# Virtual Reality Project Report
# 3D Graph Visualization and Navigation, Group 003

Diogo Sá, Jorge Nunes, Margarida Lima and Tiago Neves

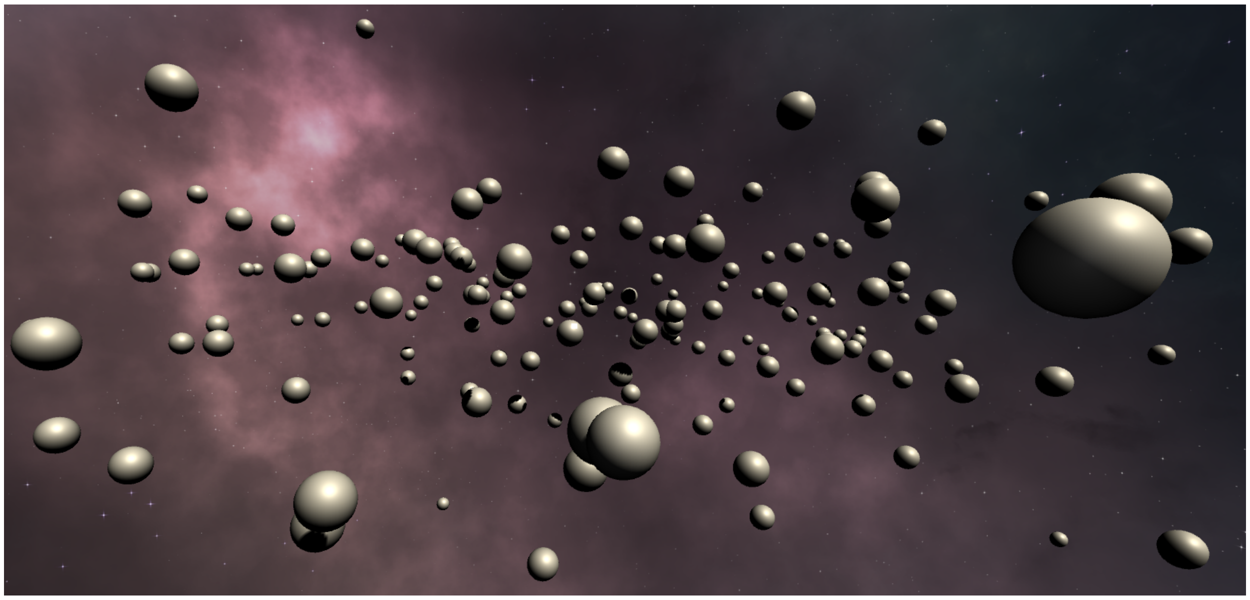

Fig. 1. First view that a user has of this implementation. This figure represents a 3D graph composed by nodes (withe spheres) and edges. However edges are hidden by default in order to not visually pollute the environment. It is possible to see a node's edges by marking the node, see Section 4.7

**Abstract**— Our project compared the usage of several devices to interact with 3D graphs by efficiency. The performance of each configuration was measured by the time that certain tasks took to do and by the amount of wrong nodes that were selected while performing the task. We created a 3D graph with real information from *IMDB* movies dataset, and showed it to the user through an appealing interface.

**Index Terms**—Virtual Reality, Graph, 3D Navigation

## 1 Introduction

Using Virtual Reality in order to change the way we interact with certain environments might be the way to further optimize and simplify what once was a difficult task.

In this project we focus on the creation of testing environments for 3D graphs, so it is possible to study the most efficient and preferable way to visualize and navigate through these graphs, while being able to retrieve high levels of accurate information.

Our goal is to compare 3 (three) different configurations of 3D graph navigation composed by different inputs and outputs in order of efficiency on performing different tasks. These 3 (three) configurations are: keyboard, mouse and a computer monitor; keyboard, mouse and an Oculus headset; Oculus headset and both controllers.

Even though the prepared tasks for testing are the same in every environment, the way it is carried out is fairly different. The change of direction of line of sight and movement trough the 3D graph is very dependent on the configuration, which allows us to study the time taken for each task and error rates.

With this project we are looking to find the answers to the following hypothesis:

- Is VR the best setup to get information from 3D graphs (time to complete tasks and errors per task)?
- Is the two-handed flying the best way to navigate in 3D graphs using a VR headset?
- Is VR the most enjoyable way to work with 3D graphs?
- Does our system make the user sick?
- Does our system have any room for improvement?

