SmartMob-VR: Virtual Reality platform for the development and validation of intelligent mobility systems

André Ponce Álvares Vieira
andre.pa.vieira@hotmail.com

Instituto Superior Técnico, Lisboa, Portugal

December 2019

Abstract

SmartMob is a project from a IST's Taguspark campus's course. This project allows students to study network communication with cars that can be controlled with scripts on a Raspberry pi. These are physical cars with batteries that are drained very quickly. The cars are used with a limited amount of physical carpet-like roads to model cities. SmartMobVR puts this project in Virtual Reality, motivating students and mitigating some problems. SmartMob VR has two main components: modelling and visualisation. Modelling is the program that allows the students to develop a city. There is a two dimensional grid where they can place roads, buildings, trees, statues, traffic lights and lamps to model the desired city to then export into an XML file. The visualisation, then, imports the XML file and builds the previously implemented city in three dimensions that can be seen in Virtual Reality. The real cars can still communicate with each other and communicate with the visualisation so the user can see what movement the cars are doing. Users can navigate through the city and select the cars they want to. Having a dynamic world and dynamic objects we study selection techniques to see which one works best for our project.

Keywords: Virtual Reality, SmartMob, Selection Techniques, Modelling, Dynamic

1. Introduction

SmartMob is a project developed in academic environment that allows students to study and develop intelligent mobility systems with infra-structure elements and vehicles that work in a cooperative and connected way. The project consists in teams of students proposing solutions for challenges such as road safety, traffic management or entertainment. These solutions, developed in practical classes, will be later applied in the real world. The course unit that provided this project has a set of equipment that supports learning activities for its students. Among that equipment we highlight the miniature vehicles, 1:24 scale, which have been transformed into intelligent objects, in order to serve as a basis for the test of intelligent mobility systems, in which it is necessary to study their movement and interaction with others in a small town.

By observing and analysing the current testing environment, we are able to identify common problems that may arise, such as interference in communication channels, propagation of problems and access to the available environment, as well as the difficulty to control and coordination of the vehicles' movement. Considering that the multiple groups of students are sharing the same physical resources, the class' schedule is limited and even if there are multiple students using different equipment at the same time, there is the possibility of interference on the communication. Additionally, as previously mentioned, it is essential that these test can be carried out on a larger scale, with conditions more compatible and more similar to the ones taking placing in the real world. With this project we intend to, not only, mitigate usability issues and limitations described previously, but also to make the usage and study of smart cars and other related components (such as the roads they circulate on and the buildings around them) easier and more appealing.

Virtual Reality opens great technological doors, allowing users to immerse themselves in new worlds and have a level of interaction with virtual environments never before achieved. VR creates new opportunities for broader and more complex tests and experiments, not only allowing greater environment control and new interaction possibilities, but also presenting less limitations, like the ones imposed in the real world.

The transition of the current SmartMob to a virtual environment would mitigate testing limits and issues as the ones referenced before, since the process of creating and experimenting can be per-

formed repetitively and unlimitedly. In this sense, Computer Science allows us to have a degree of freedom of manipulation which can help us in situations such as this. Real environments can be replicated and recreated very closely to reality, allowing us to perform both realistic tests (where the environment factors are very little, or not at all, changed) and controlled tests (where certain environment aspects can be manipulated, changed or even removed). Controlled test can be especially helpful since some factors that exist in the real world may compromise and/or limit the tests in question. Space, energy and other physical resources are examples of elements that can be manipulated when transitioning from the real world to the virtual world, making it able for users to simulate scenarios or behaviours multiple times without having to worry about, for instance, financial spending or environmental impact. Moreover, virtual reality offers further ways to visualise data when compared to other simulators, while also simultaneously allowing the collection of data.

The core idea is to implement and test a tool that can offer the ability to create a virtual city, whilst providing enough freedom and flexibility that the students are able to replicate real models into virtual ones. Furthermore, the usage of technology will make the process of modelling more appealing to them, and provide freedom to work outside their classrooms or laboratories, where the current equipment is stored.

Virtual Reality presents the challenge of selecting objects, since multiple techniques are available but without clear knowledge of which is best(with a steep learning curve - meaning that the user quickly grasps how to perform activities using the technique and demonstrates a quick rate of skill acquirement when compared with the amount of experience it has using said technique). This project resolves around a dynamic environment with dynamic objects because users can move around freely throughout the different scenes with different angles to the same cities and the objects that will be selected are moving cars with velocities and routes through the city. Therefore, part of this project will involve the analysis of different selection techniques in order to choose the most adequate, efficient and effective one for the target environment of this study, taking into account the type of activities students will perform using SmartMob VR. This analysis will require testing with users to understand which technique is better suited to the type of activities students will perform.

