entry == 6:
            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 != 6 and not player.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)

    for player in Players:
        if player.controls in controllers:
            rect = draw_text(100, HEIGHT - 110, 'To unbind a controller, press ',
                             WHITE, 32, center_x=False, return_rect=True)
            Window.blit(pygame.transform.scale(pygame.image.load(assets.controller_button('b')), (34, 34)),
                        (100 + rect.width, HEIGHT - 111))

    # START BUTTON
    if Player_amount == 5 and Players[4].name.strip() and Players[4].controls != 'controller' or \
            Player_amount == 6 and Players[4].name.strip() and Players[4].controls != 'controller' and \
            Players[5].name.strip() and Players[5].controls != 'controller':
        x = pad_x + 800
        y = pad_y + 940
        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 + WIDTH - 477
    y = pad_y + HEIGHT - 255
    tile(x, y, 'road', 77, grid=False)  # 6 player button
    tile(x + 128, y, 'road', 2, grid=False)
    tile(x + 256, y, 'road', 61, grid=False)
    draw_text(x + 190, y + 20, '6 Player', BLACK, 70)
```

Description

This window is also the same as the choose players window 2 except that it lets players 5 or 5 and 6 choose their name before continuing to the next window.

Choose vehicle 2 window





Code

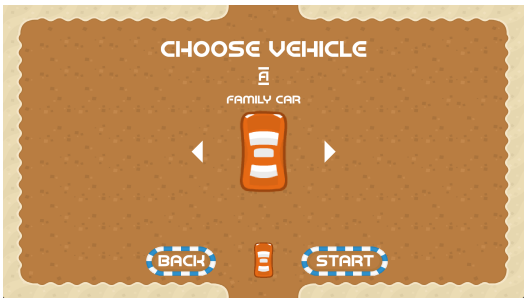
```
def choose_vehicle_window_2(curr_bg, pad_x=0, pad_y=0):
    Window.blit(curr_bg, (pad_x, pad_y))

    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 == 3:
        player = Players[2]
        draw_text(pad_x + CENTRE[0], pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + CENTRE[0], pad_y + 325, player.veh_name, WHITE, 50)

        p1_veh_rect = player.veh_image.get_rect()
        draw_triangle((pad_x + CENTRE[0] - 170 - p1_veh_rect.width, pad_y + CENTRE[1]), 'left', width=40, height=80)
        Window.blit(player.veh_image, (pad_x + CENTRE[0] - 215 - p1_veh_rect.width // 2,
                                       pad_y + CENTRE[1] - p1_veh_rect.height // 2))
        draw_triangle((pad_x + CENTRE[0] + 170 + p1_veh_rect.width, pad_y + CENTRE[1]), 'right', width=40, height=80)

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + CENTRE[0], pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

    elif Player_amount >= 4:
        # 1P VEHICLE
        player = Players[2]
        pos_x = CENTRE[0] // 2 + 10
        draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

        # 2P VEHICLE
```

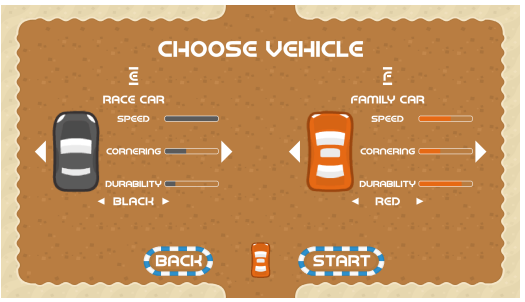
```
player = Players[3]
pos_x = CENTRE[0] + CENTRE[0] // 2 - 10
draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

if player.veh_colour == RED_CAR:
    text = 'Red'
elif player.veh_colour == YELLOW_CAR:
    text = 'Yellow'
elif player.veh_colour == GREEN_CAR:
    text = 'Green'
elif player.veh_colour == BLUE_CAR:
    text = 'Blue'
else:
    text = 'Black'

draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Sports Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Luxury Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Truck':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Race Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
```

Description

This window works the same way as the choose vehicle window, except instead of players 1 and 2 being able to choose their vehicle, this window allows players 3 and 4 to choose their vehicle(s) for the race.

Choose vehicle 3 window





Code

