GRIFFITH COLLEGE

Formula X

Documentation

Ciarán Sweeney

Student number:2838501

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Bachelor of Science in Computing at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed: _____ **Date: _____**

Contents

Acknowledgements

I would like to thank Griffith College for having me as a student for the last four years. I would also like to thank my supervisor Ralph Croly who has been a great help throughout the development of my project. I would like to thank all the all lecturers for their support throughout the course especially Tony Mullins, Dr. Faheem Bukhatwa and Paddy Fahey I wish to acknowledge the support of my fourth year peers of this computer science course. Special thanks to the Js201 group Dan Boyle, Vincent Campbell and Aziza Kireyeva. This group have helped me enormously throughout the course. Finally I wish to thank my parents Bríd and John Sweeney for their encouragement and in particular their financial support throughout the four years.

Abstract

For this final year project I wanted to produce a somewhat unique game. I decided to produce a futuristic racing game which was very challenging. I drew my inspirations from F-zero a great game that is no longer in production in this modern era of gaming.

Introduction

Formula X

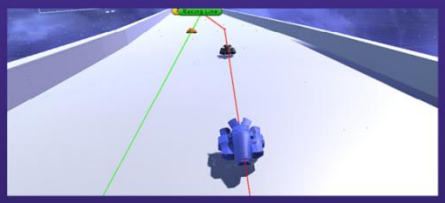
Ciarán Sweeney 2838501

Technologies Used

- Unity
- Blender
- FMOD Studio
- Visual Studio

Audiance

- The game is targeted towards racing gamers that enjoy the challenges of a racing game. This game may not be violent but it is difficult, so the ideal minimum age for this game would be about 7 years old.



Problems

- Hovering
- Acceleration
- Crashing
- Opponents AI

Solutions

- I solved the hovering issue by using ray-casts under the spacecraft. Once the the ray-cast hits the ground then upward force was applied to the spacecraft which caused a hovering affect.
- I solved the acceleration of my space craft by slowly and steadily adding forward force to the spacecraft. I also have a max speed to stop the acceleration once it gets to max point.
- To stop the excessive spinning from crashing, I increased the angle drag on my race craft.
- I Used way-points for the Opponents AI. These way-points acted as a racing line. This racing line can be seen above.

Supervisor: Ralph Croly

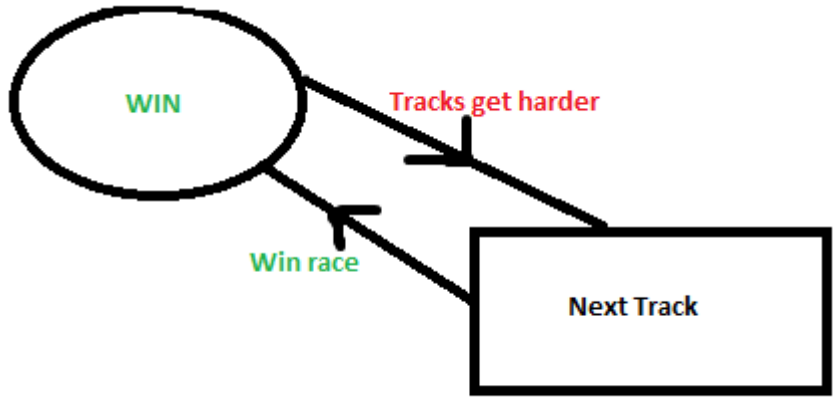
By Ciarán Sweeney

For my final year project I have decided to make a futurist racing game. In this game document I will be going into great detail of the development and the general internal workings of Formula X. As I said before that this game was influenced by the F-Zero series. I wanted to make Formula X this way because there seemed to be a hole in the market for a futuristic racing game in this modern era of gaming. The main goals of Formula X was to make a fast pace game that was very challenging for the hard core racer fan base. Another goal was to steer away from the party racing games such as Mario Kart which rewards luck rather than pure skill. At the same time Formula X was designed to be fun and I didn't want to go down the other end of the spectrum and end up with a game like Forza. The racing gaming fans are my target audience for this game and general gamers that enjoys a challenge and put their skills to the ultimate test. Basically Formula X goal is to be the Dark Soul of the racing game world. In this document I will go into four main topics which are user interface, feature set, testing and future work but I will first now explain the game's overview.

Game Overview

In this Section of the game document I will give a generally overview of the main ideas and philosophy be hide Formula X. I will also discuss the win condition needed for this game and the game play.

Game's philosophy



The main core plan of Formula X was to design a game that was enjoyable challenging for the racing game enthusiast. From a personal

point of view I find a game that is difficult to complete to be much more satisfying game experience. I have brought this ideology of challenging games to Formula X. One of the more notable aspects of this is in the difficulty of mastering the car controls in this game. Level two of Formula X is a prime example of my ideology being put into place. Level two is very challenging but I feel the reward for the player is greater for it. As you can see in the diagram above Formula X has a game cycle where the tracks increase in difficulty for every level the player wins. The difficulty of just the track will be harder on the next level. If there was more time given to work on this development project then this game cycle would have been fleshed out more. Making Formula X very fast paced game was one of the earliest idea's set in place. The reason for wanting a fast paced game was that it would give the user a great sense of excitement going that fast and being a futuristic game it would feel right that these cars were going at incredible speeds.

Win Condition

The win condition for Formula X is pretty straight forward, to win and process to the next level the player needs to finish first in the race. The lose condition is failure to finish first in the race. The win and loss condition for Formula X is very much black and white.

