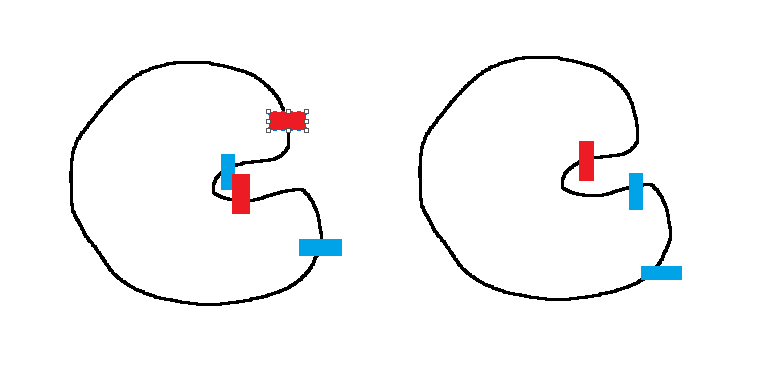
|  |  |
| --- | --- |
| Controls - 291 | |
| Initial Design | The initial design of the gameplay would consist of the player driving around the track as fast as possible. The player would fully control the car with on-screen buttons: Accelerate, Brake, Reverse and Turning. While turning and braking, the player could drift the car slightly to allow them to take sharper corners at higher speeds. Mastering of the drift mechanic will allow players to achieve faster lap times. |
| Implementation | Due to the mobile nature of the game, most of the game can only be operated with 2 inputs (thumbs). With playtesting, I found that players struggled to control all the movements needed, as they would have to take their fingers off the accelerator to turn. To remedy this, I made the car auto-accelerate as well as combine the brake and reverse function, so the brake button caused negative acceleration, rather than reducing the speed to 0.  Originally, drifting would start as soon as the car was braking and turning. This proved to be troublesome when there were slight turns in a track. To remedy this, I created a check on the sideways velocity of the car. Once it was above a margin, the drifting system would start. |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well | The simplified car mechanics allowed play-testers to manoeuvre most tracks with ease, I did receive some feedback that said allowing the player to have full control of the car could have some benefits on more complex tracks, but I found in most cases the car’s mechanics were suitable.  The drifting system was not perfect as it required the player to brake to use it. The brake button returned a Boolean (0/1), while I would have liked the drifting to be more intricate allowing the player to change the amount of braking (0.1 ->0.5 -> 1.0). |

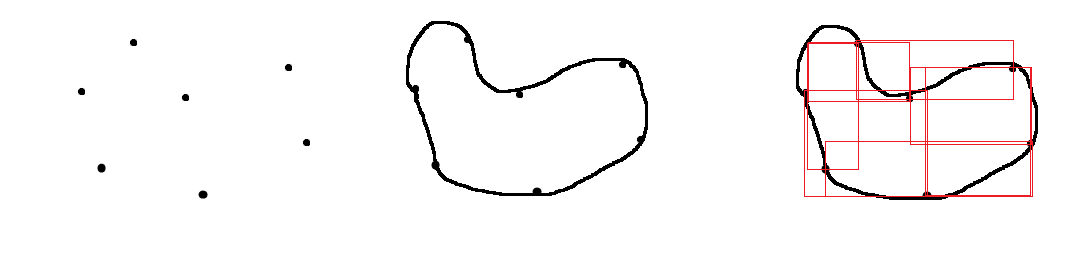
|  |  |
| --- | --- |
| Checkpoints - 391 | |
| Initial Design | To ensure the player is going around the track the right way, I wanted to add a checkpoint system to each track. These would have functions attached to them which were used by the race manager to check where the player was and to allow the player to return to them if they drive too far off the track. |
| Implementation | Each checkpoint is spawned with a positional ID, finish line bool and a collision trigger bounding box. When the player passes through the trigger, the checkpoint system checks to see if the ID of this checkpoint is next after the last passed through checkpoint. If it is not, then the player is directed towards the checkpoint they need to pass to progress. If the checkpoint passed though is the finish line, then a new lap is started and the last passed checkpoint is reset.  Due to the nature of the track’s generation, the checkpoints would have to be part of the creation process. Initially, I had the checkpoints made where each of the points of the track were. However, I found that the checkpoints could get too close together and that would interfere with the collision check required for the checkpoint system. I instead created a system which would create points along the path of the track that were equidistant.  To allow for different styles of play, the bounding box for the checkpoints would have to span the width of the track, I needed to spawn the checkpoints, perpendicular to the track. However, I could not find a mathematical way that produced the results I needed so instead I made the checkpoints look at their second control point on the curve. |
| Benefits / Pitfalls | I made the checkpoint system before I added segments to the track design, but if I were to redesign the system, I would move the checks to the segments as that would remove the need for the positioning and rotation processes. |
| Worked Well / Didn’t Work Well | Initially I wished the checkpoints to be invisible, but with testing, I found that players found it difficult to know where they had to pass. Subsequently, I made all the checkpoints visible.  If I could have made a way for the points to be perpendicular, I would have implemented it, the Look At method did not always produce the best results, but they were always within my margin of error. |
| Figure | Checkpoint Visual  Checkpoint System. Before/After  Pseudo code of checkpoint system. |