2. Background

In this study the focus will be divided into two greater topics: knowing how virtual cities are built and work; user's navigation in the virtual world and selection techniques, more specifically which techniques were studied and which was the one that fit best in the selection of dynamic objects in a dynamic virtual world.

2.1. Virtual Cities

With the help of computers, where it's possible to accomplish huge calculations and complex simulations, the study of traffic, that is, the interaction between road agents and their components, has been deepened and diversified. An important factor to consider is the interaction between the cars studied and the other agents that take action in the same environment, such as the roads, traffic lights, traffic signs and other cars that circulate around the city. In this section we will look at a few works that can help you understand what to keep in mind when building traffic simulators.

Virtual reality is a tool that has been more and more used over the years, not only because it offers new ways of visualising the scene, but also can help in interpreting data in a new, intuitive and unique way.

2.1.1 Virtual Cities

There are several track design tools with different goals, ways of operating and user interfaces, spread across the Internet. Scalextric Track Designer[1], for instance, is a tool with a very simple design. The top of the program screen layout contains several lists of objects that are available for use, such as straight roads, curves, intersections, physical boundaries like barriers, and even imaginary boundary lines that can help you build roads in accordance with certain limitations (like the space available for building). When a component in the scene is selected, a user can simply click on another component in the top panel, and this new clicked part will be inserted immediately and automatically connected to the unit that was selected previously. Additionally, the new added component will be the new selected part to which the next component will be attached to, thus creating a track with just a few clicks. In case the user desires to edit an already constructed track, for instance, change its shape, he can simply select the road piece he wishes to change using the left mouse button, and holding it down as he drags the piece around the scene and into the new desired position. When the mouse button is release, the piece is set to its new place. If, as the object is placed, there are other parts adjacent, it will automatically attach itself to them in order to maintain the road map's consistency. Even though

[1]Track Design Software, slotcarwiki. https://slotcarwiki.org/tiki-index.php?page=Track+Design+Software

this feature is, for the most part, a good addition that makes the modelling easier, there can be situations where you do not want the pieces to attach, such as when you create two parallel roads next to each other - in theses cases, this involuntary fit can make editing a little frustrating and inefficient. Concerning the position of the pieces, the user can also press the space bar to rotate certain components to the right or left.

2.2. Discussion

Building a city in virtual reality can be an interesting and fun activity, but a drawback of doing this is that users can take a few hours to complete the modelling. Modelling in a VR environment with precision and attention to detail might be harder to perform when compared with a two dimensional application. In the latter case, users can create and edit an environment in any computer without requiring any special equipment, but also without needing to worry some side effects of virtual reality such as motion sickness, spatial disorientation, vertigo or headaches. By taking this into account, we decided that to build the city we want something similar to the Scalextric Track Designer, however, excluding its the automatic connection to adjacent roads or the three-dimensional view of the modelling. We want to make sure users can add and remove road pieces wherever they want on the grid and not just adjacent to other road blocks.

2.3. Virtual Reality

Virtual Reality is a technology that exists for a while now but it has been getting more popularity and accessibility for more people. With this and the progress of technology there are more ways to utilise it. There isn't a global best way to navigate through different virtual worlds and not a global way to select different type of objects in those environments. This sub chapter talks about possible ways to navigate and select objects for our project.

2.3.1 Navigation

Navigating in Virtual reality can also be done by moving the user progressively as if he was flying. There are several techniques to do this, but the hand technique looks to be the most efficient[3]. This study has three different techniques, Elevator+Steering technique, gaze technique and hand technique. The most efficient technique from this study was the hand technique. This technique resolves around the user using his hand to define which direction to move to. If the hand is pointing upwards, the direction will follow that way. This way users can look around while pointing to the desired destination.

2.3.2 Selection Techniques

There are a lot of studies and papers regarding selecting static objects in virtual reality. All of which try to fix jittering or occlusion problems. In our scenario we want to select dynamic objects in a dynamic world. This type of techniques are still something not studied in much depth.

A simple ray being shot from the user's finger may be a straightforward approach to selecting objects, but like most techniques in some cases the hand and/or device's jitter may compromise the selection of the desired object. Another big issue is occlusion, a simple ray can't select an object that is behind another or that is covered with multiple other objects from the scene, since the ray will simply select the closest object to the ray.