```
def choose_vehicle_window_3(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 == 5:
        player = Players[4]
        draw_text(pad_x + CENTRE[0], pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + CENTRE[0], pad_y + 325, player.veh_name, WHITE, 50)

        p1_veh_rect = player.veh_image.get_rect()
        draw_triangle((pad_x + CENTRE[0] - 170 - p1_veh_rect.width, pad_y + CENTRE[1]), 'left', width=40, height=80)
        Window.blit(player.veh_image, (pad_x + CENTRE[0] - 215 - p1_veh_rect.width // 2,
                                       pad_y + CENTRE[1] - p1_veh_rect.height // 2))
        draw_triangle((pad_x + CENTRE[0] + 170 + p1_veh_rect.width, pad_y + CENTRE[1]), 'right', width=40, height=80)

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + CENTRE[0], pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + CENTRE[0] + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

    elif Player_amount == 6:
        # 1P VEHICLE
        player = Players[4]
        pos_x = CENTRE[0] // 2 + 10
        draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
        draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

        if player.veh_colour == RED_CAR:
            text = 'Red'
        elif player.veh_colour == YELLOW_CAR:
            text = 'Yellow'
        elif player.veh_colour == GREEN_CAR:
            text = 'Green'
        elif player.veh_colour == BLUE_CAR:
            text = 'Blue'
        else:
            text = 'Black'
        draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Sports Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Luxury Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Truck':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
        elif player.veh_name == 'Race Car':
            draw_slider((pad_x + pos_x + 115, pad_y + 408),
                        (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 528),
                        (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
            draw_slider((pad_x + pos_x + 115, pad_y + 648),
                        (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)

        # 2P VEHICLE
```

```
player = Players[5]
pos_x = CENTRE[0] + CENTRE[0] // 2 - 10
draw_text(pad_x + pos_x, pad_y + 220, player.name, WHITE, 60, bar=True)
draw_text(pad_x + pos_x, pad_y + 325, player.veh_name, WHITE, 50)

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

if player.veh_colour == RED_CAR:
    text = 'Red'
elif player.veh_colour == YELLOW_CAR:
    text = 'Yellow'
elif player.veh_colour == GREEN_CAR:
    text = 'Green'
elif player.veh_colour == BLUE_CAR:
    text = 'Blue'
else:
    text = 'Black'

