



UNIVERSITY OF LINCOLN

An investigation into the effect of automation in
vehicles on user perception and driving ability

Joseph Ryan Martin

MAR15596040

A thesis submitted in partial fulfilment of the degree of
BSc(Hons) Computer Science

School of Computer Science

University of Lincoln

2019

Acknowledgments

I would like to thank my girlfriend, Cookie, for her constant love and support during the entirety of this project, and my 3 years at University. She gave me motivation and strive to be able to complete this project to the best of my ability, and she gave me something to be proud about, and I will always be grateful.

Abstract

This project presents a successful implementation of a car driving simulation, to test data on humans including reaction time, road position and more, with the variance of changing the cars automation states. This includes car sensors, collision detection and car forces. The test results gained from this experiment back the thesis that automation does not have a short-term impact on human perception and driving ability, but does conclude that such tests are hard to carry out in simulation due to the difference on how participants take the tasks seriously as it is not based in real-life. Research was done in both wheelchair and driving simulation, and driving states, to determine how the human brain perceives certain actions. There was also some research done into track design to improve the feasibility of the simulation and test results.

Background and Literature Review

There are several researchers looking into personalised track design, which takes user data and tries to alter track layouts and maps to try and map the specific user (Theodosios Georgiou, 2016). His work is aimed at improving tracks, to lower average time to complete. I believe that this work can very much be tied into the work of automation in vehicles. Data could be taken using cars, into the performance of roads used in the general public. This could be used to try, and help reduced traffic flow, accidents, or could even be used in the future when planning new roads to be built. More specifically to my thesis, if these pieces of data were to be recorded on public roads, data could be analysed from both automated vehicles, and manual vehicles to see if there is a difference. Data gained from this could help improve the safety of the automated vehicles in terms of how they themselves perceives other vehicles, pedestrians, and the roads itself. With the rise of automation in vehicles, more tracking and more data can be gathered, which could lead to more efficacy in roads, leading to lower travel times and less human involvement. Even though his methodology was more aimed towards altering tracks in simulation to try and get the best times out of candidates, I do believe a more practical use could come from it.

Alongside human interactions with vehicles, smarter cars and researchers are trying to predict future car states (T. Georgiou and Y. Demiris, 2015). His work used machine learning to try and predict certain aspect of a car's behaviour, including position and speed. After a few tests, the system already started to show improvement when it came to its predictions. This work is very much linked and helps me with my thesis as it backs the idea of smart technology in vehicles and how we can use it, learn from it, and benefit from it. With his research we're able to better predict future car states, unlike we could before. This data could be then used, alongside GPS, and more car senses, to improve safety along the road, and also reduce human input. Vehicles could combine predictive data and sensory data to better help predict collision with obstacle. Other impacts it may have away from safety could include better parking assistance and lane switching. With all of these positives that the research puts forth, it also links with my research due to its impact on human control. If more research is going into the world of automation, human input in vehicle is much lower. There is recent Tesla technology that allowed cars to use sensors and GPS navigation to complete a full journey without any interference from the driver (Tesla.com, 2018).

Methodology

Project Management

Due to my project being software based, most of the time should be focussed on creating a realistic and immersion environment so I am able to gather the best data possible to try and back my thesis. As I need to create software, I need to ensure that I plan out every step I will take, with a look into risk assessment so that I have time to make adjustments. Things that will be considered during this project will be the time it takes to create the designs for the simulation. The design is extremely important, as it means I can plan out the time it will take to implement, as well as the time it shall take to do any necessary research into the areas in which I have a cap in my knowledge. Setting deadline can help focus the programmer, as it will allow them to split their time appropriately. It also allows them to give themselves good stopping points so that they don't get stuck and obsessed with perfecting one area, when several parts of the software may still need implementing. To help ensure I stick to this methodology, I am splitting the workload into several different sections, and assigning it into a Gantt chart. This includes the amount of time I spend researching assets to create the immersive world, as this section should not be taking a lot of time, when programming takes more refining and has more of an impact. More time is going to be given to sections such as programming the sensors I add onto the vehicle. This is a very important part of the experiment, and it will affect lots of different pieces of data and will affect a large majority of the immersion and results, in terms of the computer-human interactions.

Software Development

My project is going to be a car driving simulation, based around the automation of vehicles. The software that I chose to use, needs to be user friendly, easy to use, and needs to be able to fit my specific needs. This includes, adding assets and models from online, adding premade scripts created by other users, and editing models in the world. The project is designed so that the user can test their performance and to see if the thesis of automation has an affect on their perception and ability. The aim is not to learn how to use a more advanced piece of software, or learn how to model detailed cars, with realistic car mechanic and physics. Taking this into account, I believe that Unity will be the best piece of software for me to use. The software is free to use, and is one of the largest on the market, making anything I don't know very easily accessible. From car models, and trees and other world objects. I am also used to using this software and know how to navigate and manipulate the UI. When it comes to designing how the automation of the vehicle will work, in terms of sensors, I already know about how that Ray Casting is very easily accessible in unity scripts. This is combined with the easy ability to add objects onto objects, creating compound objects. I can add invisible objects onto a car model and have this act with the car itself. For example, moving and rotating in the same manor. The Software has a very large store filled with free to use assets, as well as a large forum and community so I can find answers quickly to problems I may encounter.

Design, Development and Evaluation

Design

Main Menu

The background to the environment is going to be pre-defined landscape, that includes features of the levels that have been created. The camera will rotate around a centre point, so that the user has more to see, and feels more immersed into the simulation. Aspects of the environment may include trees, roads, houses, and hills. Other than the buttons on the screen, no part of this menu will be interactive. When a user clicks onto a button, instead of loading a new scene and creating a new menu, buttons will be removed from the scene and added. This will increase the speed and will allow the background to remain consistent throughout.

The buttons on this menu will direct the user to various areas in the simulation. Appearing first in the main menu will be a “Play” button. This button will direct the user to the list of various simulations that they can partake in. This include a reaction time test, both manual and automatic, and will also include a course, which again will be available with both manual and automatic. For the main menu and level select menu, there will also be a quit and a back button respectively.

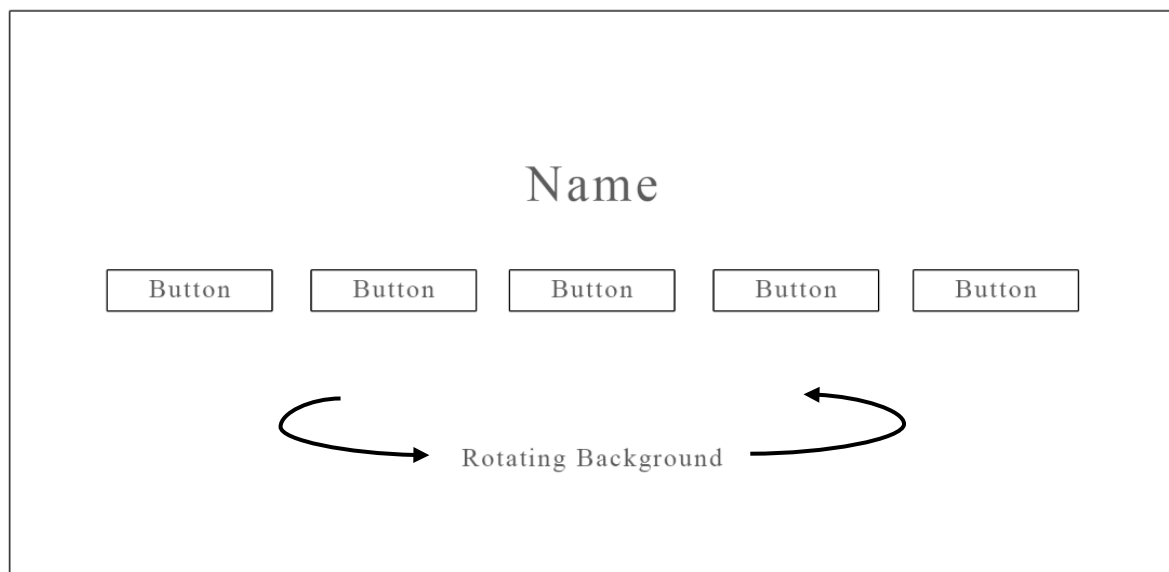


Figure 1: Design for the menu.