Continuous rays make for a great approach regarding dynamic and/or objects that are far away. Smart Ray[2] allows the user to have a continuous selection on objects and selecting the pretended one without much effort. This technique uses an algorithm based on target weights to determine which target should be selected when multiple targets are intersected. Target weights are continuously updated based on their proximity to the ray cursor. The closer the ray is to the center of the target, the larger the weight increase will be. When starting to select a target, a ray is shot and a green highlight will appear in every object intersected. The intersected target with the highest weight is highlighted red, indicating that it can be selected by clicking a button. When the ray intersects multiple targets, the user can re position the ray so that its new position still intersects the intended target. Even if multiple targets are selected by the new ray position, the intended target will have the highest weight, as its weight has been continuously increasing.

There are regular cone techniques where all objects inside the cone are selected. Other iterations of the regular cone is a Flashlight type of technique where the user still has a cone but there is a refinement. This refinement consists of selecting the object closest to the center of the cone and removing all other potential objects.

All the objects within Shadow Cone[4] are selected when the user presses the button, by holding it, and as the user moves their hand only objects that are always within the cone are selected when a button is released. When the user presses the button, the potential selection set is initialised with all objects that are within a target angle of the ray from the hand. On each frame any object that is no longer within the target angle is removed and it's deselected from the previous group. When the user releases the button the objects remaining in the potential selection set are then selected.

To ensure a seamless and effortless transition from ray casting selection behaviour, it is taken a simple ray-casting as a basis for the scoring metric. To get this, it is assigned the highest score of 1 if pointing is most accurate. That is, the ray is being pointed through the object's center point. It is also assigned the lowest score of 0 if pointing is least accurate, when the object's center point lies on the bounding surface of the conic volume. The scoring contribution is accumulated over time. Each object maintains its score over several time frames. That is, the score is not reset at every frame, but kept during a sequence of frames. So everytime the object is selected its score increases and when it stops, there is a decay. In the end of the selection ,the object with the highest score gets selected[1].

2.3.3 Discussion

For navigation purposes it will be used the Hand technique studied in [3] because it looks the most familiar and had great results in their testing.

Each technique is unique in its own way.

There's different selection volumes for each technique: Point, Ray, Bubble and Cone. Point techniques are the ones where there is an analogy on the user grabbing the objects. If we imagine a hand picking an object with the thumb and index finger, doing a pinch motion, we can see a point in space. If this point touches or intersects an object, this object is selected. The point created by the user doesn't have to be in the arms' reach of the user, for example in the Go-go hand technique, the user's hand is extensible and they can control it's reach reaching distances further than the user's hand, but still using the point analogy to pick objects up.

When it comes to types it is important to separate single and multiple techniques. There are some techniques that are worth mentioning that they can select multiple objects on their final selection. This means the user may want to select one object but if other object is in the same conditions as the first one it will also be selected. This is what it means to be able to select multiple objects. The final result may be more than one.

Then we have different refinement methods. By refinement we mean how the algorithm chooses the object over another if there are more than one being intersected by the selection volume. There are continuous refinement and discrete refinement. The main difference between the two is the amount of steps or inputs from the user for the refinement to be completed. Continuous refinement resolves around constantly excluding objects from the desired one. For example, Smart ray is a technique that focuses on adding weight to objects that are selected, the refinement is always being made since the objects either keep getting more weight, then being intersected by the ray, or have their weights stable. Discrete refinement requires the user to give more inputs when trying to select the object. By this we mean that there are more than one step to select an object. It is an iterative process. SQUAD, for example, has a bubble at the end of the ray, which has the possibility of selecting more than one object from the first user's input. If this happens a new screen will pop up and show all the selected objects in a new User Interface, because the user has to point again and give another input, this refinement is considerable discrete.

Lastly, we wanted to show the amount of studies made, from the different techniques, in dynamic environments since our work will work around dynamic objects in a dynamic world.

The studied techniques will be: SmartRay, Flashlight, ShadowCone and IntenSelect.

Selection Techniques					
Name	Selection Volume	Type	Refinement		Dynamic tested
			Cont.	Disc.	
Ray	Ray	S			
SR	Ray	S	X		
F	Cone	S			
SC	Cone	M	X		
IS	Cone	S	X		

3. Implementation

In the process of creating this solution we wanted to develop tools that would allow students to test and run simulations in a more comfortable, affordable and appealing way.

The solution is divided into two distinct parts: The modelling of the city and the visualisation of the system containing the city itself, the simulation of the traffic, interaction of cars with the remaining road elements and the usage of selection techniques.

Modelling is the creation and manipulation of the city, which will begin as an empty project in a computer program. The city modelling software uses a mouse and keyboard to provide the modelling inputs and the process is viewed by a monitor, thus being the system's output a two-dimensional view of the city.

The second part of the system has the objective of giving a visualisation of the city that was created in the previous step and its components, in three dimensions, in virtual reality environment. The simulation's visualisation is in direct contact with the simulation and the users as this is where the information about the cars, like their location, and car controls come from (Figure 1).