## 2 Related Work

*Virtual Reality* is hot topic nowadays, however it has not achieved its full potential. One of the challenges in *Virtual Reality* is the fact that there is not a defined norm, yet, that specifies how interaction should be made. This can be applied to navigation or interaction with the

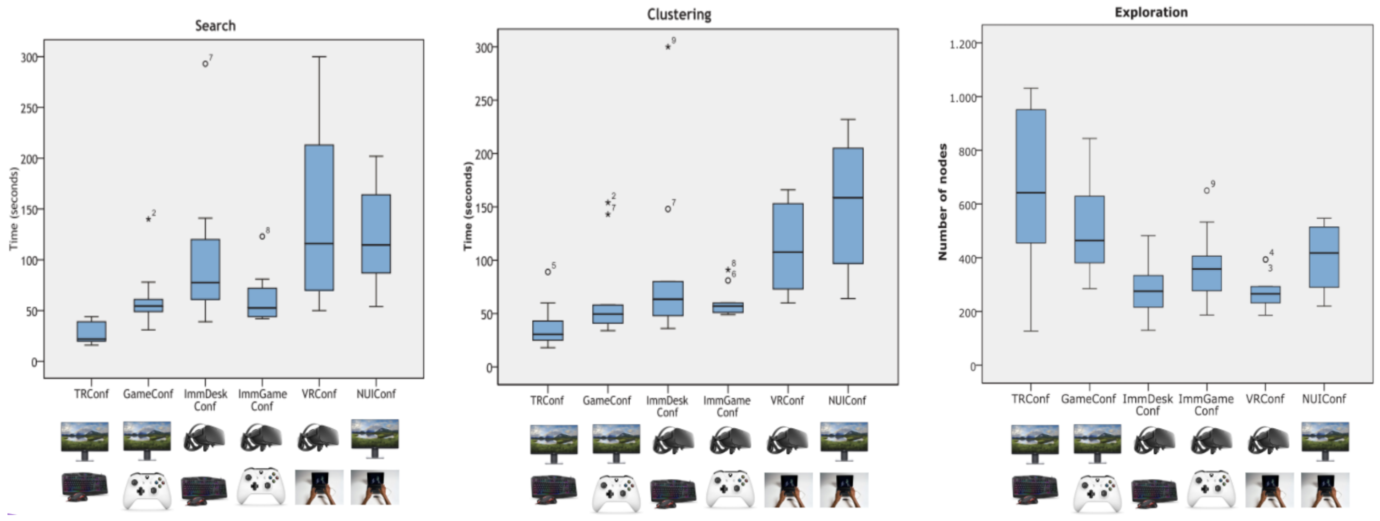

Fig. 2. Results obtained in project [1]. Each graphic represents a different test performed by the several configurations present on that project. The first one (left) corresponds to the task of searching for a specific node in the environment, the second one (middle) represents the task of clustering two nodes and finally the last one (right) corresponds to the task of exploring the environment. The first two graphics have measures corresponding to the time of how much each task took to complete using a specific configuration, while the last one uses the amount of nodes that a user clicked as measure.

| | | **Outputs** | |
|---|---|---|---|
| | | ***LCD monitor*** | ***Oculus rift*** |
| **Inputs** | ***Keyboard/mouse*** | Traditional Configuration (TRConf) | Immersive Desktop Configuration (ImmDeskConf) |
| | ***Joypad*** | Gamming Configuration (GameConf) | Immersive Gamming Configuration (ImmGameConf) |
| | ***Leep motion*** | Natural User Interface Configuration (NUIConf) | Virtual Reality Configuration (VRConf) |

Table 1. Configurations tested on project [1].

Fig. 3. A visualization of Oculus Quest from *Facebook*. The goggles are represented in the middle while both controllers are on the sideways.

virtual components. Right now, there is several gadgets that allow an immersive and efficient experience. For instance, *Leap Motion* allows the user to interact with an environment using gestures. This approach is valid because the user do not need to have anything on its hands in order to interact. *Facebook* launched Oculus Quest, represented in Figure 3, with a promise of revolutionize *Virtual Reality* experience, the device comes with a pair of goggles and controllers that allow the user to interact with the environment in an immersive way.

We based our approach in two different articles [1,3], and each one of these articles approaches the 3D graph navigation in different ways. In [3] several configurations - represented on Table - of inputs and outputs are tested against each other. In the end, the results are compared in terms of how much certain tasks took in order to measure the efficiency of the configurations. The results are represented on Figure 2.

We adopted this strategy, of pairing inputs with outputs and test each configuration, as well. However, we did not use the same amount of devices that Ugo Erra, Delfina Malandrino and Luca Pepe used in [3]. This paper explores the effectiveness of visualizing and interacting with three-dimensional graphs in VR in comparison with the traditional approach. In particular, they present an empirical evaluation study for exploring and interacting with three-dimensional graphs using Oculus Rift and Leap Motion. We observed the fact that Leap Motion executed poorly in all tasks, and decided to create a similar test case as the authors with a different goal. In our case, we wanted to compare the interactive procedures of traditional, semi Virtual Reality and Virtual Reality, for more details see Section **??**.