Reaction Time

The reaction time test is purely made to test the user's reactions, when that is the sole purpose of the simulation. There will be no other tasks that they need to complete. The environment for this will be clean, with very limited text and distractions on the screen.

The user will begin the test at the start of a long, straight road. The road will be surrounded with woodlands and will have a scatter of houses to add to the realism and the sense of immersion. The test will begin as soon as the user loads the level, and depending on the mode their vehicle is in, the car will start. At a single point down the road, which is randomised upon each run of this test, a stop signal shall appear at the centre of the user's screen, signalling an emergency stop. The user will then have to press the appropriate control, s, to slow the vehicle down to a complete stop, and then the test is over.

For this test, there is very limited live ongoing data shown to the user, in fact, there shall only be one piece. This piece of information will be the speed of the vehicle, mph, and it will be presented to the user at the top left of their screen.

Once the vehicle has come to a complete stop, post data shall be posted onto the user's screen. This means that one, they can see their results, and secondly, I can write down any errors and any results that they have. Data for this is going to include two things. Firstly, it will show the time it took the user to respond to the emergency stop warning. The second piece of data will be the time it took the vehicle to come to a complete stop. Both of these pieces of data will be to two decimal places and will be shown at the bottom of the screen. As well as the data, there shall be two buttons that appear. The back button and the again button, which will be placed in the bottom left and right corner respectively.

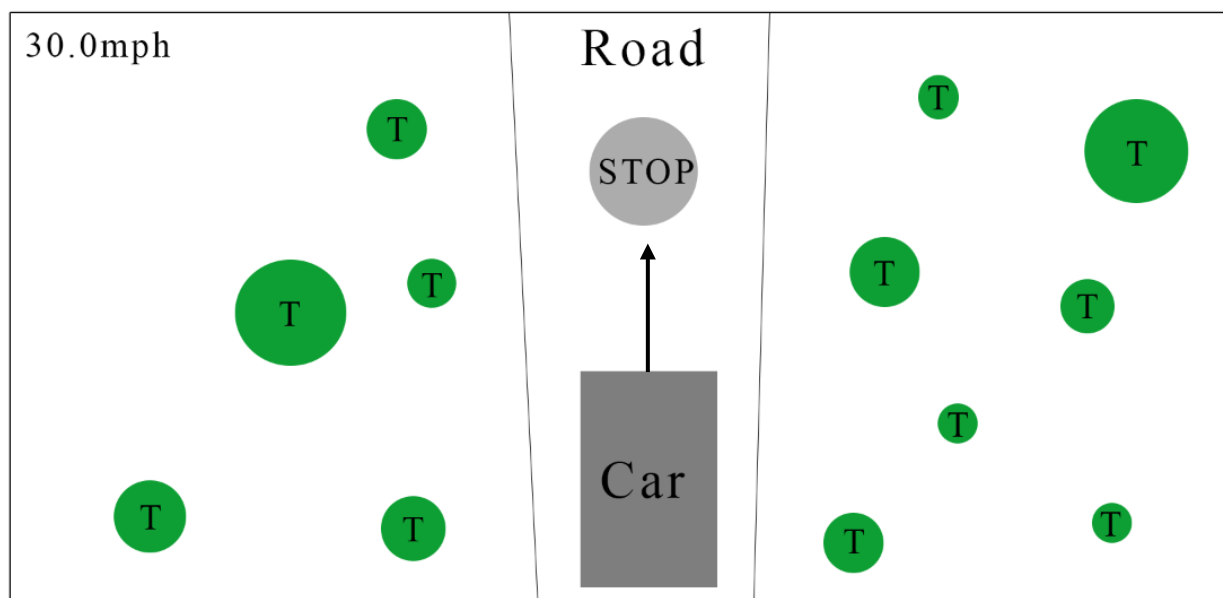


Figure 2: Design for the reaction time.

Course

The course is the main feature of this simulation. Again, it will be available in both manual and automatic mode, but this time there will be added features that shall be discussed later in this design section. The course has been designed in such a way that will allow several tests to be carried out. The course shall include straights, in varying lengths, to allow different speed zones, and shall include but left and right turns of different angles to test the user's ability to stay in the middle of the road. As well as these tests, the course also include an emergency stop procedure. This will be identical to the reaction time test previously, and the data will be show as soon as the test is over. The idea of repeating this test is to see if there is a difference in how the user reacts, when they also have to focus on other things in the surroundings.

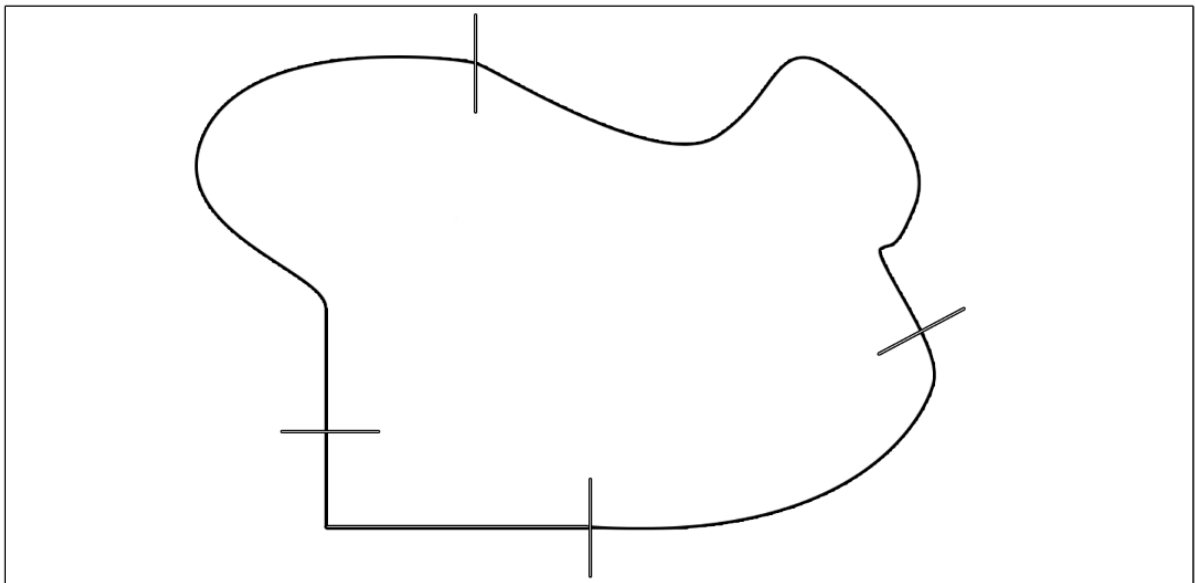


Figure 3: Design for the track.

Again, there is very little live data for this test, as most of it is calculating in the background to be presented at the end. Just like in the reaction time test, the speed of the vehicle shall be displayed in the top left of the screen. Next to that shall also be a time of the current test, in second, so the user knows how long the current lap has taken. Note, that even though the data for the current speed is shown to the user, the speed they should actually be travelling at is not presented onto the screen as text. They will have the manually look at the world to see the speed signs, as they would in a real-world environment.

This test will have much more data shown at the end, compared to the reaction time test. Data shall include: Average road position, average speed, time spent over the speed limit, and the amount of times the emergency sensors were activated. If the user goes off of the track, the test will instantly stop, and a “Crashed” message shall appear on the screen. As well as this, the back and again button will be in the same positions as in the previous reaction test.

The difference between manual and automation is simple the activation of certain car features. For this test, when the car is switched into automatic, features such as front, and side sensors shall be turned on for collision and correction, as well as automatic breaking and speed control.

The environment for this test will be very similar to the reaction time test. However, there will be more emphasis on the building around the corners to create a bigger sense of immersion for the user. Again, there will be there will be trees surrounding the track, as well as a pavement as more of an initiative. to keep the user on the track, as well as to add functionality to coding and add to the immersion.