The document created by the modelling part of the solution can, and while testing, is the objective to, be similar to the miniature city of the room with

the equipment and cars communicating with each other.

The proposal is divided into two parts , and in which part it is going to be discussed the architecture and the Interface.

3.1. Architecture

In this section we will talk about what was done in the process of creating the solution and how it works. Going from overall functionality to specific details in features, both for modelling and visualisation part of the application.

3.1.1 Modelling

For modelling the city, the use of mouse and keyboard in a computer was chosen to facilitate the activity in a classroom environment, where several people can model at the same time, which also gives the possibility to model the environments outside of the university environment, and thus out of school schedules, making it more versatile for students to model it anywhere anytime. As the modelling process can be somewhat lengthy, this approach was also chosen because the glasses can cause nausea if worn over a long period of time, which depends on person to person, leaving then only the visualisation of the city in virtual reality.

The modelling of the city is done in a executable program built in Unity 3D so it can run in any Windows computer with a recent version of the operating system.

The places where the objects can be placed are delimited by a white grid on a blue background since it makes it easier for the user, for processing and generating the output file for the modelling itself. The grid can be moved around freely while holding the left click on the mouse and moving the mouse's cursor around, the grid moves with a one on one relation to the mouse so it seems like the the user is grabbing it with it's hand making it more familiar to move around.

Users can clean the whole grid, removing all the objects in it. Open previously exported XML files by the modelling program, after clicking the "Open" button a new window is brought up and asks the user to select a XML file from his or her computer to import it to the current grid. Finally, files can be written with the "Write" button, this pops a window so the user can select the destination and name of the XML file that is going to be created with the objects in the current grid.

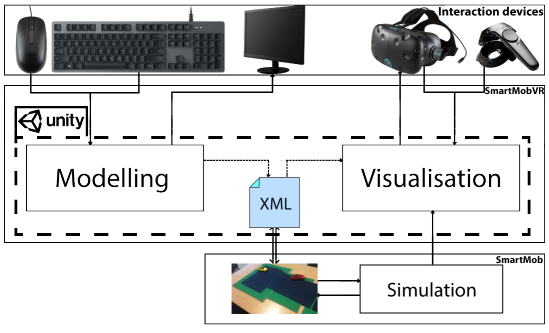


Figure 1: Solution architecture

Because it would be hard to model, since some objects might occlude others or the scene may be too confusing at times, with a camera with the same direction towards the grid, it was implemented the option to rotate the camera. There are two buttons that allow this. Each button makes the camera turn either ninety degrees or negative ninety degrees, depending on the button, with the center of the screen as the middle stationary point.

Zooming in and out is also a feature in this program. Sliding the mouse's wheel upwards will make the camera move closer to the grid, and doing the opposite will move it further.

The modelling will be done by first selecting the desired object, on the list on the right part of the screen, that the user wants to put on the grid. This is done by clicking on the object with the mouse's left click over the object on the list on the right, this will make it have a small highlight so the user knows which object is selected at the certain time.

After the object is highlighted the user can now click on the desired grid and left click on it. There is another highlight feedback on the grid itself so the user is sure where he or she will place the selected object. When the left click on the grid is done, the object that was previously selected will now be placed in the desired position. The user may place as many as he or she desires in other grid cells until he selects the same object on the list on the right to deselect it.

The list of objects include Roads, houses, trees, statues, street lamps and traffic lights.

Each object can be right clicked once the object is on the grid and new options menu will appear with the objects properties. The main property is different from object to object and we will go over them briefly one by one but all of them have a "Del" button to delete the object if necessary.

At the end of the modelling process, the final result will be a blue print created and manipulated by the users that can then be exported in to an XML file. These files contain the modelled city information, such as the object ID's, properties, for example, rotation of the roads, the buildings' number of floors, the amount of afforestation that will be used to place objects, and position. These files can then be re-opened on the same program or opened in the Virtual Reality SmartMobVR simulation program.

3.1.2 Visualisation

The simulation and visualisation of the city is done with Unity 3D as well, where the city model, the implementation of the cars and the selection techniques will all join in one environment.

The city model will be done by importing a XML file generated in the previous step, when modelling the city. The program checks object by object in the file, instantiating every object according to its ID and position, then edits its properties to correspond to the given ones. Depending on the ID given, the object will be rotated, added more floors or even number of instances per position.

The cars' movement was controlled either by a scripting, coded in Unity or by the SmartMob simulation. The script implementation of the cars started with objects representing cars, taken from the SimpleTown asset from the Unity Asset Store, in unity that were coded in C#. These scripts were used in experiments because they had to have the exact same behaviour every time the program was played. All the cars would have the same path for each scenario.