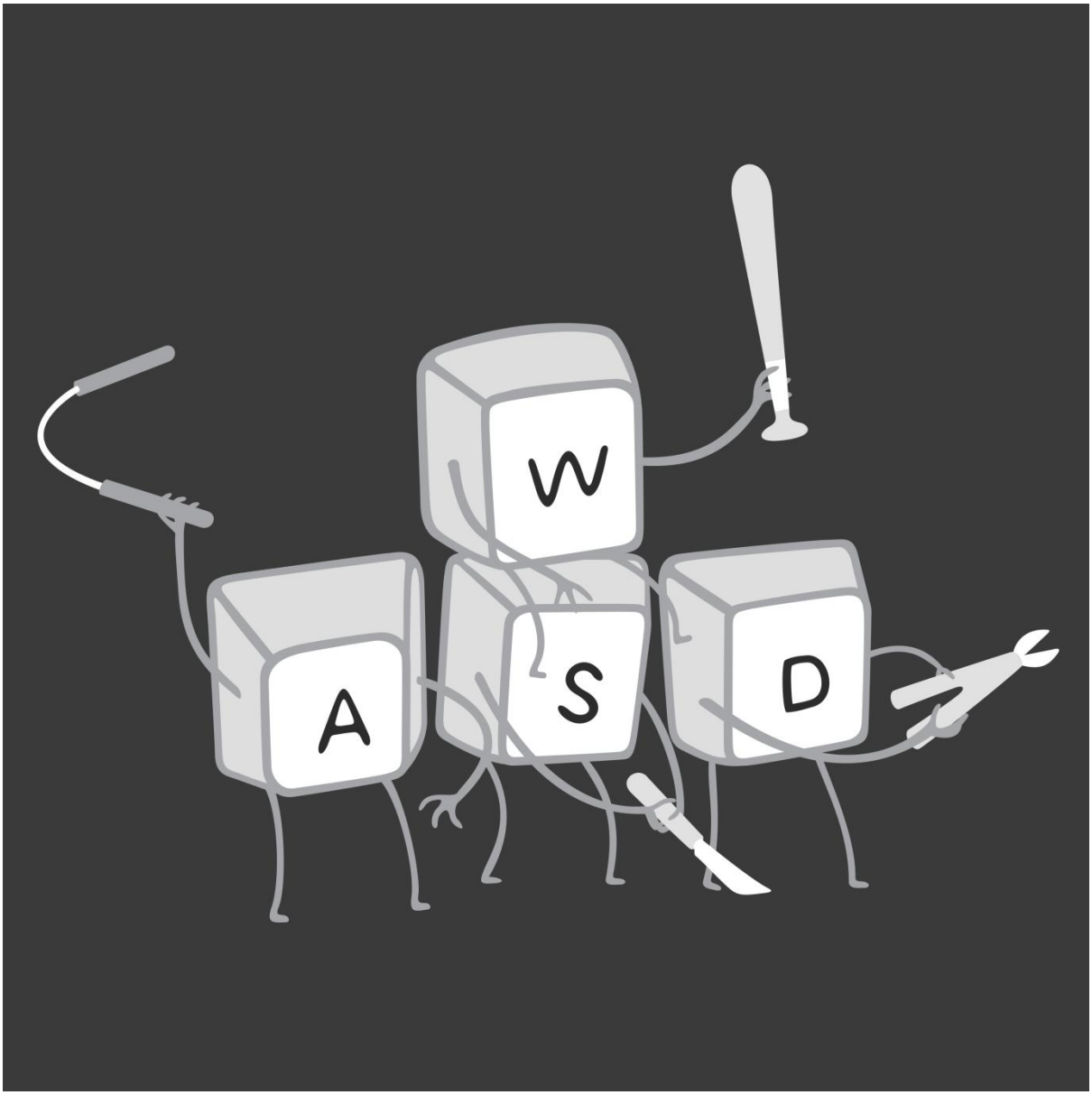
Game Play

The game play of Formula X works the same way as most racing games. The player must try and beat his opponent drivers by finishing the race in first place. Again like most racing games Formula X whole core game mechinchine revolves around the control of the player's car. There is also a limited supply of speed boost available to the player if the player uses the entire speed boast the player has to wait until he or she starts a new lap where by the speed boost will be full again. Formula X also keeps track of the player's current lap and best lap.

User Interface

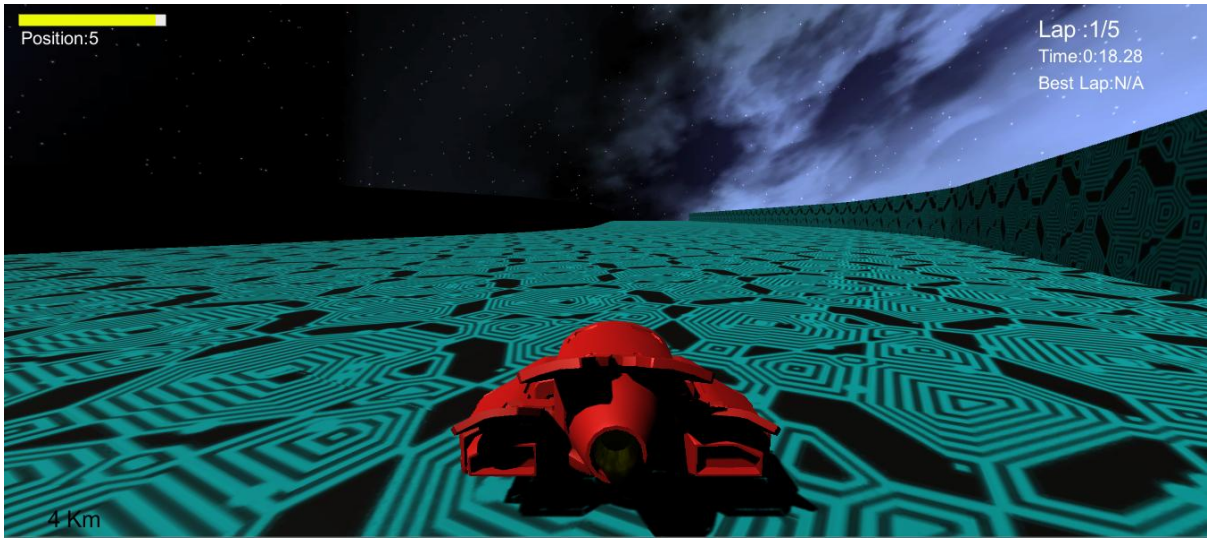
In this section I will be dealing with the user interface which involves the player's controls and display.

Controls



The controls for this game are the keys W,A,S,D for movement of the player's car. W is used for the forward direction and S for the reversing/breaking. A , D are used for right and left respectfully. The space bar key is used for applying the speed boost to the car while the P key is used to pause the game.