Building

Menu

The first thing the user will see when they launch the simulation is the menu. The menu has several options, from selecting the levels, reading instructions, and exiting the simulation. On this scene there is a button manager script. Each button is assigned an integer, this integer is then assigned to a certain task. From loading another scene, to setting new buttons active.



Figure 4: The menu with the rotating camera.

To add a simple feature, but yet adding to the immersion, I had the camera rotate around a fixed point on the scene (Sarper-Soher, 2009). This allowed the user to see what the environment was going to look like before they started the actual simulation. It was a simple feature added to try and make the whole simulation look more user friendly and immersive.

Vehicle

As it would have taken too long to have created the vehicle myself in the time given for this assignment, I decided to use the free to use Unity standard assets package (Assetstore.unity.com, 2018). This package includes lots of assets, from different types of vehicles, to different world and environment objects, for my assignment I only used the car. All the scripts were already included for the vehicle, and its controls, I just adjusted them to suit my desired features for the assignment.

The first adjustments I made to the car were to add three sensors. One to the front, and then the other two on either side of the vehicle. These sensors were used to help the car with automatic breaking, turning and collision avoidance with the automatic version of the car. As well as that, these sensors were used to calculate the cars average position on the road at the end of the test.

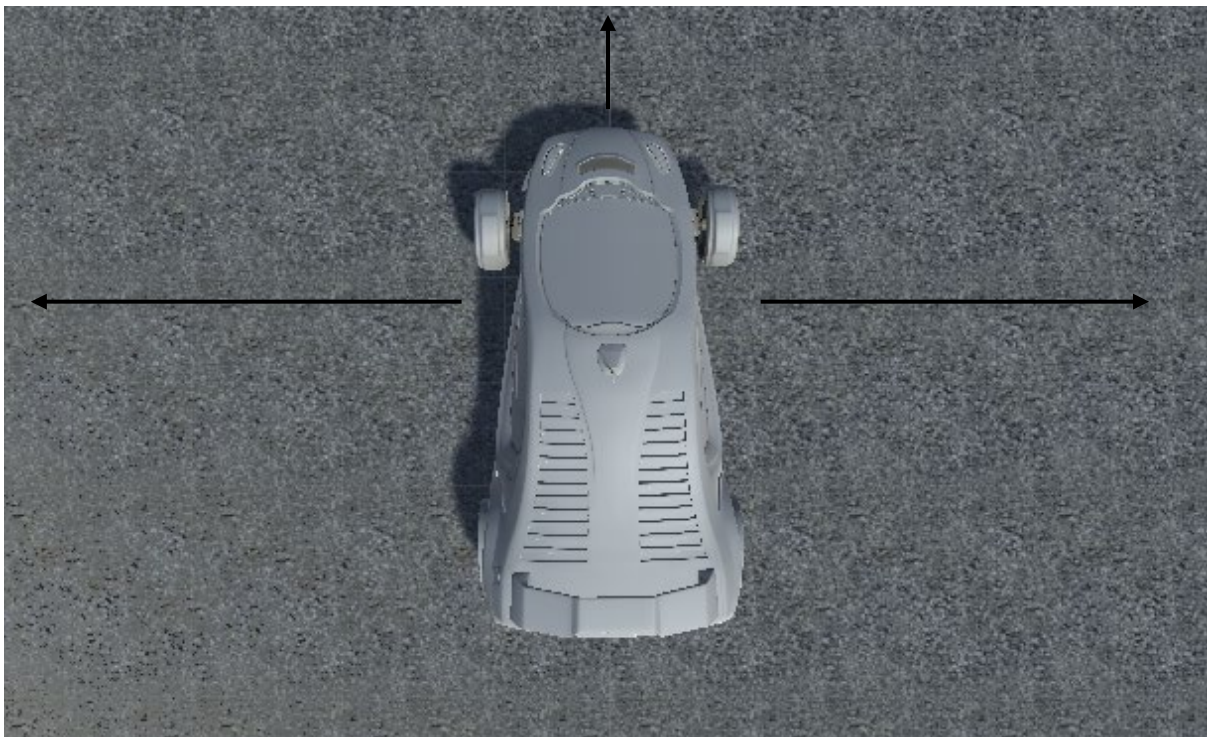


Figure 5: The position and directions of the car sensors.

The method I used to complete this was Ray Casting (Unity, 2019). The ray casts are low to the ground facing in the three desired directions, and as the course is designed with a pavement in mind, the ray cast will hit that before anything else. To ensure that this design worked, I had to ensure that the road was perfectly flat, otherwise the ray would hit against the road itself and then I would receive false data, as well as the car would react to the incorrect obstacles.

When the ray cast hits an object, it returns a float value, that I use a metre. For every frame that program runs, a new ray is cast from the new position of the vehicle (Answers.unity.com, 2014). Once the value is returned, a counter is increased, and the value is added onto another

float value that will be used later to create the average. After these calculations have been done, the program checks to see if the distance is below the value of 3.0f for the sides, and 13.0f for the front. If this is returned as true, the vehicle will gently break, and the turn away from the side it is closest too. When the front sensor detects that there is an object to close, it then gets the values for the left and right sensor, and whichever side has the greatest distance is therefore the direction the car turns towards.

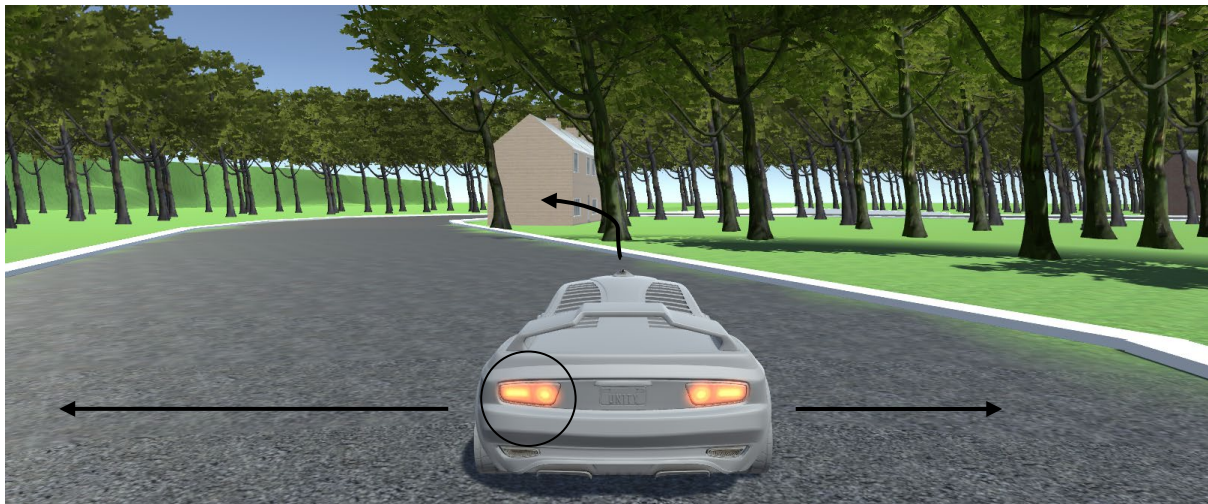


Figure 6: Car reacting to a corner using the sensors.

The above sensors are always active, in both manual and automatic. However, in manual mode, they're only used to calculate the average position of the vehicle, which shall be presented later. Another piece of data that the car is involved with is speed. The car asset package that I am using from unity, already comes with prewritten scripts. Within these scripts I was able to change the code so that I could predetermine the speed. For the manual mode, I let the car run as it was normally intended and distributed by Unity. You have to input the forces by controlling the vehicle. However, for the automatic version of the vehicle, I made it so that it constantly applied forward acceleration, and I capped the speed at the current speed limit, that shall be demonstrated in the course part of this analysis.

