Context Project - HiVR Final Report

Adriaan de Vos - addevos - 4422643
Boris Schrijver - brschrijver - 4315332
Carlos Brunal - cbrunal - 4002725
Leon Hoek - ljhoek - 4021606
Wim de With - wdewith - 4295277

June 23, 2016

Abstract

This document reports the activities and results by HiVR for the Health Informatics context project. It focuses on lessons learned and identifies the areas of improvement during the development process, integration and user-tests for the purpose of developing a map that represents a virtual world where patients receive exposure therapy to overcome anxieties.[1].

Contents

1 Introduction

The context project is a 10 week project carried out by 5 students and guided by an agile software development process, called SCRUM, alongside a set of software engineering practices. The project was issued by the company CleVR.

CleVR aims to help people who suffer from phobias or social anxieties. They achieve this by exposing the patients directly to their fears, effectively confronting them with these stressful situations in a virtual reality. Because Virtual reality is both lifelike and a controlled environment, risks such as physical harm or budget limitations such as recreating live scenario are greatly reduced. Furthermore, it makes patients more inclined to seek help. The patients don't have to face their fears in real life, and instead can stand in a tightly controlled area under professional supervision which allows them to stop the experience at any moment. One big advantage of a controlled environment is that there are no unexpected triggers and that the therapist can change the environment exactly to what the patient needs.

What makes CleVR different? Unlike conventional existing therapy sessions that only allow the patient to watch, CleVR focuses on making the virtual world interactive. This means that the user is more immersed in the environment and that the therapist can change the environment to match the patient. The patient is following the therapist's instructions but the therapist needs more information than just the head view of the patient. The therapist also needs an easy way of controlling the environment without technical knowledge. This is why we created a new solution that allows following and managing the patient at the same time: HiVR control panel.
The goal of the project is to make a intuitive, clear visual overview in the form of a top-down map that presents the virtual 3D environment in real-time to the therapist, making it more simple for the therapist to see what is happening around the patient in the virtual world and therefore making it easier to plan scenarios and anticipating the patient's reactions.

The representatives from CLeVR gave us the following guidelines for what they considered to be a successful product:

- It is important that the usage of the product doesn't break immersion for the patient in the virtual world. It would be unacceptable if unplanned unrealistic scenarios would take place in the virtual world as this might break the trust of the patients in the competency of the therapists and the validity of the treatment. Unplanned, unrealistic scenarios may also include the visible effects of lag and crashing of the 3D environment.

- The performance of the component that runs on the same system as Unity should try to limit the performance impact to 1 ms per frame or less. Therefore a large part of the resources should be spent on ensuring code quality in order to limit the chances that the code in this project will negatively impact the Unity environment.

- The focus of the project must lay on quality, not quantity. It is better to have less features that are fully tested than more features that have latent issues. The usage of Unit tests are a must.