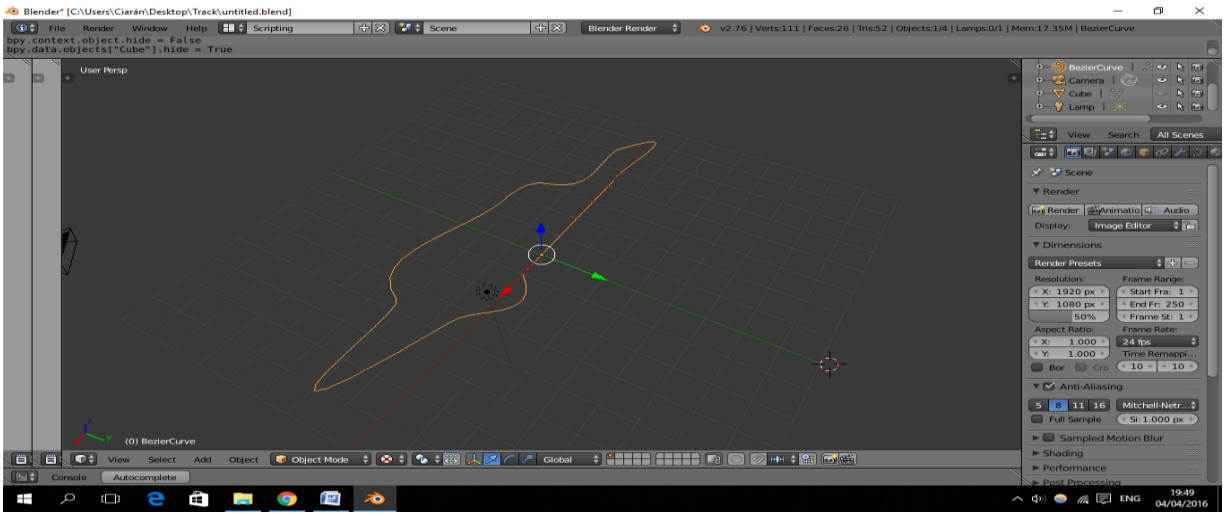
Players Display



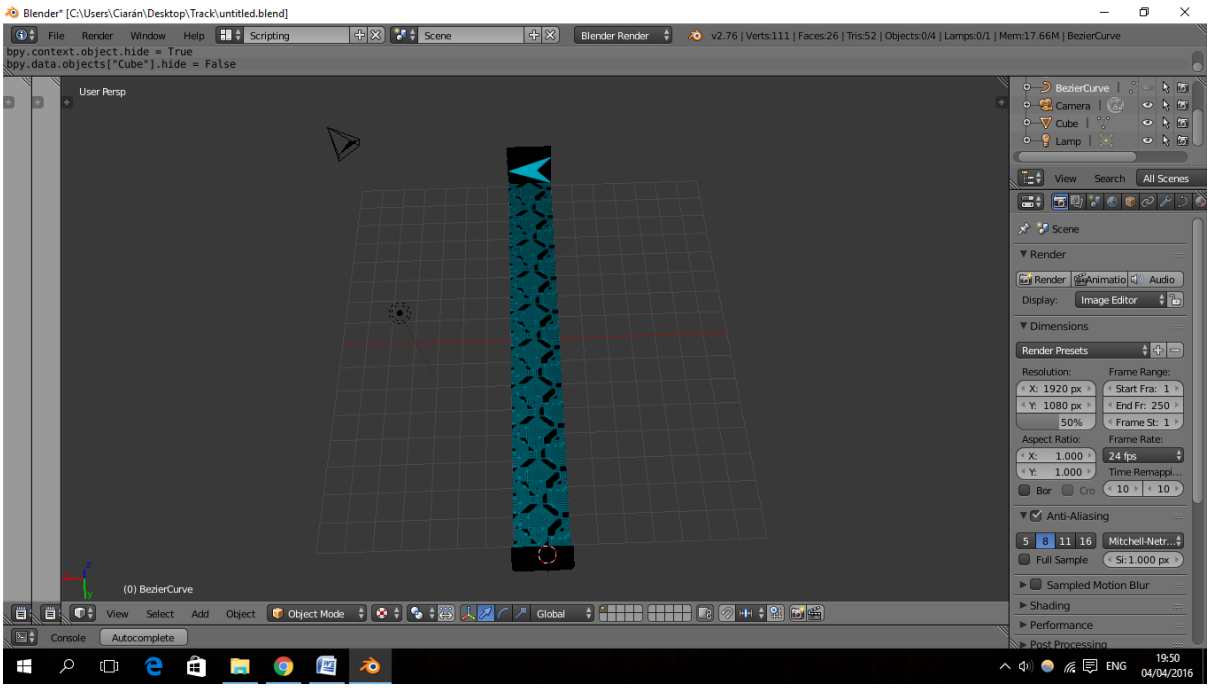
The diagram above shows exactly what the players see when playing Formula X. On the top left hand side is the current position that is displayed and is updated constantly. The position display changes when a player gains a place or losses a place depending on whether he or she is passed or overtaken. Under the current position display the speed boost meter can be found. This is a full yellow bar when none of the boost has been used by the player this lap. If the boost bar is white and fully drained of yellow that mean that there is no speed boost left to use. This means the player has to wait until he or she passes the finish line to recover the boost. On the bottom left hand corner the player will see a speedometer which keeps track of the player's current speed. The speed is worked out by the command velocity magnitude in unity. The measurement of the speed is in kilometres as can be seen in the diagram but this is just for show and does not give any real indication of actual speed in kilometers. At the top right hand side of the screen is the player's current lap time that is being displayed. The current lap time is constantly being updated, to give an accurate reading to the player of the current pace. The current lap time display is reset at zero once the player sets all the sectors to true by passing them. Below the current lap time is the best lap time. The best lap time is updated once the player sets a time that beats his or her personal best lap time and the player must also pass all the sectors to set the lap time in the first place.