In the second paper [1] the authors wanted to explore how users could navigate large three dimensional data that has no specific orientation, while supporting their sense of presence when faced with a particular task using different navigation techniques. The authors compared two commonly used techniques (Teleportation and One-Handed Flying) against two less common methods (Two-Handed Flying and Worlds-In-Miniature) and evaluate their performance and effectiveness through a series of tasks. This techniques were developed for the Oculus Rift VR system, represented on Figure **??**. This techniques were thoroughly tested to find which were the most efficient, less physically and mentally demanding and overall more satisfactory to use. The authors deemed the Steering Patterns (One-Handed Flying and Two-Handed Flying) to be faster and preferred by participants for completing searching tasks in comparison to Teleportation. Worlds-In-Miniature was the least physically demanding of the navigation's, and was preferred by participants for tasks that required an overview of the graph such as triangle counting. This conclusions followed from the data collected from the user tests,

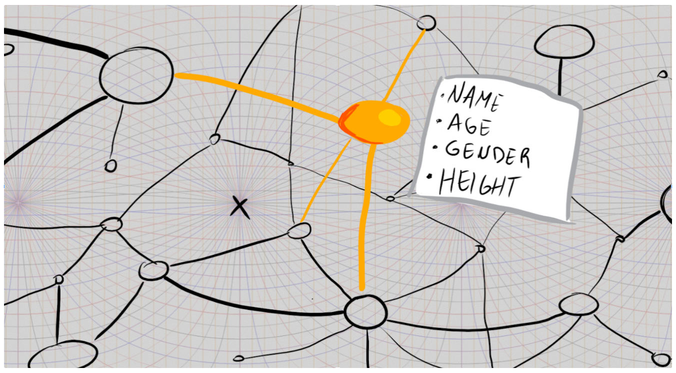

Fig. 4. 3D sketch of the graph with a selected node displaying a movie's information, seen from the perspective of the user.

some of it shown in the following figure **??**.

For our paper we combined the testing methodology of the first paper with the specific focus on navigation techniques from the second. We experimented with one of the best navigation techniques from the second paper (Two-Handed Flying) but ended up experimenting with new forms of navigation and manipulation of 3D graphs. We also tried to generate the graph so that it takes advantage of the 3D space, rather than having an arbitrary generated graph, to see if this would improve overall satisfaction and performance. This will be further explained in the next sections of this report.

## 3 System Design

The design process started after we studied the papers exposed in the Related Work section3.4. We wanted to experiment with different approaches to navigation and user testing and to have a more practical approach to the development of the application by using a real dataset to generate the graph. This project was divided into three different milestones. Each milestone was designed to aggregate themes inside a same category. Before starting with the milestones we went to the prototype phase, turning our ideas into more tangible work. Throughout the semester, we made sure to refine our design and implementation to fix design flaws and improve the application, taking in consideration the feedback given through the development phases.

### 3.1 Prototype Phase

In the prototyping phase we established the basic functionalities that we wanted the project to have. We though about the controllers in a way that would be intuitive to use, the graph surrounding the user, the ability to highlight nodes and edges and access a node's information. We drew a storyboard and a 3D image prototype that can be seen in figure 5.

### 3.2 First Milestone

This first milestone consist on creating and preparing the project in order to implement the *Virtual Reality* module, see Section 3.4 for more detail. Since we are approaching graph 3D navigation, a real dataset was used in order to add realism to the project.

#### 3.2.1 Dataset

IMDb (an acronym for Internet Movie Database) is an online database of information related to films, television programs, video games, and streaming content online. We used IMDB movies dataset in this project in order to create a graph based on movies.

This dataset consisted on more than 45000 (forty five thousand) movies with plenty information about them - release date, title, description, language, production, etc. The dataset also contains several other files like authors that entered in each movie, directors, etc. However we wanted to keep it as simple as possible and only considered the movies themselves.

#### 3.2.2 Generating 3D Graph and Navigation

Another major tasks of this milestone was to create a 3D graph with the dataset information. We accomplished this by mapping each movie values to an object - *DecodedNode* - in a way that hey were easily accessible by other objects. The dispersion of the nodes was important as well, because we wanted the nodes to be positioned in a particularly way and not at random. The final subject of this milestone was to add basic navigation to the application.

### 3.3 Keyboard Mouse Inputs

Basic navigation movements such as moving to the front, back and both sides, as well as looking around according to the movement of the mouse were implemented.

Selection and marking features were also implemented, along with the outline visual feedback and sound feedback.

### 3.4 Second Milestone

The second milestone consists on the integration and implementation of *Virtual Reality* components to our application. That implementation includes the navigation throughout the environment and basic interactions. We also continued the pursuit of a better way to generate the 3d graph.

### 3.5 Third Milestone

In this Milestone we fixed bugs that we discovered during development, implemented the UI to display the node's information and developed the software that would conduct all the user tests as well as register the relevant information from said tests.

## 4 System Implementation

In this section we describe how we implemented each part of this application.

### 4.1 The System

The system we used in order to create our system was Unity. Group members already had prior experience with this engine, which provides all needed features in order to create all the environments for testing, develop scripts for movement and interaction, integrating information-rich datasets and retrieve useful information from future user tests.

Unity allows the creation of different scenes, so a different environment was created in each scene. 3D objects were used for prefabs, which were then used to create the spheres and cylinders used for the nodes and edges of the environments. C scripts were used to create all the possible events of each environment.