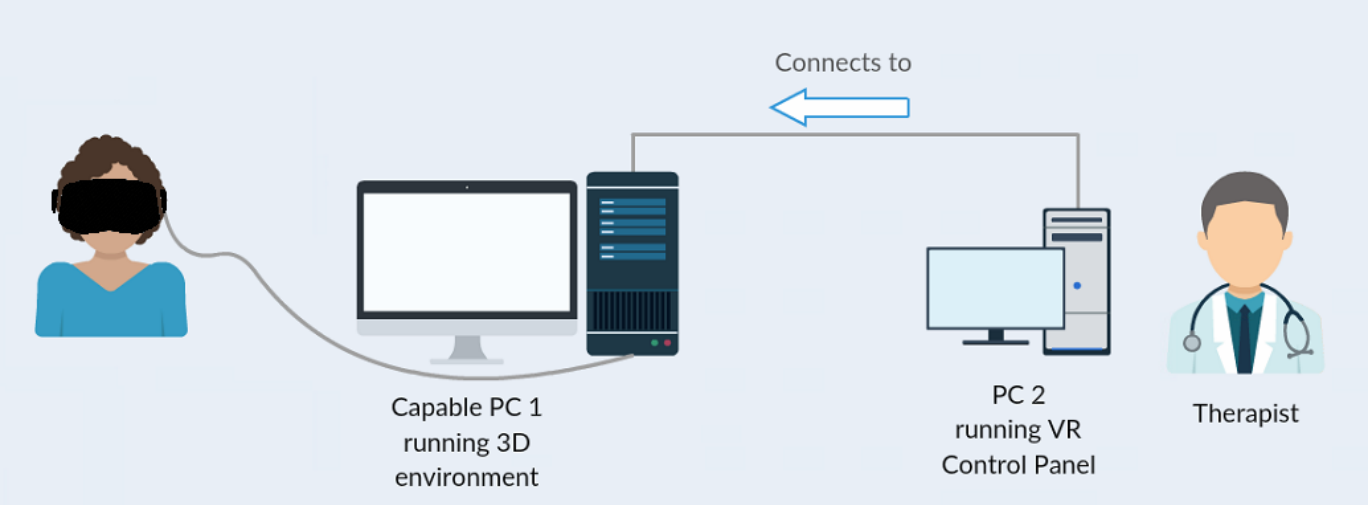
- The project is to be written in the language C#, using the framework WPF to display the map.
- The functionalities of the map are described in three steps. The first step should be completed before the second step and the second step before the third step.
  - Step 1: Display a static map. This step encapsulates retrieving the static objects from Unity (buildings, cars, trees, gardens, televisions, benches) and display them top down with correct size, orientation and location in a simple clear manner. Whenever some of these objects are changed in the virtual world it must be reflected in the map the next time it loads.
  - Step 2a: Display a dynamic map which updates realtime the positions of moving objects on the map. The moving progression of the moving objects should be displayed smoothly, without visible lag or stuttering.
  - Step 2b: The portion of the map that is displayed should be able to zoom in and center on the immediate surroundings of the patient's location in the virtual world. The reason for this requirement is that the actors and objects that are most influential to the patient's experience are generally closer to the location of the patient.
  - Step 3: Make the dynamic map interactive. This part is split in two parts. The first part is enabling the therapist to click on all the elements on the map and let them choose an action for the object to follow via a user-friendly, intuitive interface. The second part is sending this command back to Unity and actually making this happen in the virtual world.

This report describes our work and techniques used during the project to deliver the required system incrementally, generating feedback from stockholders and instructors, and feeding this back into every iteration. Additionally, it illustrates the architectural and organizational set-up of our product and highlights the features and shortcomings that will be encountered with use of the product.

2 Product overview

Our product is to be used between 2 computers. Both computers are intended for the therapists to use, and both have a specific function. The first screen shows the therapist the Unity world trough the patient's eyes and the second one will show our finished HiVR control panel.