Design

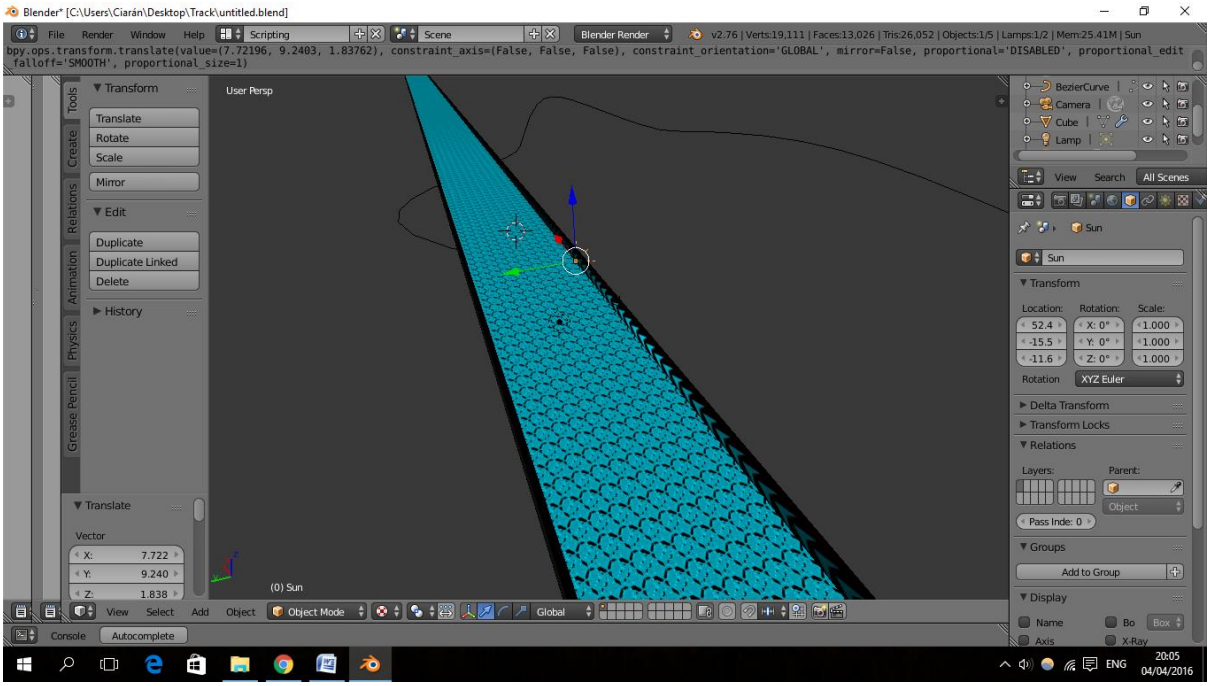
1. Create Benzie Curve



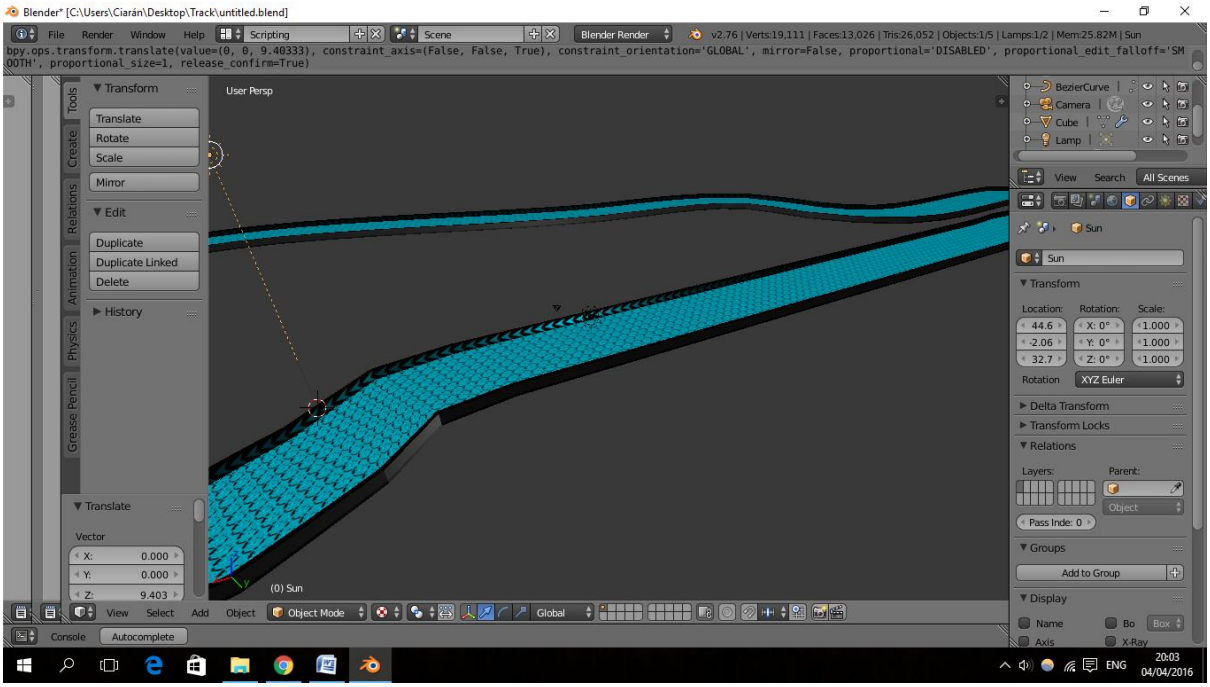
2. Create part one section of track out of an object



3.Creat array of the track section that's as long as the Benzier curve .



4. Wrap the new straight line of track around the Bezier curve.



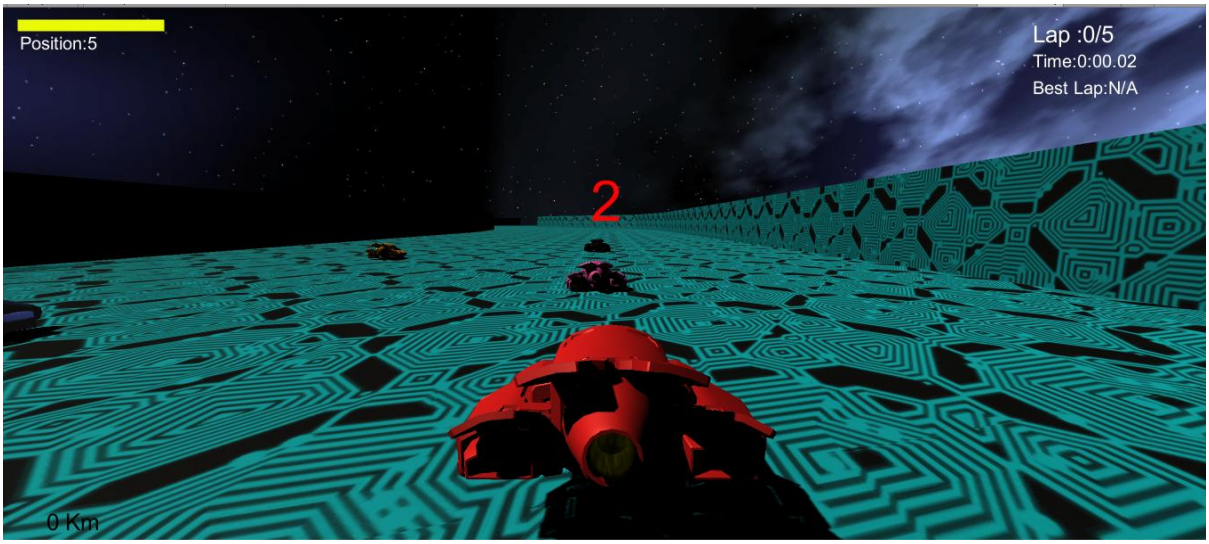
For Formula X I wanted a futurist look with a hint of a dark side to it. To achieve this look I used a space themed skybox which made the races look like they were held in the middle of space. This skybox really did add to the sci-fi look of Formula X. The players and opponent's hover cars were bought and downloaded online from the unity asset store. The cars that were downloaded fitted the style that Formula X was trying to present to the players. The player's/opponent's cars and the skybox