The scripts provided were extremely well produced and came with public variables that I could easily access to get data or change the actions of the vehicle. I could easily access the rigid body to apply force and get positional data, and I was also able to access variables such as max speed, and current speed. These variables allowed me to create the rules for the test and change the driving state. The current speed variable also allowed me to present data onto the screen, without me having to take up more processing power in repeating calculations per frame.

Course

To create the track, I wanted to try and make it as realistic as possible, hence why I selected such good pre-created car model (Assetstore.unity.com, 2018). I took the same approach when it came to creating the rest of the levels. I wanted to dedicate the time into research, development and testing, and not learning how to create realistic trees and house models. So, I went into the unity asset store, and I found the Realistic Tree 9 pack (Rakshi Games, 2016) and the UK Terraced Houses Pack (RIK4000, 2017). This was in combination to the tools already available in Unity. This includes the terrain generator, tree painter and texture painter. I was able to add different variations in trees, and I was able to paint the road material onto the predefined grass. Using the brush tools in unity, and dragging the house objects around, I was able to create a realistic environment for the user to drive around in.



Figure 7: Tree and house models used for the simulation

Now that I add all the tools that I required to build the track, I took my design and I implemented it into unity. The design included several turns, and several different straights, designed to test the concentration of the user. I selected a brush size for the road that was with relative scale to a real-life road, so that the simulation seemed realistic.



Figure 8: Birds eye view of the track

Around the entire track, on both the left and right-hand side of the track is a white outline, this is used to represent the pavement. The pavement is a collection of cubes, that are stretched to match the shape of the track. All of them have been attached with a box collider, and have been set to be triggers, so then I could detect when the vehicle had hit them. During testing, I noticed that the car would go straight through the pavement, due to it now not having the collision, and it would also not pick up the trigger on the vehicle wheels. To fix this issue, I raised the box collider to stretch above the actual pavement that was rendered onto the world. Once tested again, the collider detected and outputted into the console that it had been touched. After this, I decided to end in a game state for when the user had crashed.

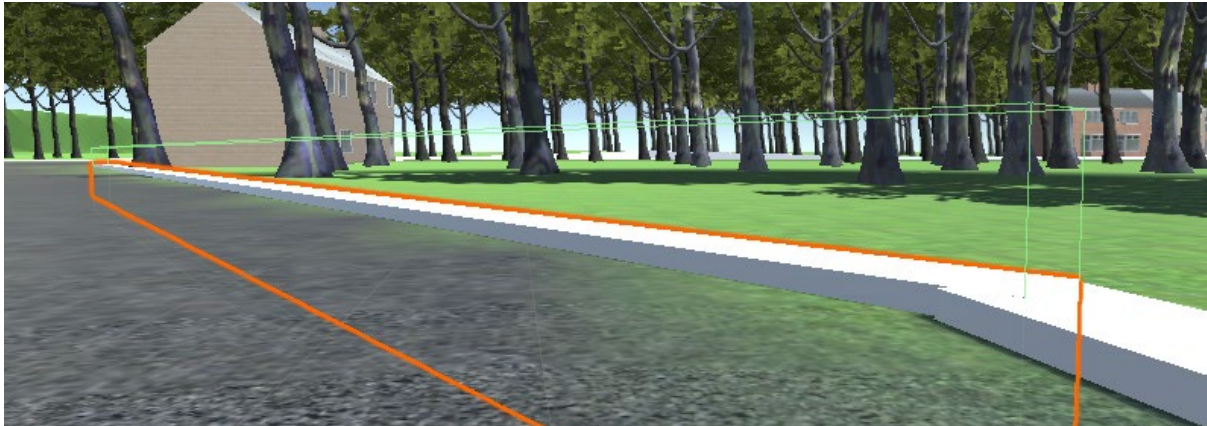


Figure 9: Green trigger box, and orange rendered pavement.

The difference between this and the end of the test screen was that I wanted this one to show a message telling the user that they had crashed, and I wanted the user to not be presented with any post data, as it would be incorrect as it wouldn't be over the entire test. The only similarity would be that the back and the again button would still be visible. To make this possible, I set a Boolean expression in the car position script, called crashed, to true if the trigger had a call back, and I then referenced the finished Boolean expression in the course controller script, and set that to true. When the code for the finished test was run, the script would then reference the average position script, and check the value of the crashed expression. If it was true, it wouldn't do any calculations, and none of the text would be updated.

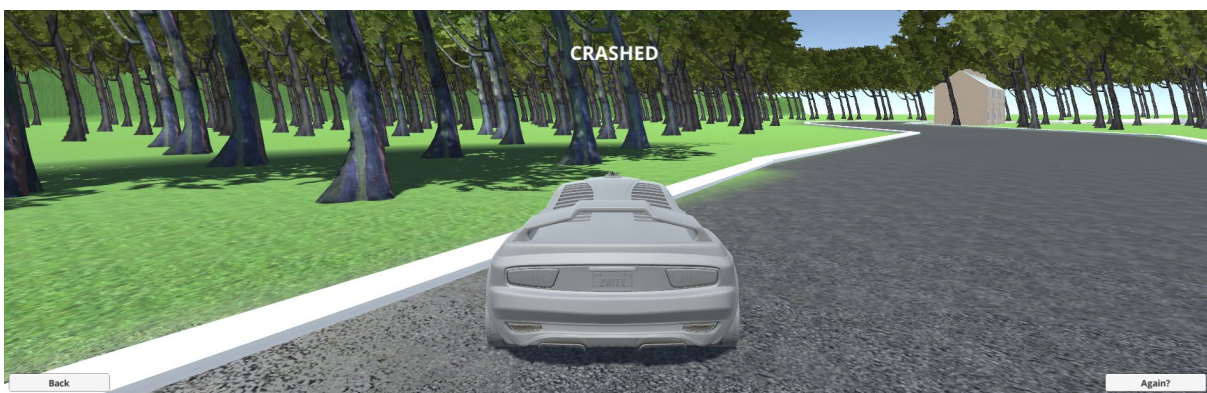


Figure 10: Crashed end state.

The next part I wanted to focus on, was controlling and measure the speed data, this would be apparent in both the manual and automatic version of the car. In both of these versions, the average speed of the car would be calculated, and so would the time spent over the speed limit, however, in the automatic version of the vehicle, the speed of the car would be limited to the maximum speed of the road, to eliminate any error, hence automated. The way in which I controlled this was by using triggers, and a visual asset package. The asset package I used was North American Speed Signs (Kirkley Entertainment INC, 2018). This package, as the name would suggest, included several speed signs. I placed these signs in appropriate places, that would be realistic to real road situation, to force the user into a reaction. For example, increasing the speed limit at a long straight, but decreasing it near tight corners near houses. Next to the sign would be a trigger box, that would stretch along the entire road, and was invisible to the user. The trigger box will be slightly above the floor, as during testing I noticed that sensors on the car were seeing the trigger, and therefore the car was reacting when it shouldn't have been. Once the user hit the trigger, the maximum speed would be increase/decrease. This would force the automatic car to alter its speed to hit the new limits, however the manual car would have no visual cue, other than the sign, of such change. If the user was over this speed limit, they will NOT receive a notification, as in real life no such thing would occur. Only at the end will they see that they were over the limit.



Figure 11: 30mph speed sign next to the trigger.

Every frame the current speed of the car, which an easily accessible variable in the unity package Assetstore.unity.com, 2018), is compared to the max speed of the track, in the section the car is in. If the vehicle is 2.0f over the speed limit, the time over the speed limit will start to accumulate. This is one of the pieces of data that will be presented to the user once the test is complete. Another piece is the average speed. For every frame that was calculated, a counter would be incremented. During these calculations, the current speed would keep being added onto the sum of all of the rest before it. At the end of the test, the sum of all of the speeds is divided by the counter, therefore returning the average speed value.

The next part of the course is the emergency stop. At the start of the test, a random float value is calculated between 5.0f and 30.0f. Once this value is calculated, the test begins, and the time of the test is being recorded. In the top left of the screen the user will see both speed, and the current time of their test, so we can just take this value and compare it to the random value calculated at the beginning. If the value is greater than or equal to the random value, then the stop signal shall be presented, and a new timer will be created.