Figure 1: Product Setup



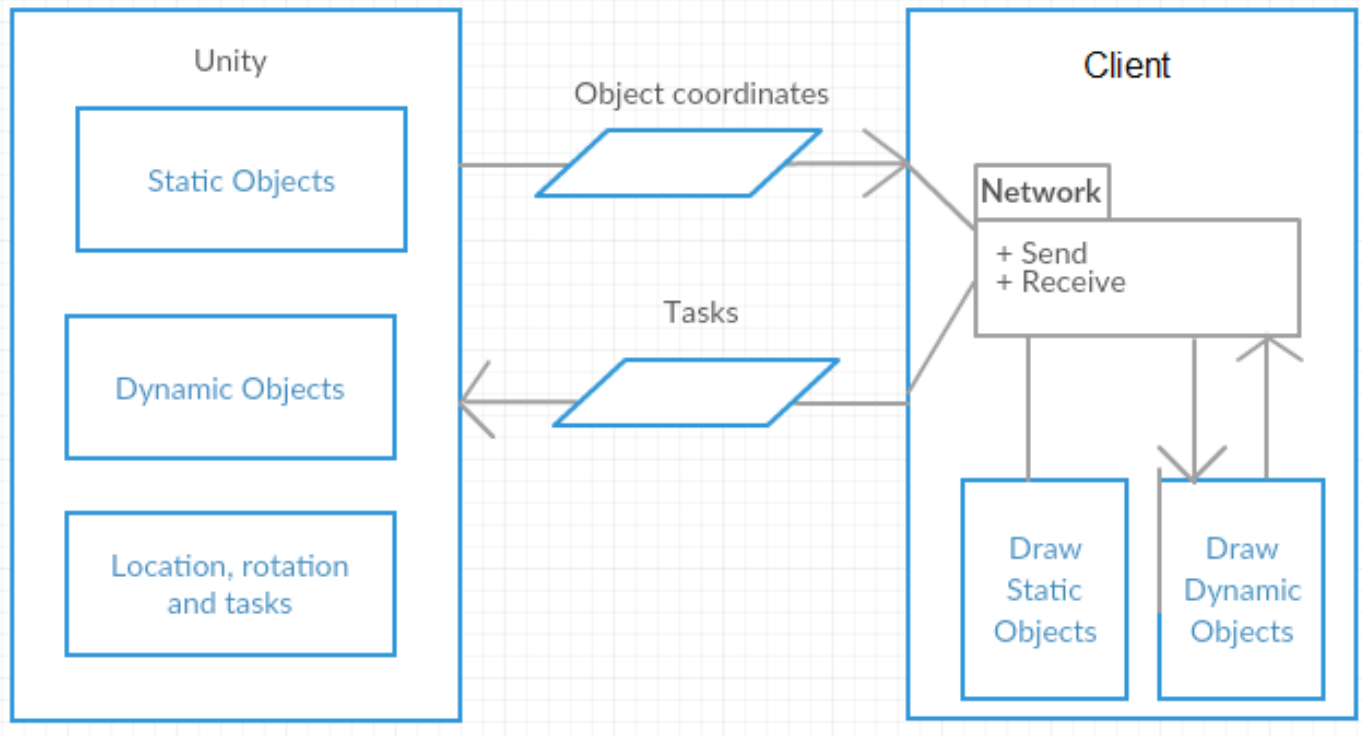
A typical office setup for our software.

The way our software works is as follows:

- The first step is making a local network that can communicate with the unity world. This network allows us to connect to the virtual environment and send and read data from it. We have created a connection window for this. See the GUI in Figure 3 below.
- We then build a library of objects both static (benches, trees, cars, buildings) and dynamic (actors and the patient) in our client, this library contains all the items that exists within the Unity world given to us by CleVR.
- After connecting with the Unity environment, we receive the coordinates, rotation and scale from each object. We then appropriately transform the objects to represent the objects in the 3D environment.
- Static objects are drawn first on our canvas, then the dynamic ones. Lastly we keep updating the locations and rotations of the dynamic objects. Most importantly we give the patient a distinctive shape that in turn also shows what direction he or she is looking.

The HiVR control panel is composed of a number of components and services, each integrated and tested individually and as a whole with unit tests and integration tests. From the above items we can show you an architecture scheme.

Figure 2: Product architecture



Communication specification between Unity and HiVR control panel.

An aspect that is missing from our final product is the interactive part. It is currently not possible to send tasks from the HiVR control panel to Unity. More on this later. In the next page we will show you some screenshots of the software in action.

The final product of the HiVR control panel looks like this:

Figure 3: Connection Screen

Figure 4: Unity map (3D environment)