were the only assets that were not built by me everything else that can be seen in this game was created by me. I used unity itself to create the finishing line billboard that can be seen at the end of the finishing line. The two tracks in Formula X were both designed in Blender. This was done in Blender by first creating a Bezier curve which the track itself would be shaped around it as you can see from diagram 1. Once the Bezier curve was shaped the way I wanted the track layout to be, I then created an object in Blender. This object was edited until it looked like a section of track as you can see above in diagram 2. Once the object resembled a section of track, I created an array of the track section object which was the same length of the Bezier curve. A long straight was then created from the array of the track section which was the same length as the Bezier curve as you can see above in diagram 3. The long straight was then wrapped around the Bezier curve as you can see from diagram 4 and thus a track for Formula X was created. The track for the first level was a simple oval design based on the Nascar tracks. The reason this kind of track was chosen for the first level was because it was a nice simple track for the player to get familiar with the controllers of the car. For the second track I made the track a lot more complex, this was done with the hardcore racing gamers in mind. A Tron style texture was given for the second track so that again the game would give off that sci-fi vibe.

Feature Set

In Feature set I will discuss the entire main features that make up my game such as hovering, car controls, Ai and so on. The heart of my game is within this section of my documentation.

Start



In Formula X at the start of the race there is a countdown which starts from three and ends at zero. The countdown is used to begin the start of the race as you can see in the diagram above the countdown of the race is at two. The way the code work for this was that the script would freeze time and thus none of the cars could move. This was done with the code Time.timeScale = 0f;.

Hovering

One of the main things that I wanted to implement in my game was the hovering effect on all the cars in the game. Having cars that can hover will instantly give the user a futuristic vibe which will set the user in the right frame of mind. Initially I was considering putting box colliders under the craft which made the craft appear to be floating. I did not go with this approach due to the static nature of the floating. The craft did not move up and down at all the way a hovercraft would. To give the craft a realistic hovering effect I decided to use raycasts to solve the problem. I added objects to the bottom of the race craft to act as thrusters. Once these thrusters are within a certain distance from the ground then an upward force is added to the race craft, more force is applied if the thrusters are closer to the track. These thrusters then created another problem which was the stability of the craft. The craft would often be uncontrollable and would often flip over which would cause the craft to be stuck. To overcome this problem I added the thrusters evenly on the craft and froze the rotation which stops the craft from rolling. The end

result of this was a realistic hovering motion on the craft which was not static movement.

Racing Car Controls

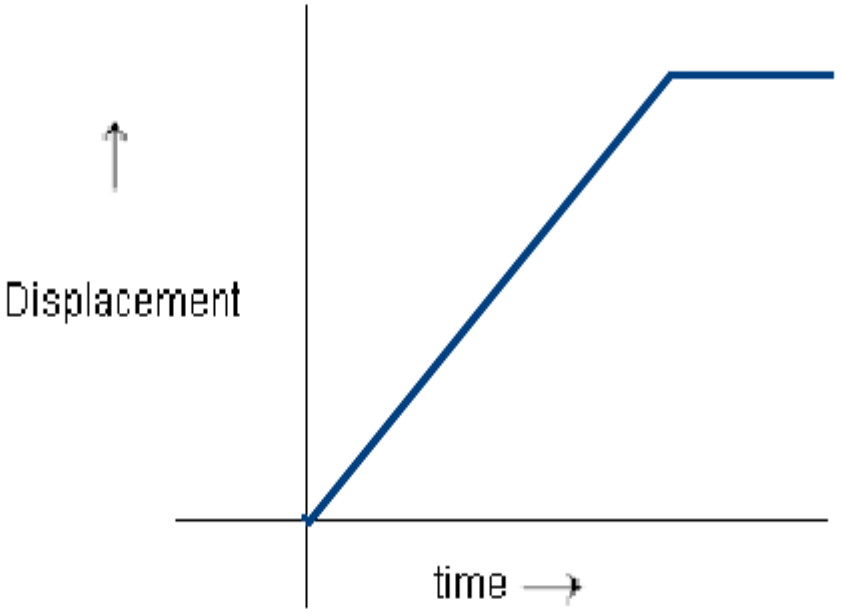
```
public float maxSpeed;
public float topSpeed;

private float currentSpeed;
private float speed;
private float accelerationPercent;
private float rotationVelocity;
private float groundAngleVelocity;

Rigidbody body;
void Start () {
    body=GetComponent<Rigidbody>();
}
void Update(){
    currentSpeed = body.velocity.magnitude;
    accelerationPercent = (currentSpeed * 100) / maxSpeed;
    speed = (accelerationPercent * topSpeed) / 100;
    //check if we are touching the ground
    if (Physics.Raycast(transform.position,transform.up*-1,3.0f) && speed<topSpeed){
        //when we are on the ground enable the accelorator and work out how much acceleration
        //calulate foward force:
        speed += acceleration;
        //Vector3 forwardForce=transform.forward*acceleration*Input.GetAxis("Vertical");
        Vector3 forwardForce = transform.forward * speed * Input.GetAxis("Vertical");
        //Correct force for deltatime and vehical mass:
        forwardForce =forwardForce*Time.deltaTime*body.mass;
        // Add the force to the space craft
        body.AddForce(forwardForce);
```

The car controls for a racing game is the most important element to get right. The controls for Formula X are difficult to get used to which was the result I wanted it to achieve. Formula X was made for the experienced racer that wants challenges. Although the controllers were challenging they were very satisfying for the player once mastered. For forward movement of the players craft I used the addForce which add force to the body of the car. The code took account of the mass and the current acceleration of the craft. The acceleration was also restricted once it reached the max speed no more acceleration was added to the car.