draw_text(pad_x + pos_x, pad_y + CENTRE[1] + 160, text, 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 player.veh_name == 'Family Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Sports Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 4, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Luxury Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 3, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Truck':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
elif player.veh_name == 'Race Car':
    draw_slider((pad_x + pos_x + 115, pad_y + 408),
                (200, 22), 5, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 528),
                (200, 22), 2, 0, 5, center_x=False, fill_color=player.veh_colour)
    draw_slider((pad_x + pos_x + 115, pad_y + 648),
                (200, 22), 1, 0, 5, center_x=False, fill_color=player.veh_colour)
```

Description

This window is also the same as the choose vehicle window except that it allows players 5 or players 5 and 6 to choose their vehicle and vehicle colour for the race then allowing them to move to the next window.

Alpha + Beta Testing

Key:

Pass: Feature works with no problems as stated
Minor fail: Feature works with no problems but implemented differently than stated
Fail: Feature was not implemented or does not work

Main

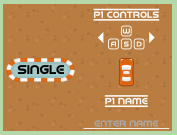
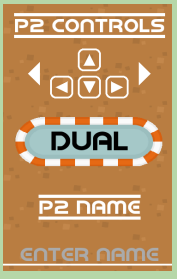
No.	Feature	Explanation	Result	Fail reason
1	Quit button	Check that the button navigates to confirm quit window and highlights on mouseover	Pass 	
2	Start button	To check that the button navigates to choose map window and updates properly	Pass 	
3	Settings button	Check that button navigates to settings window and updates properly	Pass 	
4	Credits button	To check that the button navigates to ‘credits window’ and updates properly	Pass 	
5	Window + car animations	Ensure the smooth transition of windows and proper rotation of car during animation	Pass 	
6	Mouse cursor	For all windows where the cursor is shown, exchange the usual cursor for a custom one	Fail	Not implemented

Choose Map

No.	Feature	Explanation	Result	Fail reason
7	Confirm button	Check that the button navigates to ‘choose players window’	Pass 	
8	Back button	Check that the button navigates to the ‘main window’	Pass 	

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

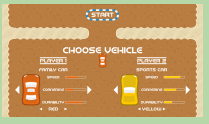
Choose players

No.	Feature	Explanation	Result	Fail reason
13	1 Player button	Check that the button links to the single player choose vehicle window	Pass 	
14	2 Player button	Check that the button links to the two player choose vehicle window	Pass 	
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)	Fail 	Players can enter numbers and symbols but still only one at a time

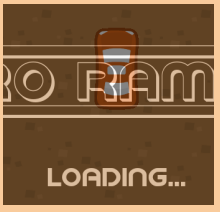
16	P2 Name	Only enter P2 name after P1 name and have pressed ENTER	Minor fail 	User doesn't have to press ENTER, they can also click away instead
17	Window + car animations	Make sure that the car rotates properly and the window animation is smooth	Pass 	

Choose Vehicle

No.	Feature	Explanation	Result	Fail reason
18	Car name	Make sure that the name always matches the current car	Pass 	
19	Car preview	Make sure that the proper car and colour are showing according to the player's settings	Pass 	
20	Speed, cornering and durability text	Ensure that text stays in proper position and is always on screen	Pass 	