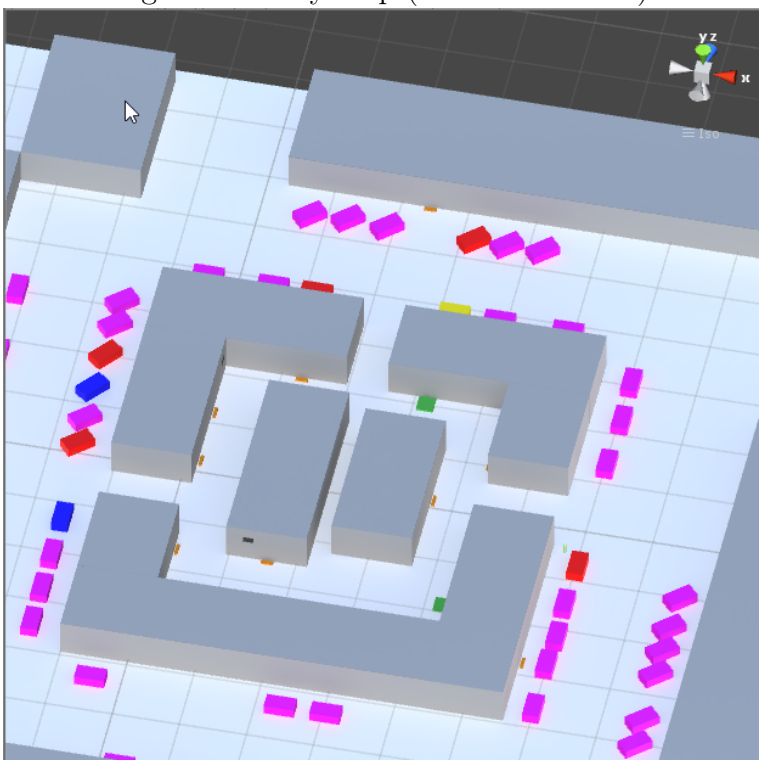
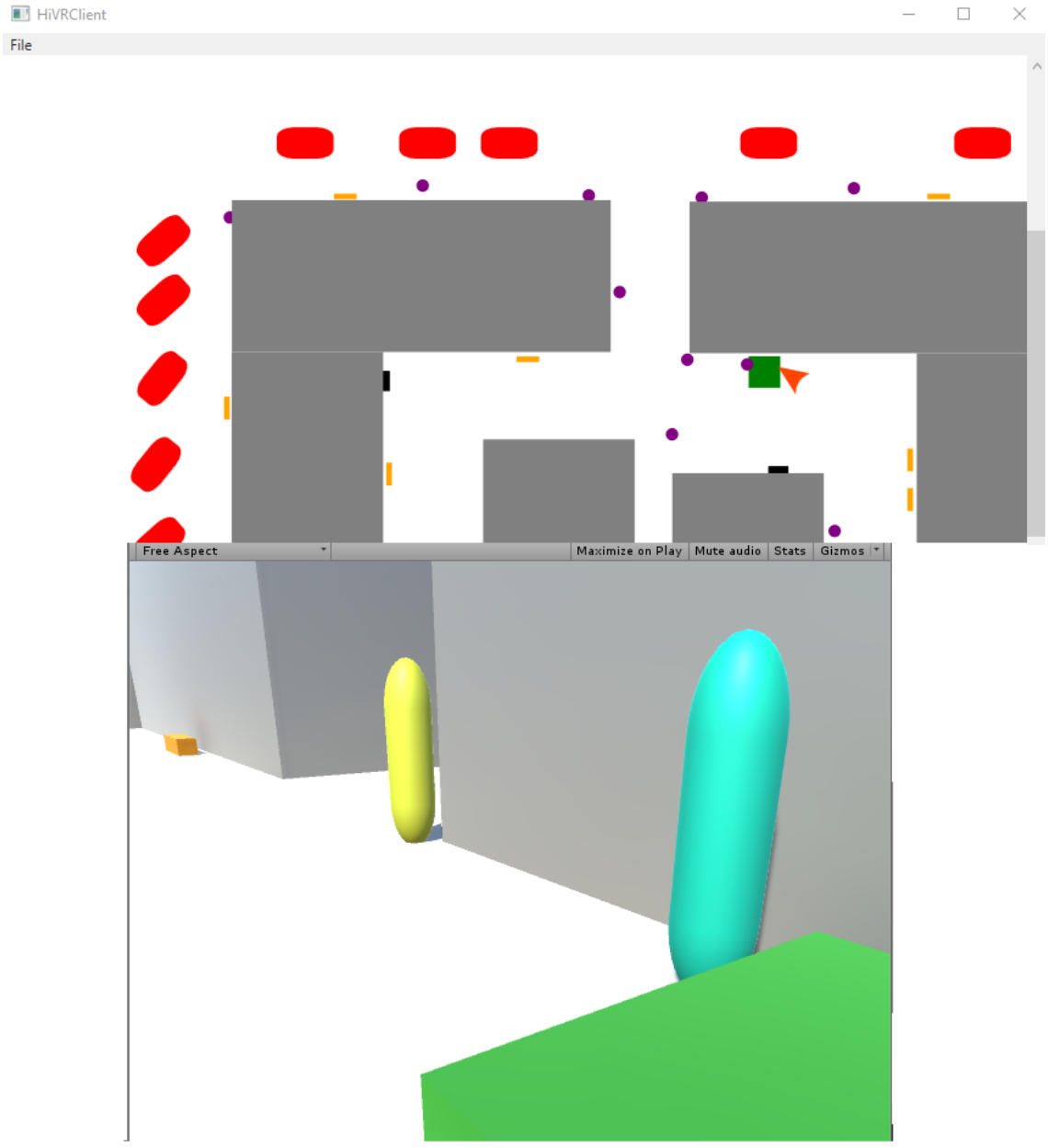