Another of Unity's good features is the Unity Store, from which it is possible to download tools to integrate the Oculus Quest and tools like a spatial-looking skybox and outline mechanic, which was then used for node interaction, selection and marking.

### 4.2 Dataset Parsing

The first step was to parse the dataset in order to make it useful for us. The first step was to be able to read the information in order that can be transcribed into our application. The information regarding the movies information is in a *CSV* file, in which each row correspond to a distinct movie and each row to a variable. Each row was separated by a comma (","), what facilitated the parsing of the file.

Each line was parsed into an object - *DecodedNode* - with variables that contained the information present in the *CSV* file. At the end, depending on how many nodes are required, we have X objects with information about X movies of the dataset.

### 4.3 Generation 3D Graph

To generate the graph we populate a specific number of nodes with movie information taken from the data-set, and then we connect the nodes with the same genres of movie. The idea was that this combined with a force function would make drive nodes towards or away from each other according to their genre's and from their some order would emerge .

We researched algorithms to do this to better use the added dimension in 3D. We used a 2D force function to obtain the wanted results from the first milestone 3.3, generating results such as the one shown in Figure **??**. The algorithm used was taken from the book *Graph Drawing* by Stephen G. Kobourov [4], and goes as follows: In the algorithm the variables c1, c2, and c3 are constants while d

```
1: place vertices of G in random locations;
2: for $i = 1,2,\dots,SIMULATION\_STEPS$ do
3:     for $i = 1,2,\dots,GRAPH\_NODES$ do
4:         for $i = 1,2,\dots,GRAPH\_NODES$ do
5:             if nodes are connected then
6:                 $vertex_force = c1 \times log(d/c2)$,    ▷ attraction force
7:             else
8:                 $vertex_force = c3/d2$                   ▷ repel force
9:             end if
10:            $c4\times$ vertex$_force$;                   ▷ apply force to vertex
11:        end for
12:    end for
13: end for
14: draw graph.
```

is the distance between the two connected nodes. GRAPH_NODES corresponds to the number of nodes we want in the graph and SIMULATION_STEPS to the number of iterations we want the algorithm to run. We also took advantage of the third dimension by giving nodes a height value proportional to the node's movie IMDB score.

### 4.4 Keyboard and Mouse Movement

For the keyboard and mouse movement controls we followed the structure used in the paper [3], it being very close to what is used in 3D applications and, therefore, the state of the art.

Movement (from both keyboard keys and scroll wheel), rotation, selection distance and marking distance parameters are easy to change with some tweaking. This can be done using the Unity Editor, since all of this variables are exposed. Users might even change these values while running the application.

The direction of the user's movement is dependent of it's local axis, instead of the global axis. This will make for a user who presses the W key to be always moving in the direction they are looking at. This logic is used for all other keys in order to keep coherence and keep the system as simple as possible.

### 4.5 Virtual Reality Integration

In order to work with the virtual reality headset and controllers, it's software had to be integrated. This was done by adding the Android Build Support to the Unity version of the project and installing and updating Android Studio.

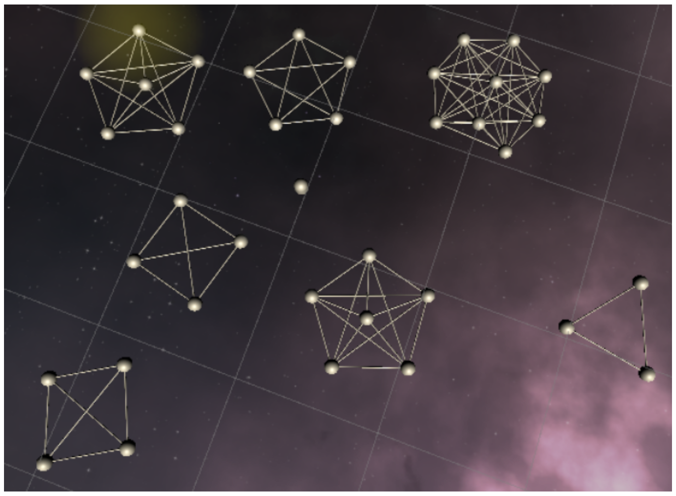

Fig. 5. Example of results from the 2D force algorithm.

| Keyboard and Mouse | |
|---|---|
| W | Move forward |
| A | Move left |
| S | Move back |
| D | Move right |
| Q | Move up |
| E | Move down |
| Mouse* | Look around scene |
| Left mouse button | Select node |
| Right mouse button | Mark node |
| Scroll wheel | Fast move forward/backwards |

Table 2. Mouse and keyboard controllers key mapping cheat sheet. (*) On the Keyboard and mouse + Oculus Quest configuration this control is replaced by the user's physical action of looking around.