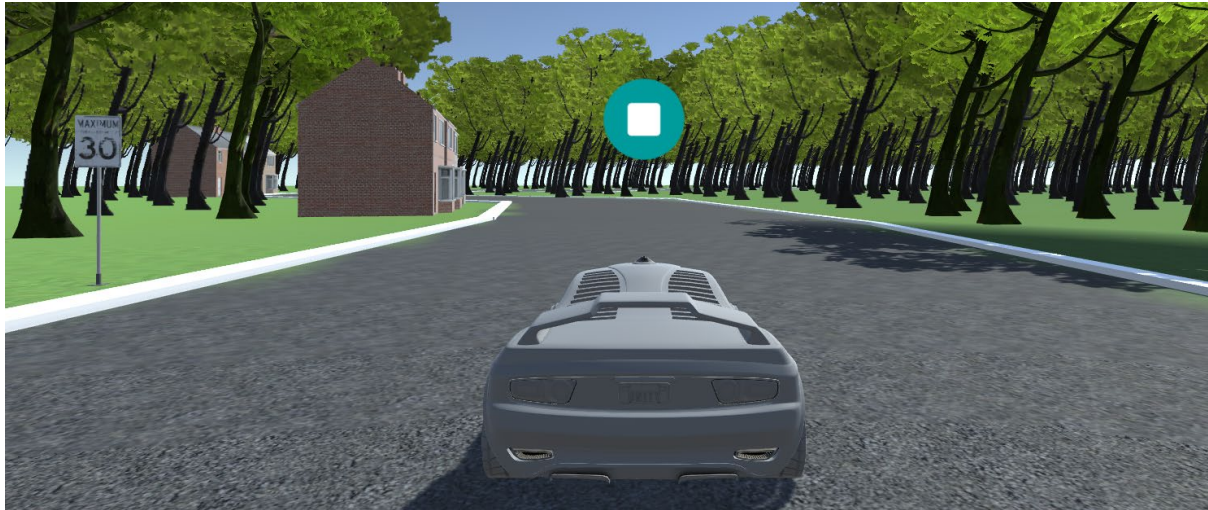


Figure 12: Emergency stop signal.

Once this timer has been started, a Boolean will be set to true, to stop the new timer from constantly being restarted, and now the programming is just constantly checking the current speed, waiting for the vehicle to come to a complete stop. Before any of this happens, once the timer has started, the program waits for the brake input. This is how the program shall get the reaction time data. After this, the timer shall continue to run, all the way till when the vehicle has stopped; this will then be the time to stop data. Once this happens, the emergency stop sign shall be removed from the screen and the time will stop and allow the use to carry on with the test. This piece of data should vary depending on how fast the user is driving. If they're going at a higher speed, or are even going over the speed limit, there will be a much greater stopping time. Both the reaction time and the time to stop data will be presented at the end, alongside the data from the rest of the test.

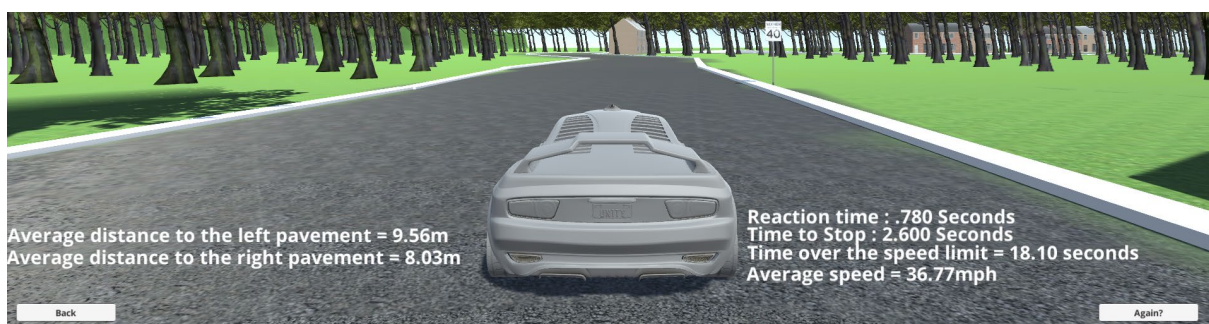


Figure 13: Post data presented onto the screen.

Reaction time

The track for the reaction time is very simple. I created it in the same way as the course test, using the unity terrain building, but this time I just made the road be a straight line. The road isn't endless, you can in fact see the end, but at no point will the player reach the end, unless they decided to not follow the simulation. They have the ability to ignore it, and also can turn left and right off of the track, as I didn't add the pavement, nor the ability for the vehicle to crash. The track is still however surrounded by the tree objects (Rakshi Games, 2016), and there are a few houses too, to add to the immersion (RIK4000, 2017).

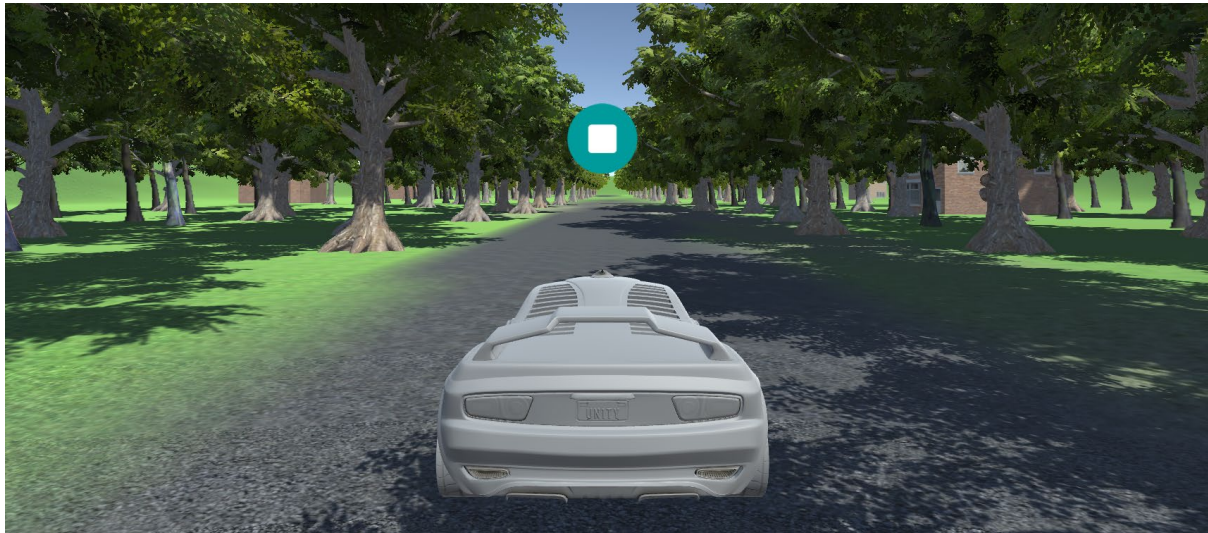


Figure 14: The emergency stop on the reaction time test.

The reaction time test here is identical to the one in the course test, other than the maximum and minimum values of the time of the emergency stop were adjusted to fit the track better, they values on this test are now 20.0f and 4.0f respectively. Once the user has made the car come to a complete stop, and the reaction time and the time to stop values are created, the test ends and those two values are presented onto the screen.