Acceleration of player car



The code that was needed for acceleration was initially more complex than I thought it would need to be. I had to work out the current speed of the craft which was done by using the unity command velocity.magnitude. Since the measurement of the speed of velocity.magnitude was different to the acceleration entered; for example a calculation needed to change miles per hour to kilometres per hour. So I had to find the velocity.magnitude of the players craft when it reached its top speed which I was able to find out in the inspector in unity where I was able to see the current velocity.magnitude. Once I found out the top speed, I was able to set the max speed in terms of velocity.magnitude scale. By having the top speed of the two scales this then allowed me to give realistic acceleration to the player car by using the calculation that can be seen in the diagram above. For the steering of the player car I used the unity command of AddTorque() on the body of the car object. The calculation was done by multiplying the rotation rate by the mass of the car's body and times the input of the use which then gave the car the ability to turn.

Laps

One of the most crucial parts at the start of development of formula X to get right was the laps because all the code in this game revolves around laps for example lap times, positions and speed boost. My first version

of the lap counter work by an invisible object that covered the finish line of the track. I set this object to be a trigger and once a car with a player tag or AI tag went passed the object I add a lap for that particle car. I used the void OnTriggerEnter(Collider col) for the finishing line collider which then update the lap on the car that passed by. There was a flaw with this lap system because a player would only need to move back and forth where the finishing line was to update his or her lap counter which is far for ideal. To combat this issue I implemented sectors which were invisible objects with triggers basically like the fishing line object. Once each of these sectors and the finishing line were passed then the car's lap was updated. The code would set each sector to true if a car passed it. I used an array of cars so that the code was not shared by multiple cars. By which I mean that a car passing sector one would only set true for that car itself and not for the other cars in that race. Once all of the cars have passed all the sectors then the lap counter will be updated by one and all the sectors all reset to false for that particular car. A start line was also needed to be added in because once all the lap would update on the last sector rather on the finishing line. This is generally the reason why the start line is not in the same place as the finishing line in a race such as formula 1. Having the laps individually for each car object was also a way to allow cars to overlap each other.

Time

Formula X also keeps track of the players best lap time that he or she produces around the track. The best lap time give the user a sense of accomplishment if he or she produces a really good lap and it also allows the user to keep track of their progress. The best lap time was updated once the player produced a faster time and has just completed the lap by setting all the sectors to true. Formula X had a last lap time initially which kept track of the users last lap but I felt that the display had too much going on that it would take away from the game so I decided to take it out. If there was more time to develop Formula X the best lap time would have taken in the computer racers best lap to give a best overall lap time. There would have also been a personal best lap time for the player that would work the same way as the best timer does now. Formula X would also have had a delta time that worked out how far up or down the player was to the best lap time.

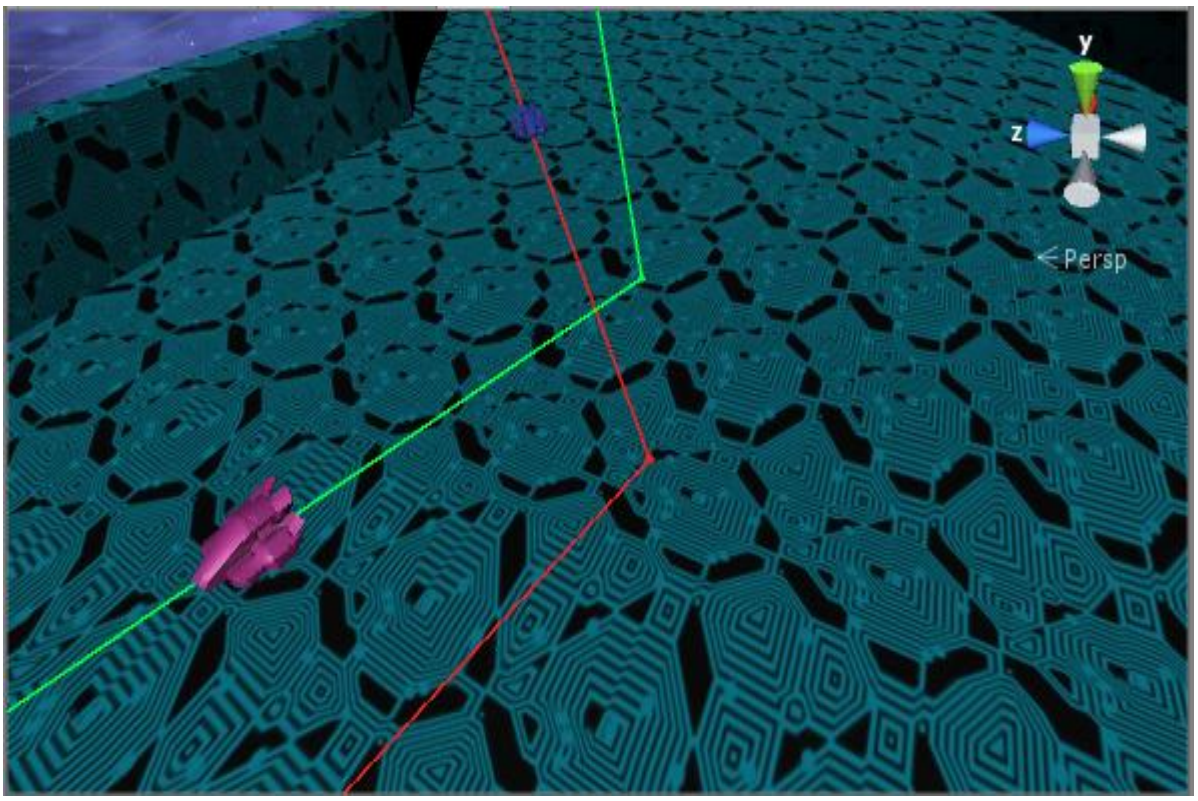
Sound

```
// Update is called once per frame
void Update () {
    AudioSource audio = GetComponent<AudioSource>();
    currentSpeed = body.velocity.magnitude;
    percentage = (currentSpeed * 100) / maxSpeed;
    currentPitch = (percentage * maxPitch) / 100;
    audio.pitch = currentPitch;
    currentVolume= (percentage * maxVolume) / 100;
    audio.volume = currentVolume;
}
```

Sound is another key area for a racing game to get right, sound in the racing game genre is arguably more important than in any other genre of gaming. The sound of the racing car needs to fill the user up with excitement and in my opinion needs to be loud . For Formula X I wanted to give it a dark sound so I added in sound a soundtrack to my main menu page which I feel works well with the game. I picked this sound track for the main menu page because it had a similar style of music to John Carpenter thing soundtrack. For the in game play itself I chose a fast pace sound track which gave the user a sense of speed and excitement. The in game music also has a futurist tone to it which would give Formula X that advanced sound. For the engine noise of the player car I used a formula 1 Honda engine from 2014. I edited the song by using FMOD studio to give the desire sound I wanted. I found the video online of the Formula 1 engine and just downloaded the sound to an mp4 file. I then imported the sound to FMOD where I started editing the engine sound. I first started with cutting the sound down to a second so that when the sound would replay it would sound the same form the start to the end. This was done for the user not to notice the loopback of the engine sound. I also gave the sound a higher pitch so the car would sound like a formula 1 car from the year 3000. Once I finished editing the engine sound on FMOD, it was then imported in unity and attached to the player's car object. There was also code attached for this sound

which would change the pitch and volume depending on the current speed of the player's car. The faster the car would go the higher the pitch and volume of the engine sound. This gave the player a realistic acceleration sound with the engine. The code for the engine pitch and volume worked the same way as the code for acceleration. Depending on speed the pitch and volume would change as you can see from the diagram above. If the player accelerates the pitch and volume would go up but the case is the opposite when the player decelerates.