#### 4.5.1 Virtual Reality Movements Setup

| Oculus Quest Controllers | |
|---|---|
| Left analog | Move commands (equivalent to WASD) |
| Right analog | Fast move (equivalent to scroll wheel) |
| Y | Move up |
| X | Move down |
| Left hand trigger | Select node |
| Right hand trigger | Select node |
| Left index trigger | Turn on/off node selection raycast |
| Right index trigger | Turn on/off node selection raycast |
| A | Mark node |
| B | Two handed flying |

Table 3. Oculus Quest controllers key mapping cheat sheet.

Movement (from both Oculus controllers joysticks), rotation, selection distance and marking distance parameters are easy to change with some tweaking. This can be done using the Unity Editor, since all of this variables are exposed. Users might even change these values while running the application.

The direction of the user's movement is dependent of it's local axis, instead of the global axis. This will make for a user who move the left joystick up to be always moving in the direction they are looking at. This logic is used for all other keys in order to keep coherence and keep the system as simple as possible.

The two handed flying technique was implemented and available when using the Oculus controllers: when pressing the B button for the first time, the average position of the controllers in the world is calculated. When pressing B for a second time, it calculates once again the average position of the controllers in the world. This creates a direction vector from the first calculated point to the second. When multiplying this vector by a value (that can be edited in the Unity Editor), we get a new position in the world, to which the user is teleported.

### 4.6 Node Selection

The primary mechanic (besides movement which enables navigation) is the selection of nodes. This enables the user to retrieve information (as referenced in 4.9).

When the user is within close range of a desired node and aiming at it, its outline turns white. When a node's outline is white it is able to be selected. After selecting a node, its outline turns orange, the node information is displayed and the user's movement is halted. This happens in order to prevent possible confusion with other nodes that are close. The outlines where implemented with the help of a shader asset from the Unity asset store [5]

Node selection is also used in the tasks, when the user must select a certain node.

### 4.7 Node Marking

Another mechanic implemented to ease the retrieval of information from the 3D graph is node marking.

When marking a node, its edges appear. These edges connect all the nodes with the same gender as the one that was marked. The node and its edges also have their outline color changed to better contrast with the rest of the environment. When aiming at a node, if it's marked, its outline color will change to green and, after other mark, it will turn blue, red, yellow and no outline.

Node marking is a useful feature to ease task completion.

### 4.8 Other Mechanics

After implementing all the project's mechanics, there was a need for a last one: rotating around a selected node. After selecting a node, the user's movement is disabled. Although handy, this might affect negatively the user if there are marked nodes and hence edges on the map, which might stand between the user and the displayed node information.

The possibility of losing sight of the information and not being able to turn on itself, the rotate around the node mechanic was implemented, which solved both the last faced problems

### 4.9 Node Information Display

After a node being selected, its information is displayed between the user and the respective node. The scale of the information panel is inversely proportional to the distance of the position of the user (in the system) to the selected node.

All information is parsed from dataset, but only some of the information was chosen to be displayed since not all were as useful. The chosen displayed information is the title of the movie, its genres, the IMDB score and the release date.

The node information is only hidden after deselecting the node.

## 5 User Study

### 5.1 User Tests Procedure

There was a total of 3 (three) different tests within-subjects test models, which means that every user is exposed to all 3 conditions in a latin-square sequence. This sequence was implemented based on this guide [2]. The conditions themselves are very similar, where the only changing variable is the equipment. There are two inputs and two outputs, and their combination creates the tests shown in Table 5.1.

| | | **Outputs** | |
|---|---|---|---|
| | | *LCD Monitor* | *Oculus Quest* |
| **Inputs** | *Keyboard + Mouse* | Test 1 | Test 2 |
| | *Controllers* | * | Test 3 |

Table 4. Tests resulting from the combination of outputs and inputs. The combination (*) will not be tested because the controller's input can't be translated to an output on a monitor.

Since each user performs all tests, let's assume that we first start by Test 1, followed by Test 2 and end with Test 3. For each one of the tests on Table 1 there is be a correspondent questionnaire, to gather information about it.

### 5.2 Subject Characterization

*You will start by sitting comfortably in a chair with a computer in front of you and fill the first part of the first questionnaire, which will ask information about you anonymously.*

This kind of information is crucial in order to map the results obtained from the tests to some kind of demographic group that might exist. For example, people that study/work in IT might turn out much better at the tests than people that have a different occupation. All 3 questionnaires will have this first section, and you should answer them in the exact same way.

*After the first part of the questionnaire is completed, and we already have the information about you that we need, it's time to test the project. Since you're performing Test 1, which means that you will be manipulating a computer with a keyboard + mouse and watching it through a monitor (which is the "normal" way of using a computer nowadays), no much adaptability is needed in order to know how to use the peripherals.*

### 5.3 Exploration Phase

Once the application starts, the user is free to explore it up to 10 minutes. This means that for 10 minutes, the user navigates freely through the application and tests things on their own: moving, clicking, pointing and interacting with the nodes. The user is surrounded by nodes in every direction and is be able to navigate through them. Each node consists of a movie, and nodes are related by their categories/genres.