The simulation will be specified in depth in the next subsection.

The selection techniques are chosen before the simulation starts to run, in the unity editing tool. Each technique produces a CSV file that contains information about the selections done.

Within the visualisation the user will see the city created but this time in three dimensions with the help of the Oculus rift glasses. Besides the HMD, users will also have two controllers with them, one for each hand. Each controller has a unique purpose. The left is used for moving the camera around the virtual scene. The user is able to see his or her hands inside the virtual world. The direction of where the player in the game will be heading is obtained while pointing with his or her index finger with the controller. To move around, in this virtual world, and according to the index finger's direction, the user can simply move the joystick in any direction. The right controller is used to activate the current selection technique. Like previously stated, the technique is selected before hand when the program is about to be played. Once it starts the user can point with their right hand and click on the back button of the controller and start the process of selecting an object.

3.1.3 Simulation

With the integration of SmartMobVR, the cars, from the laboratory in which the students are performing tests, can communicate with the cars in SmartMobVR. When in the same ad hoc network, the physical cars send CAM type of messages through UDP to the VR application. These messages sent by the cars have two parameters that are used to locate the car: its latitude and longitude. If the cars are used outside and are equipped with GPS they can send the coordinates used by the device, otherwise, an estimation is used instead. The velocity is determined beforehand, according to its battery life and the script controlling its wheels. With the velocity, being distance over time, the script uses the current and initial time to calculate where the car can be positioned at the current time. The CAM message is sent with these parameters calculated.

The visualisation part of SmartMobVR is prepared to receive UDP communication. Every span of time, for example, a third of a second, a new update on the car's position is sent to SmartMobVR so the respective car (the information on which car to update is taken from the CAM message's parameter "id") would move towards that position. The desired car starts at a given position, this information is gathered from the first message received. From the moment the position's queue has 4 or more pairs the car starts to calculate where to move next and the velocity.

The CAM messages also include a variable that represents the time on which the corresponding message was sent. With the time and pair of coordinates for each message, the velocity and direction is calculated and applied to the car. After the car reaches the second position, corresponding to the desired position at the given time, the first position pair is discarded from the queue and the same algorithm applies with the next position.

3.2. User Interface

This section will cover the visualisation aspect of the application that will be seen by the users.

3.2.1 Modelling

The application that allows users to model the city has a very intuitive interface: starting from the right side, the user can find the various types of objects that can be placed in the city.

The user starts with the screen in Figure 2. There is a big blue background with a white grid on top of it where objects can be placed. The grid's cells are where the objects can be placed, making it easier to distinguish each objects possible position.

On the top left of the monitor there is a lighter blue button with the name "File". If this button is pressed, three new buttons will appear under it. The "New" option cleans the whole grid, removing all the objects in it. "Open" makes a new window pop up and asks the user to select a XML file exported previously by the modelling program to import it to the new grid. Finally, "Write" pops a win-

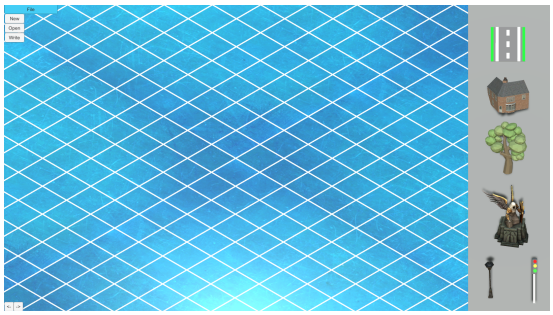


Figure 2: Modelling starting screen

dow so the user can select the destination and name of the XML file that is going to be created with the objects in the current grid. On the left lower corner of the screen we can see two buttons with characters representing an arrow pointing to the left and another pointing to the right. Each button makes the camera turn either ninety degrees or negative ninety degrees, depending on the button, with the center of the screen as the middle stationary point. On the right part of the screen there is the list of objects that can be placed on the grid itself.

After the object is selected, and by consequence, highlighted, the user can now try and place it on the desired grid's cell. Before placing the object in one of the cell's grids there is another highlight feedback on the grid itself so the user is sure where he or she will place the selected object. The list of objects include Roads, houses, trees, statues, street lamps and traffic lights. When accessing the object's properties a new options menu will appear in the bottom middle of the screen. The selected object has a blue tint to it so it is faster for the user to recognise which object is going to be manipulated. All of them have the red button "Del" on the top left corner to delete the object if the user desires.