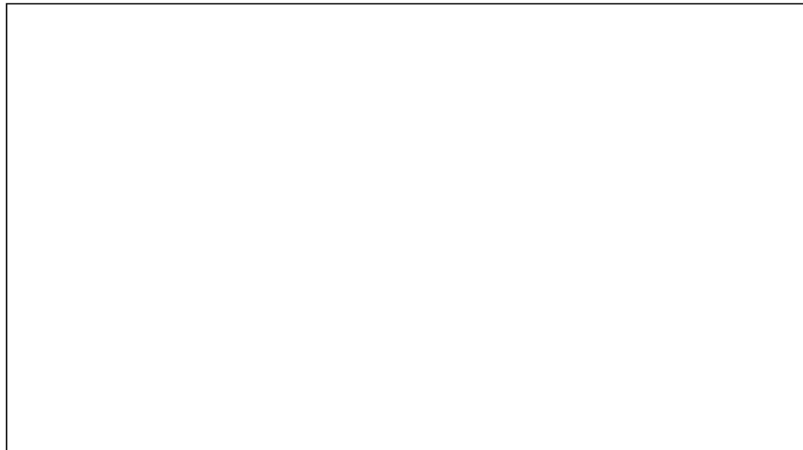
When it comes to the difference between the two vehicle types on this test, the only difference is that the automatic car keeps a constant speed of 30mph, compared to the car that the user has to control the speed. In the instructions written, the user is to see the speed sign, and they're to use that as a minimum speed. For this test, that minimum speed is 30mph. This is to prevent the user driving at a slower speed to increase their reactions and stopping time. However, when trying to maintain that speed, if it raises to high, then the car will have a much larger stopping time, and the forward force is much greater. There is also a fail safe on the car, to stop the user from breaking too early. If the user hits the break before the emergency stop sign is set active, then they're presented with a error message, and no test data is produced. Again, the buttons for either exiting or repeating the test shall be presented.

Testing

Plan

The testing plan is to get 5 participants to separately complete each test 5 times. They will be given basic instructions, from the game controls, and the reason for the tests/their participation. They will be told this is a simulation, and every aspect of the game must be taken as if it was a real-life situation. From road position, to speed, to the emergency stop. Once I have gained data from all 5 participants, I shall average out the results and I shall create a graph to plot their results. I will also be asking all of the participants one question and ask them to sign as consent to their participation. All of the testing data for the individuals shall be left confidential and placed in the supporting documents for this assignment. All participants to the following tests were ages 18-25.

Overall, what did you think of the simulation? Both positives and negatives. For example, this could be about the immersion, the controls, or the test results, etc.



Date :

Signed:

Figure 15: Question and candidate signature.

Automatic Course Test						
Test Number	Reaction Time	Time to Stop	Average Distance to Left	Average Distance to Right	Average Speed	Time Spent Over the Speed Limit
1						
2						
3						
4						
5						

Figure 16: Testing Table.

Reaction Time Test

Reaction Time Test

Test Number	Manual	Automatic
1	0.532	0.44
2	0.536	0.372
3	0.492	0.466
4	0.452	0.308
5	0.624	0.364

A graph to show the average reaction time for the automatic and manual car (Reaction time test)



Figure 17: The average reaction time of all candidates.

Time to stop

Test Number	Manual	Automatic
1	3.22	2.704
2	2.876	2.34
3	2.708	2.434
4	2.832	2.288
5	2.852	2.332

A graph to show the average time to stop for the automatic and manual car (Reaction time test)



Figure 18: The average time to stop of all candidates.

Course Test

Position Left		
Test Number	Manual	Automatic
1	9.326	9.152
2	9.296	9.422
3	9.26	9.322
4	9.514	9.338
5	9.182	9.074

Position Right		
Test Number	Manual	Automatic
1	8.188	8.308
2	8.162	8.098
3	8.066	8.018
4	7.886	8.126
5	8.184	8.298

A graph to show the average position to the left



A graph to show the average position to the right



Figure 19: The average position of all candidates.

Average Speed		
Test Number	Manual	Automatic
1	32.836	33.314
2	32.236	33.798
3	31.848	32.534
4	30.554	31.94
5	31.744	32.828

A graph to show the average speed



Figure 20: The average speed of all candidates.

Average Time Over		
Test Number	Manual	Automatic
1	8.764	0
2	5.992	0
3	4.876	0
4	2.092	0
5	2.432	0

A graph to show the average time over the speed limit

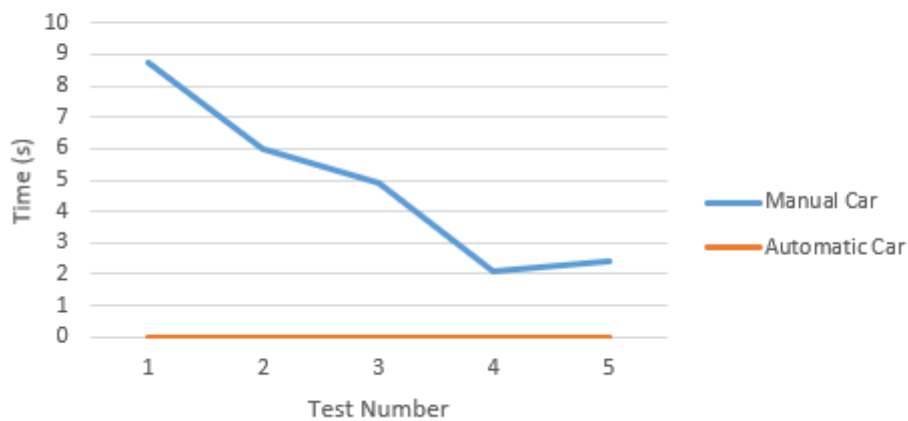


Figure 21: The average time over the speed limit of all candidates

Reaction Time (Course)

Test Number	Manual	Automatic
1	0.708	3.624
2	1.08	0.652
3	0.72	0.732
4	0.6	0.58
5	0.684	0.612

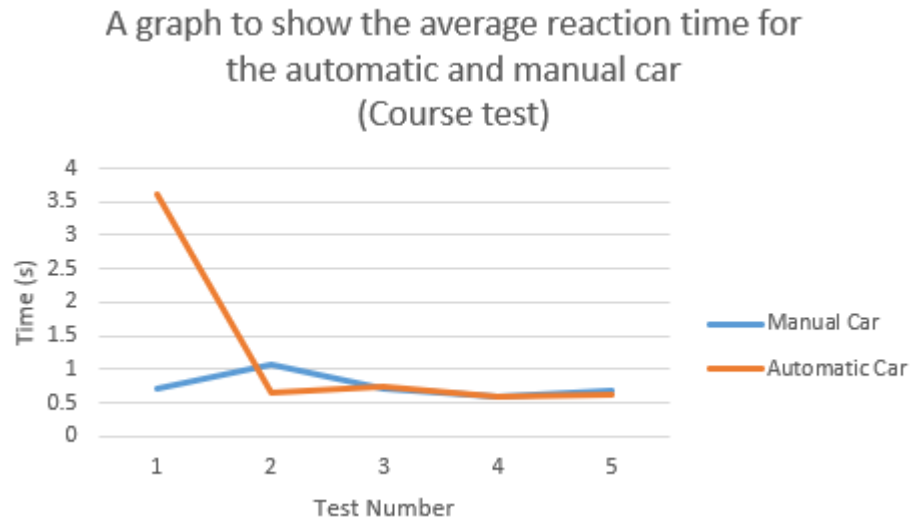


Figure 22: The average reaction time of all candidates on the course.

Time to Stop (Course)

Test Number	Manual	Automatic
1	2.9	6.286
2	3.276	3.044
3	2.82	3.012
4	3.736	2.56
5	2.618	2.592

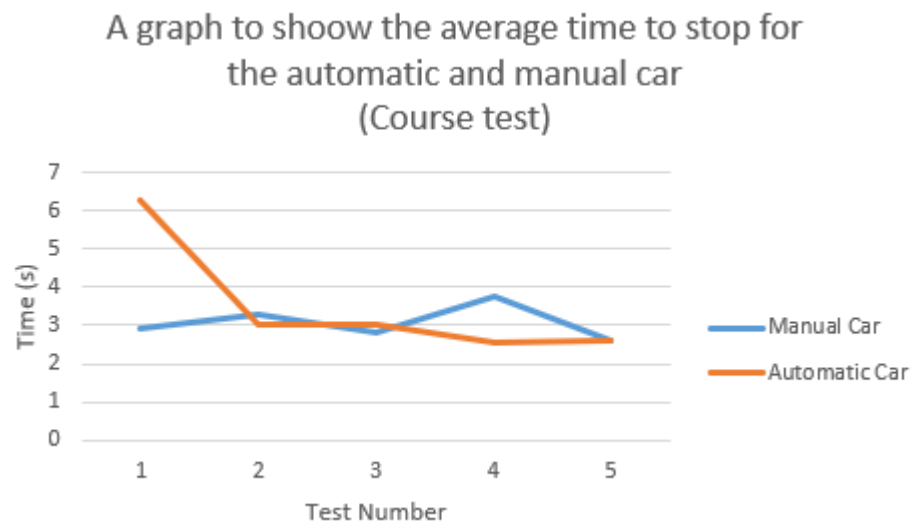


Figure 23: The average time to stop of all candidates on the course.