At the end of the timer, the user is more familiarized with the project which will help them with the next stages of the testing. Users also need to complete the questionnaire about the exploration phase. This section is important, because it allows to understand if the application is intuitive or not. If the user is able to do most things that the application allows in 10 minutes on their own, it means that the app is in fact intuitive.

### 5.4 First Task

Exploration task is followed by the test's first task. In order to start the task itself, ENTER key must be pressed. This way, users can spend the time they need to fill the questionnaire. As soon as ENTER is clicked, a specific node will turn red and the user's goal is to find that node. Node will provide visual feedback (bright red light) to signal its position in the map.

Once the node is found, it needs to be kept at the center of the screen (user is still performing Test 1) and to be close enough to be able to select it by checking if it's highlighted with a white outline. If it is, it can be selected by pressing Right Mouse Button RMB. Since each node corresponds to a movie, whenever a node is selected, a 2D panel appears with some information about that movie.

However, this is a small task that can be done in less than 30 (thirty) seconds. For this reason, this needs to be repeated 14 (fourteen) more times and find in total 15 (fifteen) different nodes. Once a red node (the goal node) is clicked, that node will become white, and another node will turn red, until this process is repeated 15 times. At the end of the 15th repetition, the first task is concluded and the user can fill the section of the questionnaire referent to the first section. After filling that section of the questionnaire, the ENTER key must be pressed to continue to task 2.

## 5.5 Second Task

First task is followed by the second task after ENTER key is pressed. After task 1 ends, the last red node from that task becomes white and the first red node from task 2 appears. Both tasks look the same, but they have a different goal: while in the first one a red node must be found and selected, on task 2 the red node's neighbor that has higher classification has to be found and selected. That information can be seen on the 2D panel that appears when you click on a node, among the rest of the node's information.
A good strategy for this task is to use the ability to mark nodes. To mark a node, the user must be in position to select it (have the goal node at the center of the screen and close enough), but instead of clicking RMB, Left Mouse Button (LMB) must be clicked. This action will create a colorful outline to the marked node and to all of its edges. This feature allows the user to find neighbors more easily, which helps them in their task of comparing all neighbors from that node.

However, this is a small task that can be done in less than 1 (one) minute. For this reason, users need to repeat this task more 4 (four) times, and find in total 5 (five) different node's highest classified neighbor. At the end of the 5th repetition, the second task will be concluded and the user can fill the section of the questionnaire referent to the second section. After filling that section of the questionnaire, the ENTER key must be pressed to continue to task 3.

## 5.6 Third Task

Second task is followed by the third task after ENTER key is pressed. After task 2 ends, the last red node from that task becomes white, and the red node from task 3 appears. The goal of this task is to find all nodes that have the exact categories as the selected (red) one.

To complete the goal, the user should navigate close to the red node, select it and check their neighbors afterwards. The marking feature will come in handy in this task as well in order to find out the red node's neighbors. The task ends when all the neighbors nodes are selected.

Contrary to the other tasks, this one will be performed just once. At the end of task 3, the test finishes and the users need to complete the rest of the questionnaire - it includes the part referring to task 3 and the final part. This final part aggregates a few normalized questionnaire types, such as System Usability Scale (SUS) and NASA TLX, in order to summarize the usability of this specific configuration (Test 1 from Table 1).

## 5.7 Results

This section contains a detailed explanation of the results obtained in Section 5. Since we actually live in a pandemic situation, it was not possible to test this application with a wide group of people. All results shown in this document were obtained in each of the group members house, and the testers were family members. For this reason, the result might not be trustworthy, and we invite other researchers to perform this tests too with a more reliable group to see if it has any impact on the results.

We needed to completely remove 3 (three) tests because tests were run on a wrong version of the project.

### 5.7.1 Subject Characterization

The total of subjects used in the tests were 6. We understand that this value is not even close to be acceptable as a study, and that is for that reason that we invite all researchers to confront our results and reach their own conclusions. This individuals had the age between X and X and their knowledge about graph theory diverged a lot: X individuals knew about it very well, while X didn't anything about it. Their experience with *Virtual Reality* also diverged: X individuals were very familiar with *Virtual Reality* technologies, while X people weren't.

### 5.7.2 Time Analysis

A type of evaluation that we will do here is by evaluating how the users performed the tasks. We will evaluate this by measuring the time that each user took per each task, the results are represented in Figures 6 and 7. On these graphics it is possible to see that in general, Test 1 (Keyboard/mouse + LCD monitor according to 5.1) were the best ones. This is a fair result, because it shows that the traditional interaction is the most efficient of them all.

The second best configuration proved to be Test 2 (Keyboard/mouse + Oculus Quest according to 5.1), both Test 2 and Test 3 had a similar average execution time. However, the difference on the amount of wrong selected nodes is clear. On Test 2 the average of wrong selected nodes is approximately 2 (two) for Task 1, 8 to Task 2 and 0 to Task 3, while for Test 3 is 1 for Task 1, 13 for Task 2 and 1 for Task 3. Thus, even if the execution time is the same on both tests, we conclude that Test 2 is more efficient because the users selected in far less wrong nodes than on Test 3.