3.2.2 Visualisation

The user starts with a blank environment and a single button on the up left side of the screen. This button opens an operating system's windows to let the user choose which XML he or she wants to open. After choosing the modelled city file the corresponding city is automatically built.

The art for the city is a low poly type of style, which means objects do not have much three dimension detail but the simplicity makes the city a less confusing and overwhelming one, and by consequence, a welcoming feeling.

The roads are similar to the modelling program ones. They can appear in four formats: Single, turn, T crossroad, and '+' crossroad. With the XML file the VR program knows how to rotate the road element so it sticks together with the adjacent ones. There are three type of buildings where each one has a unique colour: blue, beige or black. Then each building has three different levels of floors ranging from 1 to 3. The trees have a simple brown log with three different types of leaves. Statues are also an aesthetic aspect of the environment. The dynamic objects in this world will be cars, simple four wheeled objects that move around the environment. Their movement is controlled mainly by the communication with the students code received over UDP. Although in the tests that will be referenced in the next chapter, these cars are controlled manually. By manually, it means each car has its own script for the specific route or test run. For more artistic objects, Street lamps, can be placed in the city and can illuminate the nearby roads, since each lamp has a source, cone shaped, light pointing towards the ground. Lastly, traffic lights, also get information through the same system as the cars over UDP from students making it switch its state from red light to green light to yellow light and repeating.

After the simulation reads the previously exported XML file created with the modelled city information, users enter this world with the Virtual Reality glasses.

4. Results

4.1. Experiment Overview

4.1.1 What is going to be evaluated

Tests were run to, mainly, compare the selected selection techniques: SmartRay, ShadowCone and IntenSelect, in order to see which one would have the best results when it comes to selecting a dynamic object in a dynamic virtual world, having three different objective metrics.

4.1.2 Methodology

The tests were performed by randomly picked users individually, each one doing three different scenario, for each of the three different techniques. The order of the tests were as follows: Presentation of the project, it's goals and an overview of the techniques at cause, getting the user's consent, users answering a simple questionnaire about their own information, starting the experimentation, users giving feedback on the completed task, presenting the users a small game to play as reward for taking the experiment.

To gather information about the tests, photos and videos were taken and recorded with the help of a smartphone's camera, registering the computer's main monitor (using Open Broadcaster Software, a third-party program) to get what the user could see throughout the whole experiment and saving raw numbers by Unity's scripts, where the tests were being run.

4.1.3 Tasks

The users were presented with the following task:

- A yellow car will appear three times in three different locations at different times in each scenario. Select only the yellow car between the other coloured cars (can be red, blue, green or black) for each of the three different scenarios with each of the different selection techniques.

4.1.4 Tests

For each technique, all of the three scenarios are played separately and individually, meaning they all have a start and an end, being executed sequentially. All cars appear with fixed intervals. The yellow car has a fixed timer for when it appears in the current scene, this way the consistency on the tests is preserved. After the yellow cars appears three times, then the application is restarted but this time with the next scenario loaded until all three of them are over.

The first scenario is a simple roundabout. Cars in this scenario move counter clock wise and usually take the second exit, except the targeted yellow car, making it the simplest scenario, user can get used to the experiment before going to more complex maps. The second is a small city with some buildings and other aesthetic objects, this way users were encouraged to move around the city or find new viewpoints to better look at everything at once, making the environment more dynamic than a usual steady camera with a single viewpoint. The last case is a mimic of a highway with four lanes. Cars move at different speeds, with higher speed the more to the left lane they are, this scenario is the most complex one because the point is to see how each technique works when the object that needs to be select is in a cluster and/or at a distant range. For each scenario the yellow car appears three times, with different routes. This way we give a second and a third chance to the user to get used to the technique if needed.

The tests begin with SmartRay because the technique is easier to get familiarised with, this way users can get used to the environment without being too overwhelming with the technique they first encounter. Shadow Cone is the next technique to be tested, and the experiment ends with IntenSelect.

4.1.5 Metrics

For the previously stated experiments the objective metrics being saved were: Technique's success rate; Time to select the car; Number of clicks; Hit rate.

The technique's success rate is calculated whether or not the user was able to select the targeted car before it could leave the scenario, for example, if the participant was able to select only 1 car out of 3 from a given scenario, the success rate for that scenario for the given technique is 33.3(3)%. This lets us know how hard or unfamiliar it was for the user to use and succeed with the given technique.

In each scenario the yellow car has three different courses. For each time the car appears, the participants can try and select the target as many times as they want to. If we take t as the number of tries to select the target on one of the three times and x as the time for each try, we can calculate its mean, mt. Because three yellow cars appear in each scenario we calculate the mean of all three means for the specific scenario, $mThree$. After getting the mean time for one scenario ($mThree$) and doing the same process for the other scenarios with different x's, we gather the three different means from each scenario and get the mean of the whole technique for the current participant, $mExp$:

$$mExp = (\sum_{j=1}^{3}(\sum_{i=1}^{3} mt_i)/3)/3$$

Time to select the car is calculated by doing a final mean with all the tests run using the different 15 $mExp$'s.