Artificial Intelligence



One of the harder elements to code was the artificial intelligence (AI) of the opponent's cars. To get the opponents to go around the track I used waypoints for the AI to follow which acted like a racing line. I had an array of way points which cover the track from start to end. In the code the distance from the waypoint is worked out from the AI's current position. The Vector3.MoveTowards() command of unity is used to move the position of the opponent's car towards the waypoint. Once the waypoint has been reached then the opponent moves on to the next waypoint in the list of waypoints. The opponent's car are also roatnates towards the next waypoint with the Quaternion.Slerp command of unity.

Once the opponent's car reached the last waypoint in the list the opponent goes back to the first waypoint which is zero in the array. I found that only using one list of waypoints around the track gave the opponents cars very predictable movement as they all followed the same line every lap. This made the opponents cars seem very robotic in their movement and which in turn made the experience with the AI fairly boring to the player. To combat this array of waypoint lists was created to generate some different racing lines for the opponent's car. By doing this opponents would change their racing line once per lap and would differ from each other. Random.Range(0, racingLine.Length) was the command used to generate randomly which racing line was going to be used by a particular opponent's car. The change in racing line happen one per lap of each opponent's car that finished their respective lap, once the lap was finish a new racing line was randomly picked for them. If there was more time to work on Formula X, there would have been more advance AI that would have gone of its set racing line if the player was close to it. This would have been done with ray casts at the sides of the opponent's car and action would take place depending on the personality given to the AI driver. If the AI driver was more aggressive then the AI would move the car towards the player or if the AI was more timid then the AI would move away from the player as he or she draws closer. I wanted each AI driver to have it's own personality in formula X very more like the ghost in Pac man. For example in Pacman "Blinky begins each level moving at the same speed as all of the other ghosts" and "Unlike the other ghosts, Blinky will also tend to follow close on your tail even when you turn and will often still chase you even in scatter mode". "Clyde is either short-sighted or stupid. He will often turn off rather than approach you. His heart doesn't seem to be in it at all. A consequence of Clyde's unwillingness to take part is that it's often hard to round all of the ghosts up into a single cluster which is nice to do just before eating a power pill". These different character traits that each of these Pac man ghost is something that I tried adding into Formula X in terms of max speed, acceleration and steering angle but I would have liked to be able to have the AI behave differently if the player was close to it.

Creating Waypoints list

As was explained in the heading above waypoints were used to create a racing line for the opponent drivers. This was done by writing code which would allow all the empty objects that were created in the unity scene to connect up with each other. This was done by attaching the RacelineEditor script to an empty object which will be called racing line. Once two or more objects were added as a child to the racing line then a line was created between the children of the racing line. In the scene the empty objects that I will call points were set around the track. Once this was done the points were added as a child to the racing line, this caused a line in the scene which however cannot be seen playing the game. The line was created by the Gizmos.DrawLine() unity function.

Position

```
bool sectorDistance(){
    //sectorNum+1 is used to check the next sector ahead
    nextSector = pos[i].sectorNum+1;
    if (nextSector==sector.Length){
        nextSector = 0;
    }
    float distanceA = Vector3.Distance(sector[nextSector].transform.position, pos[i].transform.position);
    float distanceB = Vector3.Distance(sector[nextSector].transform.position, pos[i-1].transform.position);
    if (distanceA<distanceB){
        return true;
    }
    return false;
}
```

Working out the positions of each individual car in the race was the most difficult thing to achieve in this project. The position was worked out by first comparing what lap was each driver was on, the driver with the most laps completed were put ahead. If the drivers were on the same lap then the drivers with the most sectors completed in that lap were put ahead. Each sector was give a number and once the driver passed that sector then the number was given to the car object which then be able to work out which driver were further up on the lap by then sectors. If the driver was on the same lap, same sector then the position was worked out by which driver was closest to the next sector on the

track. This was done by getting the current position of each car and comparing the distance between them to the next sector. The driver that was the closest to the sector was put ahead. The command that was used to check the distance between the cars position and the next sector was “Vector3.Distance(sector[nextSector].transform.position, pos[i].transform.position”); which can be seen in the diagram above. All this working out of the position was done inside a method call swap which returned true if a swap needed to take place between the positions that are being compared which was done if Lap, sector or distance to next sector ended up being true. An insertion sort was used to sort out the array for the positions of the drivers. The insertion sort algorithm is $O(n^2)$ which is the same as bubble sort but it is still quicker than bubble sort. “The maximum number of comparisons for an insertion sort is the sum of the first n−1n−1 integers. Again, this is O(n2)O(n2). However, in the best case, only one comparison needs to be done on each pass. This would be the case for an already sorted list.
One note about shifting versus exchanging is also important. In general, a shift operation requires approximately a third of the processing work of an exchange since only one assignment is performed. In benchmark studies, insertion sort will show very good performance”.

Testing

Formula X had a wide variation of people testing the game. There was a lot of interesting results gathered from the testing that point me in the right direction with my project. One of my tester was named Patrick aged 23, who is an experienced gamer however was a novice when it came to racing games which he admitted to me during testing. Patrick found the initially experience of the game to be too much of a learning curve and had some great difficult to getting a grip with the controller of the car. I have another tester called Sean aged 22. He found the Formula X controllers to be very realistic and liked the fast pace nature of the game. Sean felt that the cashing caused too much spinning and it was too difficult to regain control of the car again once the wall was hit. With this feedback I decided to increase the angular drag of the player’s car so that the spinning would be reduced. I also took on board Patrick comments on the game by reducing the difficulty of the first level of the game.