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	Pass 	
22	Car colour text	Make sure that the colour text always matches the colour of the car	Pass 	

23	Back button	Ensure that the back button links to the main window and that it updates properly	Pass 	
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.	Minor fail	Not implemented as feature makes no sense since the player always has a valid vehicle option
25	Car left button	Make sure that the button properly cycles the car and all related info left once	Pass 	
26	Car right button	Make sure that the button properly cycles the car and all related info right once	Pass 	
27	Colour left button	Check that the car colour left button properly cycles the car's colour and text left once	Pass 	
28	Colour right button	Check that the car colour right button properly cycles the car's colour and text right once	Pass 	
29	Window + car animations	Make sure that the car properly animates during window animation and that all assets stay in the right position	Pass 	

Race Settings

No.	Feature	Explanation	Result	Fail reason
30	Setting titles	Make sure that the titles are all in the correct position and size	Pass 	
31	Setting values	Ensure that the values are correct and correspond with any bars	Pass 	
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	Pass 	
33	Selected map and vehicle(s)	Ensure that the text for the current map, season and vehicles is properly centred at the top of the screen	Fail	Not implemented
34	Start game button	Check that the start game button properly links to the correct map and updates correctly	Pass 	
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.	Minor fail 	Car does not drive off the screen, instead the window fades to a loading screen with threaded animation (if enabled)

Main gameplay

No.	Feature	Explanation	Result	Fail reason
36	Map background	Make sure that the map background has been properly generated in the correct position and season	Fail	Map seasons are not implemented
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	Pass 	
38	NPC vehicles and control	Ensure that the NPC works properly and manoeuvres the track(s) efficiently. Also check that the NPC colour setting matches their vehicles	Pass 	
39	Powerups	Make sure that power ups only appear on the track and that they work properly	Pass 	
40	GUI	Ensure that the GUI is properly updated according to what is going on in the game, and includes visuals for player position and damage	Pass 	
41	Start positions	Make sure that all of the NPC and players correctly start in the positions, and the players start at the front if easy and work backwards depending on difficulty	Minor fail 	Difficulty not implemented but players can choose starting position
42	Map crossovers	Ensure that the cars can only go straight and cannot cut off a part of the track	Fail	Maps with crossovers not implemented

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	Pass 	
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.	Minor fail 	A repair powerup repairs all of the car's damage rather than incremental
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	Pass (Can't screenshot hidden mouse)	

Leaderboard

No.	Feature	Explanation	Result	Fail reason
46	Quit button	Make sure that the quit button links back to the confirm quit screen, and that it updates properly	Minor fail	Feature was implemented then removed as irrelevant