Path Generator

|  |  |
| --- | --- |
| Track Storage | |
| Initial Design | Tracks are stored as a sequence of 2D points, the path generator uses these points to create a closed Bezier curve. By performing tests on the points and path, most of the aspects of the track can be found such as Length, Height, Width, and the number of straights and corners. |
| Implementation | Originally, I had planned to stop here when it came to represent the track, but I started to develop the evolutionary algorithm I found that I needed a more in-direct way for it to view track as. In my research, I found a paper that evolved tracks by cutting the track into segments and then modifying the segments during the evolution process. I decided to use this methodology for my tracks, so I then added a process that creates segments from a track.  Each segment is comprised of 3 points, the start, middle and end points. Using vector maths, the distances and angles within the segment are calculated. With these measurements, it can be determined what direction the segment turns as well as its size and area. This is done for each point in the track to create an array of segments, which can be used in other processes. |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |
| Figures | Point Track  Path Track  Segment Track |



|  |  |
| --- | --- |
| Random Track Generation | |
| Initial Design | Random tracks were required so that I could test the algorithm tracks vs a random track to test if there was a noticeable difference. My requirements for a random generated track were:  Diverse – If every random track looks the same, it allows the player to make the distinction.  Closed Circuit – Allows for multiple laps  Completable – The player must be able to finish the track |
| Implementation | When making the random track generator, I needed to decide what type of random track to generate. Togelius (2007) used three different ways to make a random track: Straightforward, Random Walk and Radial. As they found in the paper, Random walk produced tracks that could be hard to drive on naturally and Radial method tracks made tracks that looked very similar (flowerlike).  I originally tested the straightforward methodology but found that tracks were still to similar. To try to fix this I decided to use elements of the methodology for the initial generation by making the original shape with a convex hull instead of a rectangle.  The convex hull was generated by creating a collection of randomly placed points and calculating the smallest polygon that encloses all the points. Instead of modifying the existing points, I decided to add points in-between the current points. These new points were randomly placed within the bounds of their parent points. |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |
| Figure | Straightforward / Radial / Random Walk  Straightforward vs Convex Hull  Midpoints |

|  |  |
| --- | --- |
| Track Testing | |
| Initial Design |  |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Player Tracking | |
| Initial Design | Each player can design their own profile with their own username and icon, this is not linked to the internal player profile used for tracking data. That profile is only created once they have read and passed the consent pages. For example, if someone below the age of 18 enters their age they will be able to access the game, but an account will not be made for them. I decided to let users under the participation age to play the base game with no data-tracking, so no player-based tracks, so that they are not encouraged to lie about their age in order to play the game. |
| Implementation | -Pre Race  -During the Race |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Ratings | |
| Initial Design | I decided to use a rating system after the race has finished instead of creating |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Track Evolution | |
| Initial Design | In my research, I found that some papers performed the mutations on a single track. |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Population Control | |
| Initial Design |  |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Direct Testing | |
| Initial Design |  |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

|  |  |
| --- | --- |
| Analytics | |
| Initial Design |  |
| Implementation |  |
| Benefits / Pitfalls |  |
| Worked Well / Didn’t Work Well |  |

Future

* Custom Cars -Based on Playstyle