### 5.7.3 System Usability Scale (SUS)

The *System Usability Scale (SUS)* provides a "quick and dirty", reliable tool for measuring the usability. It consists of a 10 item questionnaire with five response options for respondents; from Strongly agree to Strongly disagree. Originally created by John Brooke in 1986, it allows you to evaluate a wide variety of products and services, including hardware, software, mobile devices, websites and applications. The *SUS* survey has 10 statements and it uses a *Likert Scale*. The user can evaluate at which level the statement is agreeable. So the user answers each question with a value between 1 and 5. Who is interpreting the data then has to apply a simple formula to the number. The resulting value is between 0 and 100 but is not a percentage.

The results that we had on our project depended on the configuration that the user was using. Test 1 had 60, Test 2 had 61 and Test 3 had 74. By observing this results alone, we would consider the Test 3 the best option to interact with 3D graphs. However, by looking at Figures 6 and 7, we understand that Test 3 is the worst configuration in terms of

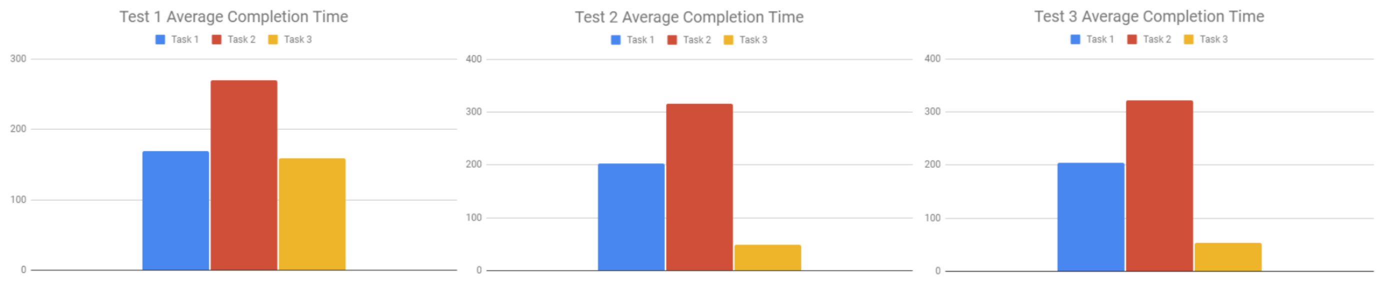

Fig. 6. Average time that each trial per task took in milliseconds. On the left we have the correspondent times for Test 1, middle corresponds to Test 2 and right corresponds to Test 3 (see Table 5.1).

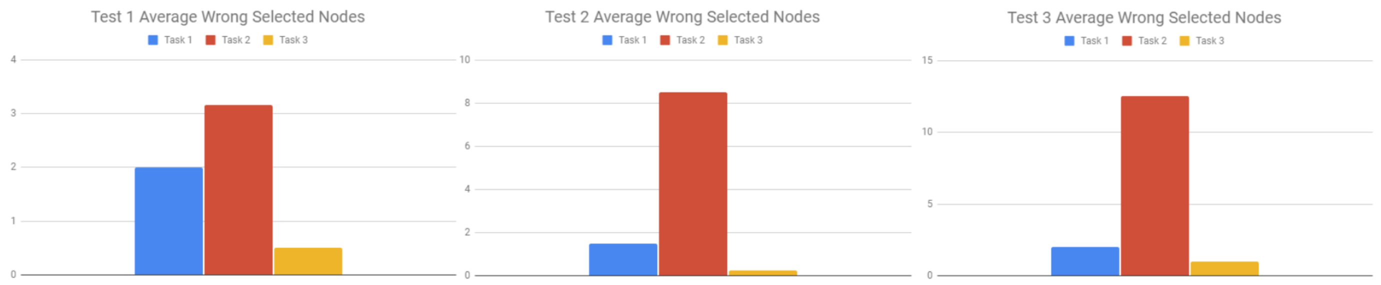

Fig. 7. Average wrong selected nodes per task. On the left we have the correspondent times for Test 1 (see Table 5.1)

efficiency. We assume that the results obtained in this form are because of the enjoyment that the users had at using a *Virtual Reality* application.

### 5.7.4 NASA TLX results

#### Keyboard/mouse + Monitor

In this configuration users displayed overall the most confidence in the system, and reported less physical demand. This is probably due to the familiarity of working with keyboard, mouse and monitor. We notice some users tendency to look at the physical inputs which also helped with the learning process and diminished feelings of frustration.

#### Controllers + Oculus Quest

Using the Oculus and Controllers users where standing up and reported average physical demand despite having to rotate and point with their arms. Users also reported a low to average level of frustration, the controls despite being new to most users proved fairly intuitive, probability resulting from the button layout and the fact that they can see the controller in the virtual world.