The number of clicks is a metric that shows how many times the user, in average, had to try to select the yellow car to finally select it.

It was also considered hit rate which is a different metric similar to success rate. Our previous one, just took into account if the yellow car was either selected or not at the end of its course. Now we can calculate it for each click. If it took a participant 4 tries to select the desired car before the end of its path, we'll now consider the success rate 25% instead of 100%.

4.2. Analysis

4.2.1 Demographics

A total of 14 people, half male and half female, with ages ranging from under 18 to 55. More than 70% played any type of games (mobile games, computer games or browser games) around an hour or two a day. Half of the remaining percentage played less than one hour a day or haven't played games in years, while the other half played from 7 to 8 hours a day. When asking about past Virtual Reality experiences it was included HmD, tablets and smartphones, making the percentage of people that didn't have any sort of VR experiences around 30%. When talking about Occulus Rift's experience, only 2 of the participants did have past experiences with it.

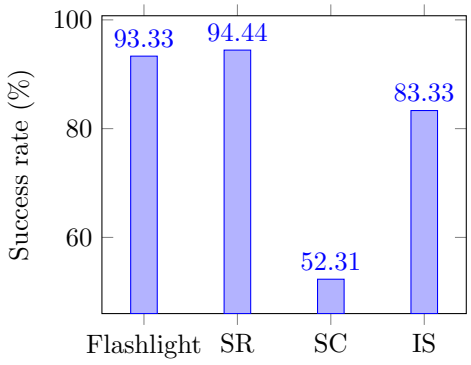
4.2.2 Results

The first objective metric is success rate. The yellow car appears three times per scenario, this means each technique will try and select 9 yellow cars at different times. Success rate is a percentage representing how many times the technique selected the targeted car out of the possible 9 times.

We can first see all success rates are high (above 80%) except ShadowCone. This was due the fact participants sometimes had a hard time selecting only the yellow car, meaning, the car wasn't usually the only one being selected before leaving the scene, considering then a miss and not a hit. Other times, the participant's hand's jitter and/or some occlusion of building or other objects in the city would make users miss the yellow car even for a split second and had to retake the selection or miss completely.

Flashlight, SmartRay and IntenSelect had the best results considering success rate with no major differences. Participants would usually hit the yellow car, with some small exceptions, for example, another car obstructing the pretended one, losing vision for a while or even the remote would send not accurate information about its position because sensors are not accurate all the time.

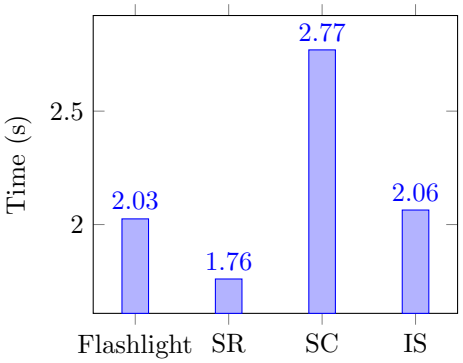
IntenSelect could have a lower success rate because a few participants, when answering the post questionnaire, said the technique was too overwhelming sometimes. Having a cone and a ray may be too much sometimes. Not only this subjective aspect, but when analysing the calculations made in the program, when two cars were close enough to each other (because IntenSelect adds weight to every car inside the cone and the closer to the ray the heavier the weight is) and because of the remote's and participant's hand's jitter the non-yellow car could be selected instead.



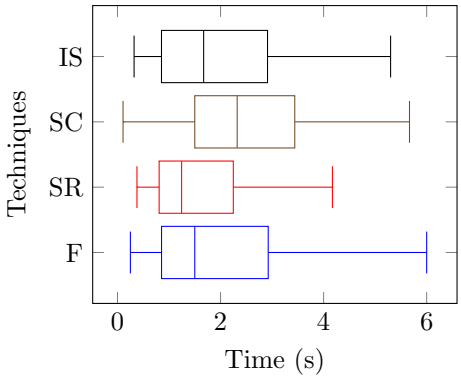
ShadowCone was the technique that had the higher amount of seconds to select the car since it's a technique that requires some time to refine the pretended objects, participants had to keep pointing at the car until all the other possible selected objects would leave the cone selection.