Future Work

In the future work section I will discuss what i will include in the coming months in my game formula X which I couldn't include in the last six months due to the time restriction.

Story

The first thing that I wanted to include in my game is a story line to match the intensity of Formula X fast paced nature. I want a dark story to my racing game because a story is never really done in a racing game let alone a dark one. The story that I have in mind was about a drive that managed to just get into the big time that was Formula x. This driver has been dreaming about this moment his entire life he grew up watching and looking up to his hero's that were Formula X drivers. This driver devoted his whole life to get to formula X leaving friends and family on the way. In his first practices session with his new team he had a bad crash which caused him to go to get a check up by the medical doctor. On examination the doctor found something he wasn't looking for and diagnosed the driver with a severe illness that if not treated could be terminal. If the driver takes the treatment it would mean the end of his racing career as the treatment will make him weak and unable to race. The driver has to make a choice his life or his dream that he has been working towards his whole life?

Improve AI

In the future I will also plan to improve the AI in Formula X so that the player can enjoy interacting with the AI more. The current AI could be made less static and be given more of a personality. Like I mentioned before I would like to have The AI drivers to each have their own unique characteristics of their own.

Tracks

In the future work of Formula X I will add in more tracks that will excite the users. As I explained early in this document there are only two levels available to the player so far. I would like to be able to expand upon that. The more tracks that will be available in Formula X will allow for a more robust experience. The tracks will again follow the same formula as it does now in which for every new level that is created the track gets

harder and harder rather than the opponent getting faster. These new tracks could be set in more exotic locations such as jungle or a ruin of old city from a civilisation that is now gone. The most disappointed element that has not been implemented to Formula X is multiplayer function. Multiplayer is perhaps one of the most fun and competitive elements of a racing game and indeed most games in majority of genres multiplayer is the first new feature that will be implement in the newest version of Formula X . The multiplayer will consist of a simple split screen there will be a calculation done that will split screen up depending on the number of players that are taking part in the race. Formula X with multiplayer would work so well especially considering the changeling nature of this game.

Time Trial

Formula X lacked a time trial system in the final completion for the development project. A time trial element would be made in mind for the serious racers out there that want to test themselves and set the best lap time possible around a partially track. The time trial system in Formula X will be like most racing games with this feature, in that it will be a single player event where it's just the player against the clock. The Speed boost will also still be allowed in this mode and also the player's best time will be permanently saved.

Car Selection

Car Selection is a cool feature that couldn't be implemented in time for the demo. This is a feature that I am keen on introducing to formula x. The ability for the player to pick his or her car and chose what colour it could be will give the player a felling of self expression that he or she cannot currently feel in Formula X. Each type of car will have its own set of advantages and disadvantages such one car could have good acceleration but terrible max speed. Alternatively another car could have terrible acceleration but great top speed.

Conclusion

I have now gone through all of the aspects of Formula X from user interface to finally testing. Testing told us that Formula X has a bit of a learning curb but with time it can become a very fun game to try to master. From a design point of view Formula X still needs more work to

be done but I feel that the game play is of a high standard. My overall conclusion is with some more time Formula X will need a bit of polishing of the AI code and other areas but Formula X can become an even more interesting and enjoyable game than it already is.

Bibliography

Assets

Asset store (no date) Available at: https://www.assetstore.unity3d.com/en/#!/content/28663 (Accessed: 4 April 2016).

Skybox:

https://www.assetstore.unity3d.com/en/#!/content/3392

Bevilacqua, F. (2013) *Understanding steering behaviors: Path following*. Available at: http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-path-following--gamedev-8769 (Accessed: 3 March 2016).

Blender Free Tutorials by ianscott888 (2013) *Blender Tutorial making a racing track for a car game (design from Scalectrix track designer)*. Available at: https://www.youtube.com/watch?v=SDLLbKvEeBY (Accessed: 4 April 2016).

Miller, B. (2014) *The insertion sort — problem solving with Algorithms and data structures*. Available at: http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html (Accessed: 4 April 2016).

Millington, I. and Funge, J. (2009) *Artificial intelligence for games*. United Kingdom: CRC Press.

Pacman ghost psychology (no date) Available at: http://www.webpacman.com/ghosts.html#personalities (Accessed: 4 April 2016).

Citations, Quotes & Annotations

Asset store (no date) Available at: https://www.assetstore.unity3d.com/en/#!/content/28663 (Accessed: 4 April 2016).

(*Asset store*, no date)

Bevilacqua, F. (2013) *Understanding steering behaviors: Path following*. Available at: http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-path-following--gamedev-8769 (Accessed: 3 March 2016).

(Bevilacqua, 2013)

Blender Free Tutorials by ianscott888 (2013) *Blender Tutorial making a racing track for a car game (design from Scalectrix track designer)*. Available at: https://www.youtube.com/watch?v=SDLLbKvEeBY (Accessed: 4 April 2016).

(Blender Free Tutorials by ianscott888, 2013)

Miller, B. (2014) *The insertion sort — problem solving with Algorithms and data structures*. Available at: http://interactivepython.org/runestone/static/pythonds/SortSearch/TheInsertionSort.html (Accessed: 4 April 2016).

(Miller, 2014)

"The maximum number of comparisons for an insertion sort is the sum of the first n−1n−1 integers. Again, this is O(n2)O(n2). However, in the best case, only one comparison needs to be done on each pass. This would be the case for an already sorted list. One note about shifting versus exchanging is also important. In general, a shift operation requires approximately a third of the processing work of an exchange since only one assignment is performed. In benchmark studies, insertion sort will show very good performance." (Miller, 2014)

Millington, I. and Funge, J. (2009) *Artificial intelligence for games*. United Kingdom: CRC Press.

(Millington and Funge, 2009)

Pacman ghost psychology (no date) Available at: http://www.webpacman.com/ghosts.html#personalities (Accessed: 4 April 2016).

(*Pacman ghost psychology*, no date)

"Clyde is either short-sighted or stupid. He will often turn off rather than approach you. His heart doesn't seem to be in it at all. A consequence of Clyde's unwillingness to take part is that it's often hard to round all of the ghosts up into a single cluster which is nice to do just before eating a power pill." (*Pacman ghost psychology*, no date)

"Unlike the other ghosts, Blinky will also tend to follow close on your tail even when you turn and will often still chase you even in scatter mode." (*Pacman ghost psychology*, no date)

"Blinky begins each level moving at the same speed as all of the other ghosts, but after you've eaten a certain number of dots, he begins to speed up." (*Pacman ghost psychology*, no date)