47	Settings button	Check that the settings button links to the settings window and that it updates according to the mouse position	Minor fail	Feature was implemented then removed as irrelevant
48	Main menu button	Make sure that the main menu button properly updates and that the button links to the main window	Pass 	
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	Pass 	

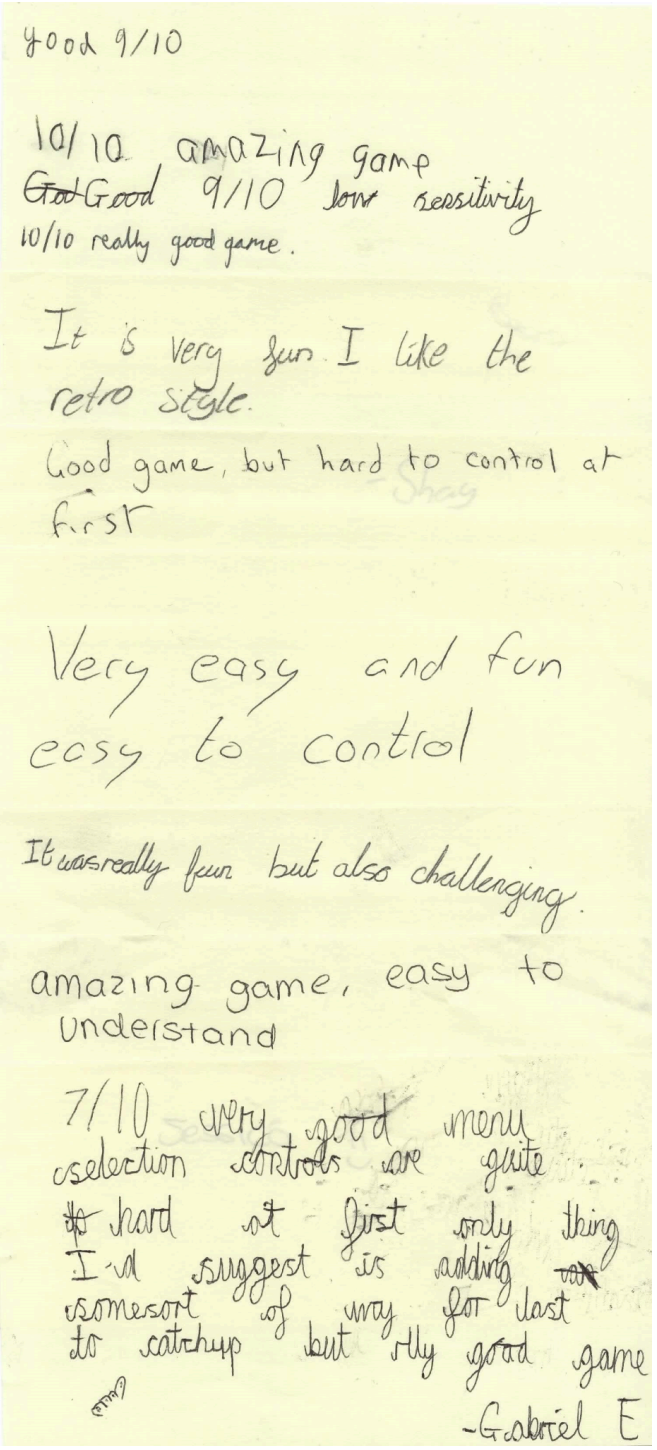
50	Map settings	Also include a small section to the right of the leaderboard that shows the map settings that were used for that race	Pass 	Race details are across the bottom of the screen
51	Window animations	There is no window car at this point, so just have the screen either change to the next window or have no animation	Pass 	Window transitions by fading to and from black

User Feedback

As a part of testing this product, the game was showcased at a local secondary school's open evening so that people of various ages that attended could try out the game and leave some feedback / review the game at its current state. Over 40 people played the game throughout the night and more people observed others playing the game. 33 of the people who played the game left some kind of feedback on the game. The game was set up on a bench with 2 monitors so more people could see the game and the music / audio was also on high to attract more people to come and try out the product. For accessibility and testing reasons, 4 controllers were available for people to play with their preferences against up to 4 other people or NPCs. A few examples of some reviews are below:

All of the reviews are in the following table (Names and pictures excluded):

User reviews
"good 9/10"
"10/10 amazing game"
"good 9/10 low sensitivity"
"10/10 really good game"
"It is very fun[,] I like the retro style."
"Good game, but hard to control at first"
"Very easy and fun[.] easy to control"
"It was really fun but also challenging."
"amazing game, easy to understand"
"7/10 very good menu selection[.] controls are quite hard at first[,] only thing I'd suggest is adding some sort of way for last [place]to catch up but r[ea]lly good game"
"The game was very fun but difficult"
"10/10 game"
"very good"



“It is amazing”
“Good Game”
“It was fun”
“Very good”
“it was good”
“good game great creation”
“very good and entertaining and has music”
“very good game well designed”
“The racetrack is challenging”
“Amazing, very talented!”
“Pretty good[,] slow a little [but]nothing bad[,] really decent game.”
“Very good.”
“fantastic game[,] graphics [are]fantastic”
“Very good game easy to play”
“good”
“good”
“very good game”
“Very good”
“Good 10/10”
“It[‘]s a very good game and it must …[have taken] a while to make.”

Evaluation

Success Criteria

Key:

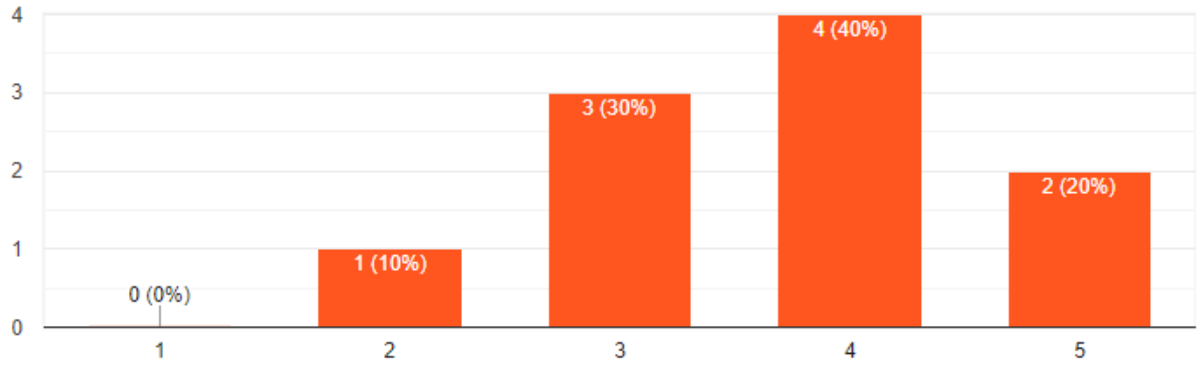
Pass: Feature works with no problems as stated
Minor fail: Feature works with no problems but implemented differently than stated
Fail: Feature was not implemented or does not work

Number	Criteria	How to implement	Alternative	Outcome
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	Pass
2	Confirm quit screen	Yes or no button in centre	Quits without asking or waits before quitting	Pass
3	Travel between screens / windows	Screens stitch together and move across screen	Windows change with no animation	Pass