Conclusion

The premise of this entire thesis was to determine if there were any effects on reaction time and driving ability when the automation of a vehicle was changed. Some of the results were as expected, and others were slightly surprising, the reasoning is what shall be discussed now.

Firstly, I thought that the reaction time of the user would be different on both the reaction time test, and the course test, and the results do backup that. On the reaction time test, the user only has to focus on travelling in a straight line, and on the manual mode, keeping the correct speed. Compared to on the course, where the user also has to navigate around the track and check on the environment for speed signs. On average, the user was 0.24 seconds slower to react on the course test, if we remove the outlier in the results. If we keep that value in, then the users are on average 0.54 seconds slower. Other reasons in to which the user's performance may be affected on the course is the random aspect of the emergency stop. There is a chance that the user would have got that stop during a corner, and if they were on the automatic version of the vehicle, they could have been fighting with the car with the breaking and turning at the same time. Participants during the questionnaire outlines two main issues they faced. They found that the sensors on the vehicle were way too reactive. They would force the user to turn when the distance from the pavement was too great, and it force it would apply against the car was way too hard and would often make the vehicle hard to control after. Another issue that they mentioned was the position of the current speed of the vehicle. During the testing, the speed was located in the top left of the screen, which candidates indicated was impractical, unrealistic, and was too much of a distraction. From this feedback, I have changed the position of the speed to be just above the car.

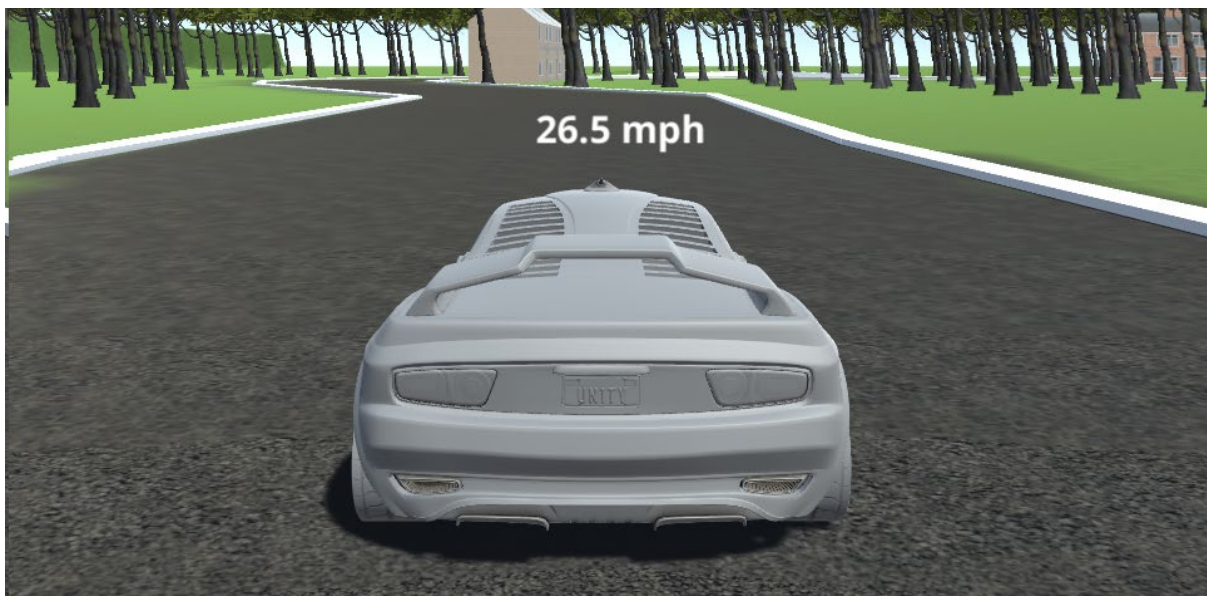


Figure 24: The speed of the car moved to above the car.

Another part of the test was to see if there was any difference of reaction time between the manual and the automatic version of the car, and here I think I got some interesting results. I expected that the manual car reaction time would be slightly higher than the reaction time for the automatic car. My test results show that this was correct for the reaction time test, but however, the reaction time on the course was the same, regardless of the car that was selected. On the reaction time test, on average the user reacted 0.14 seconds slower using the manual car, compared to the course test where the user reacted less than 0.1 second slower using the manual car. During the testing, there were some outliers, and if we take those into account, then perhaps the graph could be deemed as showing that the automatic car did have a quicker reaction time, but even with this, the gap between the two averages was much smaller compared to the reaction time test alone.

The final part of the reaction times is the time to stop times. On both the reaction time test and the course test, there was a noticeably higher stop time with the manual cars, and also a higher time with the course test. This is down to both the randomness of the emergency stop, and also the human participants. The course has different speed sections, so there is a chance that the zone the participants had to stop in could have been 40 or 50 zone, compared to the reaction time test, where the entire road was a 30 zone. If the vehicle is going at a faster speed, it will therefore take more time to come to a stop. On top of that, there is the human factor. There was a chance that the participant was also going even faster than the speed limit, and much higher than the fixed 30 on the reaction time test. This again would have caused the car to slow down slower. The final factor I noticed from observing, was that the participants sometimes struggled to determine if the car had come to a stop, and they would start to drive again, even though the test was yet to finish. Meaning that the participant would have to stop again, increasing the time it took.

Another part of the tests was the average speed of the car during the course, as well as the time they spent over the speed limit. I expected the participants would have a much higher average speed, and a high time spent over the speed limit. I thought this would happen because of several reasons. This includes the fact that it was a simulation. No matter how immersive I can make the experience, or how accurate I can make it to real life, it still isn't real life. Therefore, the participants will not use the same instincts they do on a real road, especially when it comes to following speed laws, as there is no real punishment for failing to do so on the test. When it came to reviewing the test results, I was shocked to see that the average speed on the manual car was actually lower than what it was for the automatic. The automatic car hits the speed limit, and it stays there constantly till the speed is changed, and then it will either speed up or down to hit the new limit. The time over the speed limit here is always 0, as the car adjusts very fast. As the automatic car doesn't have to worry about going above or below the speed, it has a high average speed, as it is always going to the law, unlike in the manual car, where the participant would constantly be dropping above and below the limit, trying to control the vehicle. When the participants are aware, they're being monitored, they were most likely going to have a lower average speed to try and stay under, as it is lower risk. But the time to react to the new speed zones made it so that the time over the speed limit was still high.

The final part of the test was the average position of the car. I predicted that the test results for this test would be varied for the manual car, as the player would have to concentrate on more than one thing, but I feel like I got some interesting results. The test results showed that there was very little effect on the car position when the automation on the vehicle was changed, however on both, participants preferred to drive closer to the right-hand side of the road.

Firstly, I believe that the participants drove closer to the right-hand side of the road, as the track was running clockwise, meaning they would be making more right than left turns, forcing them more to that side. If the track was run backwards, I predict that the values would flip, and the participants would prefer the left-hand side. Secondly, I think that the values were similar on both the manual and the automatic car for several reasons. One being that the data is an average of the entire track time. There might have been points during the test that the user got closer to the pavement on one version of the car, but the test wouldn't show it. Nor would it show any difference if the participant wiggled the car down the road, as it would average out to driving down the middle, or as we saw, preferring the right-hand side. Data about the furthest and closest point to each pavement might have also been a nice piece of data to record. Another reason to why there might have been no difference was due to the user not being too distracted to not be able to focus on the road. As the main reason for the course is to be able to navigate around it, most of the user's focus was most likely aimed at their road position. A final point on this would also be that data is not presented if the user crashed. So, all of the data averages that were calculated, were from when the user made it to the end of the track. These tests may have been perfect runs for them, compared to tests where they were closer to the pavement, and therefore maybe crashed. The crashed data, and failed data is also in the supporting documents for this thesis.