While SmartRay and IntenSelect had similar selection time, flashlight had a way lower selection time. This is due to the fact participants usually didn't hold the button to select when using flashlight unlike when using the other techniques. SmartRay and IntenSelect have a continuous refinement, meaning they can take some time to choose one car over the other. Flashlight on the other hand was used as an instant/no refinement selection technique. Participants usually pressed and released the selection button in under a second and try again if they missed.

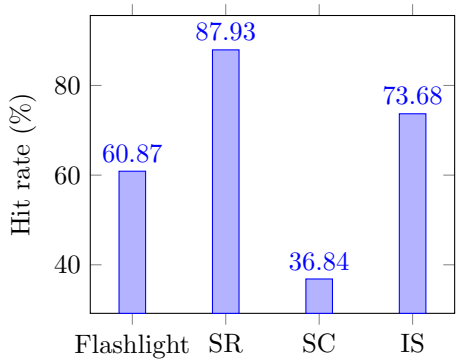
SmartRay had the least amount of clicks since participants could usually be able to select the car with one try. ShadowCone had a similar number of clicks because to select the car the time required was high meaning they had less opportunities to shoot the cone again. This way we can see Flashlight had a higher amount of clicks than all the other techniques. Because users wouldn't refine their selection, there were many misses before actually hitting the desired object. If now we say the time to select the object is the sum of every tries' time (instead of the tries' mean) we see the mean time usually rises, but even more significantly when looking at Flashlight. This data shows the average time a user takes to select the car from the moment he or she starts to try and select it. ShadowCone takes the longest to complete the task while SmartRay is the fastest.



Even though Flashlight had the second fastest median from all the techniques, it wasn't consistent. Flashlight was the technique with the highest time recorded, with 6 seconds, to select the wanted car. With this data, we can see SmartRay has the fastest scores, not only with its median, but also a smaller quartile compared to other techniques.



Finally, after getting data from the success rate and number of clicks, we see each technique's hit rate, which shows us the participant's difficulty or accuracy while using a technique. SmartRay has the best hit rate since less tries were needed to select the target. This is because participants usually followed the car for enough time to make its weight higher and higher compared to others. Users could hit other cars by mistake or lose sight for a while, but the yellow car's weight was high enough to still select it. Now it can be seen that Flashlight falls down from the previous metric: success rate. This is simply because it is now taken into consideration the number of misses over the total amount of tries. Lastly, ShadowCone has the lowest hit rate, not only because the number of tries was high but also because, before the yellow car could finish its course, participants couldn't select it in time.



5. Conclusions

The modelling and visualisation applications were developed successfully, according to their objectives and goals with some minor bugs. The modelling allows users to model cities, edit objects and export the modelled city. The development and integration between Modelling and the Visualisation was also implemented successfully. The visualisation imports the cities correctly and allows users to move around the imported environment. The simulation communicates with the visualisation, with the current SmartMob's status. The visualisation represents cars according to the received information from the cars in the laboratory. In this, it is also possible to select objects using SmartRay.

SmartRay was the most successful technique in this experiment. Having the highest hit rate and lowest average amount of time when selecting the right object. It is a simple but effective technique with a small learning curve that outperforms the other techniques in the studied scenarios. IntenSelect had similar results to SmartRay, its weight algorithm and considering, not only the intersected object, but also objects inside the cone, was beneficial with some participants but a problem to others. That's why some users had better results with it. The experiment was also a success. We were able to test all four techniques with multiple different people with different ages, experience with games and education completed.

5.1. Future Work

Due to the lack of time and resources a validation of SmartMobVR with the course's students was not done in this project. Information about cars is received by the simulation, if this project continues to be worked on, there could be a visualisation of that data in different ways for different objects, for example, cars and traffic lights.

Cars in a city can be predictable and have limited areas where they move around. For future work and to better test the techniques in a dynamic environment, a flock of birds could be a scenario to test these on. Birds are unpredictable and have different speeds and turns in a three dimensional space, while cars are bound to the ground, which is two dimensional.

References

[1] G. De Haan, M. Koutek, and F. H. Post. Intenselect: Using dynamic object rating for assisting 3d object selection. In *IPT/EGVE*, pages 201–209. Citeseer, 2005.

[2] T. Grossman and R. Balakrishnan. The design and evaluation of selection techniques for 3d volumetric displays. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 3–12. ACM, 2006.

[3] D. Medeiros, A. Sousa, A. Raposo, and J. Jorge. Magic carpet: Interaction fidelity for flying in vr. *IEEE transactions on visualization and computer graphics*, 2019.

[4] A. Steed and C. Parker. 3d selection strategies for head tracked and non-head tracked operation of spatially immersive displays. In *8th International Immersive Projection Technology Workshop*, pages 13–14, 2004.