4	Users can change settings	Screen from main menu with options	Button in pause menu or corner to page	Pass
5	Users can view credits and licences	From main menu through button	Shows licences on launch of game	Minor fail - Credits on credits screen
6	Choose amount of players	Window from main menu	Directly from main menu	Pass
7	Select specific map or choose random	New window after 1p or 2p	On same window as AI amount and lap amount	Minor fail - Random map not implemented
8	Choose vehicle and colour	After map and amount of players has been chosen, one by one player	Split-screen player control independent at same time	Pass
9	Amount of NPCs	Window after vehicle chosen	Before on main menu or after map selection	Pass
10	Confirm settings	Separate window before starting race	On same window as the race starts	Pass
11	Countdown to race start	Text in middle of screen	Text centred at top of each screen	Pass

12	Names above cars	Centred above player's car	Always vertical above car	Pass
13	Position of each player	Top left / right of screen	Both centred in middle top of screen	Pass
14	Amount of laps and progress	Centred at top of screen	Centred at bottom or above minimap if implemented	Pass
15	Ability to pause game	By pressing esc key	By pressing Enter or other button	Pass
16	Change settings from pause menu	Button to go to same window as from main screen	Custom window with limited settings	Pass
17	Separate control of audio and sounds	In settings menu	Upon first time launch	Pass
18	Option to randomise race settings	On race settings window	On any window before the race	Fail - Low priority implementation
19	Leaderboard	After race has ended	After championship has ended	Pass
20	Ability to chain races into single championship	By selecting multiple maps at once on map window	Separate windows for each map chosen	Fail - Low priority implementation
21	Performance settings for low-end computers	In settings window	Together with other settings	Pass
22	Change NPC difficulty	When choosing NPC options	When choosing the map	Minor fail - Implemented force vehicle instead
23	Toggle menu scrolling animation	Through an option in settings	Together with other settings	Pass
24	Change the window resolution	On settings window	Together with other settings	Pass
25	Have an auto option for resolution	In settings window	Together with other settings	Pass
26	Be able to specify a display	In settings window	Together with other settings	Pass

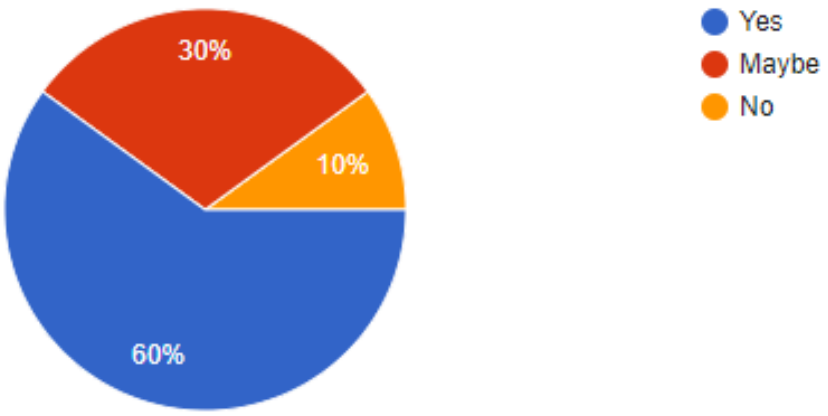
Usability Questionnaire

Question 1 - How easy did you find the menu navigation?



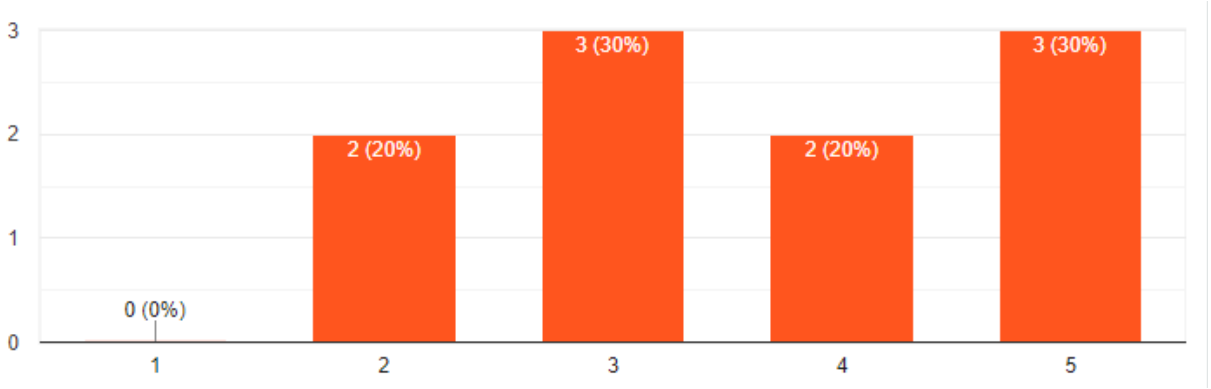
The responses to this question were overall good and showed that most people found the menus easy to navigate and use, meaning they should be accessible to almost all people.

Question 2 - Do you think the menu navigation will be intuitive for most people?



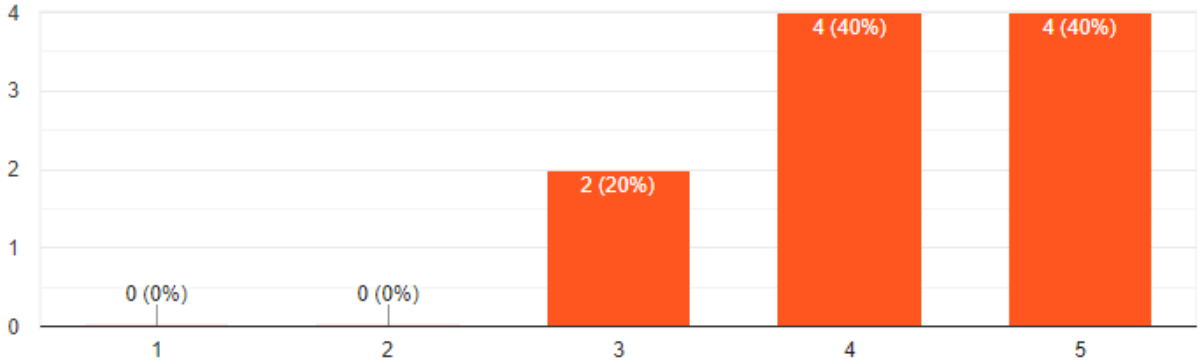
Only 10% of people thought that the menu navigation wouldn't be intuitive for most people, leaving 90% of people thinking that the menus would be. This is an overall positive response as even if 10% of people don't find the menus easy to use, there is a good chance that they will figure it out after a bit of time.

Question 3 - How easy was it for you to start a race?



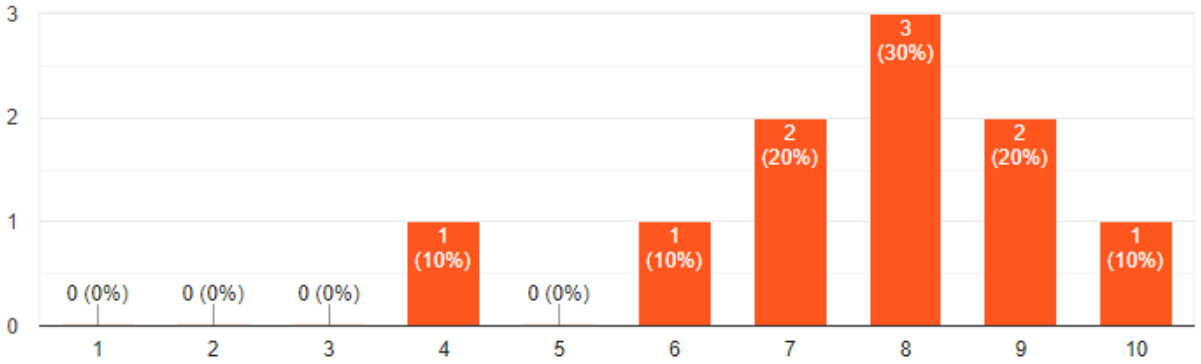