To answer the thesis topic of, "An investigation into the effect of automation in vehicles on user perception and driving ability", I would say that no, no my test results show that automation in vehicles actually improves users driving ability and perception. When the automation on the vehicles was turned on, the participants of this study showed improved perception and driving ability, and in no way did it impair their abilities. These test results were based off of a simulation however, so test results may differ if the same tests were done but on real life scenarios. There would be more at stake for the user, and it would be realistic. It would also be better as it would allow the data to be tracked over a much longer amount of time, as 5 tests per participant is not an accurate representation of a real-life scenario. However, despite these implications, I do think that my test results do have significance, and they do give an insight to how humans are reacting to automation in vehicles. This is a significant topic that must keep being measure and analysed, as over years of use with automation, the topic discussed in this thesis may actually start to become impaired, which could lead to issues involving safety.

Reflection

Overall, I am very happy with how my project worked out, and I am happy with the work that I produced. I planned much of the work very early on, around Christmas time, and the designs were drawn, and I kept to my schedule that was drawn up in the Gantt Chart. I was happy with these designs, and as I am used to the Unity software, I had a good idea of how I would be able to implement a solution for the problem.

Something that went badly was that I had a coding problem when it came to the reaction time. Sometimes when recording the reaction time, the value returned would be NULL. I spent several hours trying to find a solution to this, but I couldn't figure it out. I tried to fix this issue quite late on, and I needed to start my testing and focus on my Thesis. The value for reaction time works the majority of the time, so instead of no value coming up, I decided to add in a small fail safe, and that was that it just checked to see if the float value for reaction time was NULL, if so, it would output an error message.

Something else that went bad was some of the testing. Firstly, I feel like I left this stage quite late, and this forced some of it to be rushed. I wanted to gain a large range of data, so the averages were better. Another issue I had with this was that after the first couple of participants I realised I hadn't recorded the time it took to complete the course test. I feel like this would have been a very good, significant piece of data to keep, so I could compare it to the average speed, and see if it changed my thesis answer. I decided to test all four tests, five times with each participant, and this took a lot more time than expected. Firstly, it took between 20-30 mins per participant to gain the data, then I had to write it up onto my excel spreadsheet, before calculating the results. This meant that I couldn't add more participants into the testing, as there simply wasn't enough time to near the end. This is also linked to the above error with the reaction time. Sometimes the user would have to repeat tests due to this error, which could add five more minutes onto each participant.

Contrary to those adjustments, I did actually like the test data I acquired. It allowed me to map out the results very well using line graphs, making it much easier to see the data. The supporting documents are filled with the individual candidates scores, and also the averages of all of the candidates together, and I like how it is presented and kept.

I am glad I decided to use premade assets from the Unity store, as it allowed me to save a lot of time. The main point of this thesis was to gain data to back the idea of automation in vehicles. I did not want to allocate some of that time into researching into making models for cars, trees and houses, as this would be a new area for me. Also related to this issue of my own knowledge gap in 3D modelling, I am not used to shadows and lighting effects in Unity. I noticed during testing of the game that whilst the car was driving down the road, the trees shadows would load in as you drove close to them. I was unsure of what could have caused this issue due to the viewing distance being fine, but it was also mentioned by a couple of the participants as they said it was quite distracting and caused an unfair testing environment.

If I was to repeat this thesis again, I would have given myself much more time to perfect the simulation and given myself more time to do more tests. The more test data I could have got, the more accurate and reliable my results could have been to a real-life implementation. I would have also liked to put more time into the end product of the simulation as some parts of it made it quite difficult to use. For example, the sensors on the vehicle did make it quite

hard to handle, and therefore it did impair the immersion and make it a lot harder to gain data. Another thing that was apart of my project plan, but never got implemented was the custom map creator. This did not get added due to time constraints, but I feel like it would have made the simulation much more industry ready, as anyone would be able to make a test with their specific needs in mind. In addition to that, I would have spent much greater amount of time researching into other researcher, and how their work was similar to mine, and how I could have used information and data they have already studied.

References

Answers.unity.com. (2014). *Raycast rotating with object rotating - Unity Answers*. [online] Available at: <https://answers.unity.com/questions/619181/raycast-rotating-with-object-rotating.html> [Accessed 8 Apr. 2019].

Technologies, Unity. (2019). *Unity - Scripting API: RaycastHit.distance*. [online] Docs.unity3d.com. Available at: <https://docs.unity3d.com/ScriptReference/RaycastHit-distance.html> [Accessed 8 Apr. 2019].

Assetstore.unity.com. (2016). *Realistic Tree 9 [Rainbow Tree] - Asset Store*. [online] Available at: <https://assetstore.unity.com/packages/3d/vegetation/trees/realistic-tree-9-rainbow-tree-54622> [Accessed 8 Apr. 2019].

Sarper-Soher. (2009). *Rotate the camera around the object..* [online] Available at: <https://forum.unity.com/threads/rotate-the-camera-around-the-object.47353/> [Accessed 8 Apr. 2019].

Assetstore.unity.com. (2018). *Standard Assets - Asset Store*. [online] Available at: <https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-32351> [Accessed 8 Apr. 2019].

Assetstore.unity.com. (2017). *UK Terraced Houses Pack FREE - Asset Store*. [online] Available at: <https://assetstore.unity.com/packages/3d/environments/urban/uk-terraced-houses-pack-free-63481> [Accessed 8 Apr. 2019].

Assetstore.unity.com. (2018). *North American Speed Signs - FREE - Asset Store*. [online] Available at: <https://assetstore.unity.com/packages/3d/props/north-american-speed-signs-free-117484> [Accessed 8 Apr. 2019].

Georgiou, T. and Demiris, Y. (2016). *Personalised Track Design in Car Racing Games*. [online] Available at: <https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/stamp/stamp.jsp?tp=&arnumber=7860435> [Accessed 8 Apr. 2019].

Georgiou, T. and Demiris, Y. (2015). *Predicting car states through learned models of vehicle dynamics and user behaviours*. [online] Available at: <https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/document/7225852> [Accessed 8 Apr. 2019].

Kucukyilmaz, A., Sezgin, T. and Basdogan, C. (2012). *Intention Recognition for Dynamic Role Exchange in Haptic Collaboration*. [online] Available at: <https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/document/6203504> [Accessed 8 Apr. 2019].

Carlson, T. and Demiris, Y. (2008). *Human-Wheelchair Collaboration Through Prediction of Intention and Adaptive Assistance*. [online] Available at: <https://infoscience.epfl.ch/record/150450/files/carlson-icra-2008.pdf> [Accessed 8 Apr. 2019].

Ezeh, C., Trautman, P., Holloway, C. and Carlson, T. (2017). *Comparing shared control approaches for alternative interfaces: a wheelchair simulator experiment*. [online] Ieeexplore-ieee-org.proxy.library.lincoln.ac.uk. Available at: <https://ieeexplore-ieee-org.proxy.library.lincoln.ac.uk/stamp/stamp.jsp?tp=&arnumber=8122584&tag=1> [Accessed 8 Apr. 2019].

Zolotas, M., Demiris, Y. and Elsdon, J. (2018). *Head-Mounted Augmented Reality for Explainable Robotic Wheelchair Assistance*. [online] Available at: https://www.researchgate.net/publication/327720611_Head-Mounted_Augmented_Reality_for_Explainable_Robotic_Wheelchair_Assistance [Accessed 8 Apr. 2019].

Tesla.com. (2018). *Introducing Navigate on Autopilot*. [online] Available at: <https://www.tesla.com/blog/introducing-navigate-autopilot> [Accessed 11 Apr. 2019].