#### Keyboard/mouse + Oculus Quest

People generally reported medium to high values of mental and physical demand, this probably stems from the fact that users tended to use a lot of head movement rather than minimizing it by better using the navigation controls. In this configuration users reported the most confusion and insecure in their abilities. The users who reported higher levels have frustration tended to have an hard time finding the proper keys on the keyboard without having the ability to look at them. This difficulty lead to a higher use of head movement leading to more physical demand and sometimes resulting in nausea.

### 5.7.5 Overall Configuration preference

Preference between configuration proved independent of the tasks being made, this is probably due to the overwhelming differences between configurations which overshadow the influence the type of task as on performance. 90% of users preferred the keyboard, mouse and monitor configuration. Reasons ranged from it being the fastest to use, easiest to learn or simply because it didn't cause motion sickness. The least favorite configuration was the keyboard, mouse and Oculus, with reasons such as it being motion sickness inducing, having to rotate the neck a lot and having a difficult time finding the keys for the movement controls.

## 6 Discussion and Future Work

### 6.1 Discussion

With this project we set out to find which is the best way to visualize and navigate through 3D graphs. This turns out to be useful to use the most suitable configuration for specific tasks.

Deciding a dataset was surprisingly fast and we were quite happy to successfully integrate real information to our system and to use it for the task testing.
When thinking on how to display the 3D graph, we quickly resorted to a force-graph, which turned out to not be the most friendly to use, since graphs had to be created fast and had to stay the same from test run to test run. Despite this challenge, we managed to create a 3D graph that answered all our needs.
Integrating VR turned out to be a bigger challenge than expected, but the group was quite resourceful on working together to overcome every faced challenge.

Unfortunately, despite being able to get a sample to study and analyze, the COVID-19 situation we are currently living gave us a huge challenge on getting people to perform user tests. To keep contacts to a minimum, the group stuck with family members to do the user tests.
Despite differences in configuration we noticed that older users tended to have a tougher time with learning the different control schemes and got motion sickness more easily, while younger users where a lot faster to come to grips with the controls and didn't get nausea as easily.

### 6.2 Future Work

Despite being happy with the result of the project on the final delivery, the group recognizes there are things that could be further improved, implemented and tested:

- Extend the user-tests to a bigger sampler in order to further proof or refute the conclusions we made;

- Implement other configurations such as: Gaming controller + monitor, VR headset with hand-recognition;
- Further implement the system so the application is not computer depending, which means not having the need to test while running Unity and only needing to run an .exe or .apk from the computer or android device, from which the user would be able to select the desired configuration
- Being able to change the reason for two nodes to connect to each other on runtime. For example, changing from same *Genre* to the same year on the *Release Date*;
- Change the spheres 3D object to something that would allow the user to distinguish movies (nodes) without needing to select them;
- Experiment with techniques to reduce motion sickness, such as reducing the FOV using vignettes when the user is moving in VR.

## 7 Conclusion

According to our results, the *Virtual Reality* configuration is not the best to deal with 3D graphs, while the traditional methods (keyboard + mouse), are. This concludes that our first question ”Is VR the best setup to get information from 3D graphs (time to complete tasks and errors per task)?” is false. This might be because the tests were mainly made by people that are not experienced in *VR* technologies, and therefore might impact the results. Despite VR being more immersive there have to be some major changes with how we take advantage of the controls and the 3D space, in order to justify the jump to VR which will always require more mental and physical effort to use.

Our second question was ”Is the two-handed flying the best way to navigate in 3D graphs using a VR headset?” we were surprised this technique was close to not used. After being inquired about it, the general answer was that the joysticks seemed more precise and simple to use.

For the third question ”Is VR the most enjoyable way to work with 3D graphs?” we concluded that this is in fact true. We observed that even with the worst performance times, it was the configuration with the best results on *SUS*, that evaluates applications based on users opinions.

We found that high amounts of head movement and high speed movement in VR cause motion sickness in practically all users but had a bigger effect in older ones. This is a major problem with most VR applications, specially if they involve motion and certain developments in this are have to be made to improve the user experience.

As we referred in the Future work section **??** there are several improvements in multiple areas that can be made to improve navigation and manipulation of 3D graphs in VR

## Acknowledgments

The authors wish to thank professor Augusto Esteves for the continuous support along the semester, professor Joaquim Jorge for inspiring us to be creative and Ken Pfeuffer and Soraia Paulo for the feedback on our project pitch.

## References

[1] A. Drogemuller, A. Cunningham, J. Walsh, B. H. Thomas, M. Cordeil, and W. Ross. Examining virtual reality navigation techniques for 3d network visualisations. *Journal of Computer Languages*, 56:100937, 2020.
[2] N. e-Handbook of Statistical Methods. Latin square and related designs.
[3] U. Erra, D. Malandrino, and L. Pepe. Virtual reality interfaces for interacting with three-dimensional graphs. *International Journal of Human–Computer Interaction*, 35(1):75–88, 2019.
[4] S. G. Kobourov. Force-directed drawing algorithms.
[5] C. Nolet. Quick outline.