The responses from this question were a bit lower than others, however this is probably the result of there being a large amount of options such as cars, NPCs and map options that users found it harder to start a race. Unfortunately this is inevitable yet most people still put 3 or above which is good.

Question 4 - How helpful / useful did you find the accessibility features and settings?



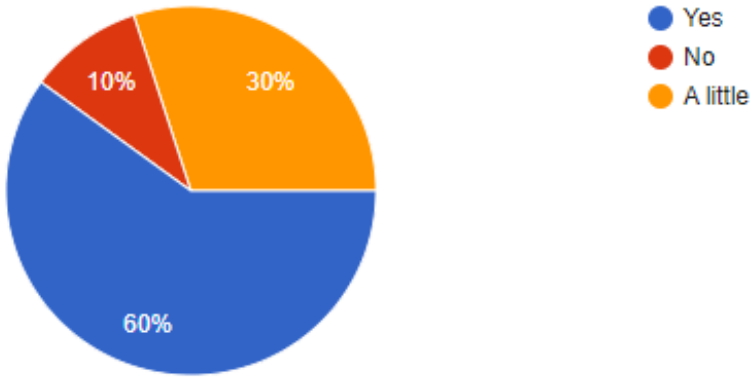
People responded very highly to this question with not a single person choosing below 3. This clearly shows that the chosen features and settings are the ones that people will find the most useful when using our product.

Question 5 - Did you like the theme and style of the product?



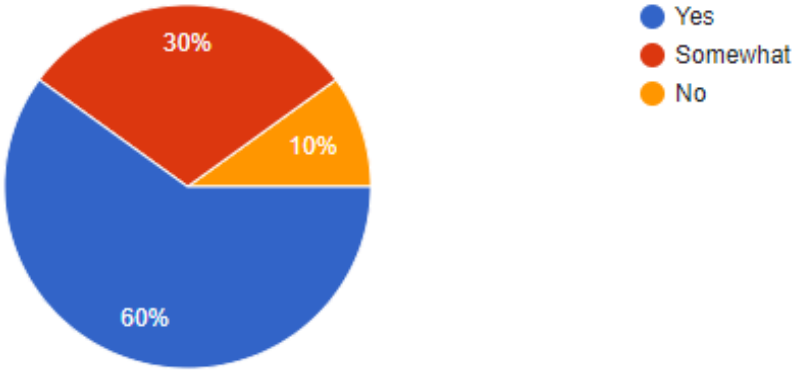
The responses from this question show that 90% of people liked the theme and style of the product. This is important as it is crucial that a user likes the style of the product, otherwise they may not like other things such as gameplay only because they didn't like the theme.

Question 6 - Do you think the artwork, functionality and audio all follow the same theme?



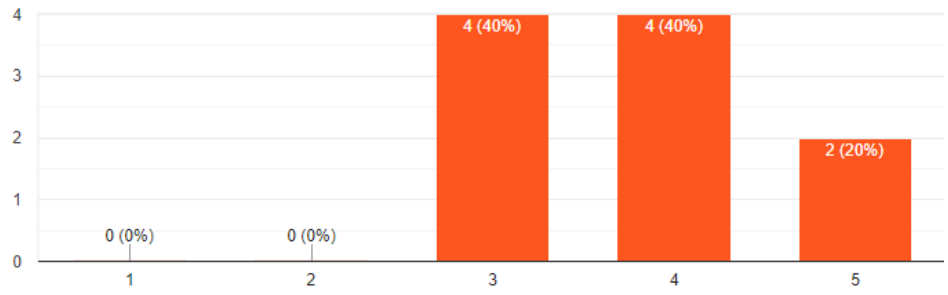
90% of people who completed the questionnaire thought that the theme and style of the product was consistent or partially consistent throughout the product. This is good as many people liked the theme and style is important, it is arguably more important that the theme and style is consistent. For example, Rock music wouldn't go with a rainy day much like extremely good graphics wouldn't work for this game as it doesn't fit the style.

Question 7 - Were the settings relevant to what you were looking for?



More than half of the respondents found the settings adequate to what they were looking to change. This is good as it means the settings cover a large area of useful and specific settings that users will use the most over others.

Question 8 - Did you like the menu animations and transitions?



None of the respondents disliked the menu animations and transitions, this is likely because even if the user didn't like the animation sequence they are able to turn it off in the settings.

Future Developments

Based on the responses of the current state of the product, overall most people found the game very accessible and general responses were good. However, if we were to improve the product in the future based on the feedback, some of these improvements could be:

- Even though overall people found the menus easy to navigate, some still struggled, therefore an alternative to navigating the menus such as being able to use a controller would be an easy and effective solution to improve the accessibility of the menus.

- The only other aspect of the game that users found difficult was starting a race. Even though this is inevitable, it could still be mitigated in the future with things like a random option for a race that would allow the user to start a race with a single press. Another alternative could be a preset system that users could set up once and be saved for future races, although this would be harder to implement and therefore cost more to develop than the first solution.

- Some of the settings initially described in the design stage were not implemented due to other settings taking priority, meaning these additional settings such as ‘V-sync’ could still prove useful to some people if it were to be implemented in future updates.

Additional Features

Much like the future development plans, there are also some additional features that could be added to the product to improve it overall. Some of these are:

- A larger amount of animations and transitions such as lightning effects while ‘zapped’ to show reason for stopped, and NPCs appearing transparent and having no collision for a few seconds after resetting to the last checkpoint would improve the amount of feedback to the user, and improve the ‘flow’ of the game.
- There are some limitations with the method that was used to move the vehicles that does not allow for any kind of velocity implementation and is bound to the refresh rate of the game. By changing the way the cars move it will allow for a higher frame rate and the implementation of velocity for each car making the game more realistic. However, this would also substantially increase the difficulty of the game in the process.

Because of the simplicity of the game, some users may even create modifications for the game such as different menu themes or car colours however more may add their own content to the game such as their own custom maps or vehicles since the code has been developed to be as simple as possible while making it easy to change features and options even without knowing too much about the game overall. This will help in the popularity of the game by bringing more communities to play and use our product and will therefore not be frowned upon or discouraged since the more people that play the game and have interest in it then the more money is made from developing it. It may also give more insights to what

users would modify the game to implement, providing second-hand feedback for implementation ideas and updates in the future to the base game.

Maintenance

In order to maintain the product in the future, several features have been developed in mind to add more to the product later on. One of these features is making more maps for players to race on which is very easy to develop. Other features such as adding more vehicle options for players and NPCs will be easy to implement because of the way that it has already been developed. For example, the entire GUI for selecting a map goes off a single list variable that can easily be added to, then the code does all of the preview and buttons itself. Much like the maps, vehicles work in the same kind of way with only having to change a few lines of code and the rest handles the extra options. Furthermore, other features that were considered during development such as an online leaderboard and even online multiplayer could be implemented to maintain interest in the product.

In terms of maintaining the product on bugs and minor fixes, there should already be very little bugs in the code because of the automated tests, however some of the game's profits should be invested in a developer to fix any bugs that users may find when using our product.

If additional online features such as a global leaderboard or multiplayer were to be implemented into the game then it would also come with the cost of hosting and maintaining a server to handle the online operations. This can be done by a third-party however it will still have to be maintained by costs from the profits of the game. However, at the same time this will also increase the popularity of the game bringing more profit from the game.