Figure 5: Therapist perspective (top) and Patient perspective (bottom)



On top you can see the map of the HiVR control pannel, in the bottom you can see the Unity world from the patient's point of view. Notice the matching positions of items, actors and tree. Also notice that the perspective of the patient in unity is indicated in the map by and arrow icon.

3 Product and process reflection

3.1 Product reflection

The main requirements from CleVR are described in the introduction and more information is available in the product planning document. Our implementation of the network connection on the Unity side is really efficient. There is no noticable performance impact. We checked this using the Unity profiler, which showed us that the impact of the networking code was barely visible in the profiler and not noticable by the patient. We missed some features, most importantly the interaction with the environment (step 3) but the features we have work pretty well, and our code quality is good. We used StyleCop, which we will write more about in the next section, to ensure code quality and we used NUnit to unit test our product. We implemented some design patterns to make our code more readable and easier to extend, most notably a factory pattern to create objects that need to be drawn, and a observer pattern to notify the view that something has changed.

3.2 Process reflection

3.2.1 Language and framework

Because we worked on behalf of CleVR, we had to use the programming language C# and the accompanying WPF framework. None of us had any experience with this, and it proved to be quite difficult to get up to speed with it. WPF is a rather large framework, and it took us some time to actually understand how we were actually supposed to use it. Especially because we needed to apply the model-view-viewmodel (MVVM) architectural pattern, while we are used to model-view-controller (MVC), which added an extra complexity. In the end, we learned a lot about it, but that knowledge only came relatively close to the end.

3.2.2 Tools

We used C# with WPF, so we needed to use Microsoft Visual Studio as our IDE. Again, none of us was familiar with it, so we also needed to learn how to use it. We got a list of plugins for Visual Studio from CleVR. The most important of these was StyleCop. Unfortunately, we couldn't integrate this in our Continuous Integration, so we had to make sure it we ran it before we pushed code to our repository. This went pretty well, almost no pull requests needed adjusting.
We used Git for version control. Some of us did not really have experience with it apart from the normal commit and push procedure, but others had more experience, and they just explained the more complex parts to them. This worked pretty well, so we did not really encounter problems with it. We used a pull based development model, where we had developed features on separate branches, and then have two other team members review them before merging it into master. This worked pretty well to a certain extent, and we caught some mistakes, but because we divided our tasks, not everyone was familiar with the specific technology that was used in a pull request. Therefore not everyone could always add useful comments.

For continuous integration we set up a Jenkins server with Windows slaves, which would build every pull request and every commit on master. It took us some time to find out how to do this in the first week, but after that it worked pretty well. We added a plugin to publish, build and test the results on GitHub, which made it easier to see the status of a pull request.

3.2.3 SCRUM

In general, creating backlogs and assigning tasks and priorities went pretty well, but we had problems with actually doing everything each week. Especially halfway through the project, we stalled a bit because of a lack of direction, also unfortunately our team leader and SCRUM master fell ill for two weeks. We also underestimated the need to meet in person each day and limited it to twice a week and the rest of the week via Skype, but that turned out not to work at all, as no one took the initiative to actually start a call. In the last weeks, we met every day and worked the whole day together, and we achieved far more than we would have otherwise. So we learned from this that it very important to actually meet because it forces you to work, speeds up the peer review process and makes it much easier to ask questions about components you did not write. We also found it hard to estimate time required for a task because as mentioned before, we were unfamiliar with the technology, and we learned at different rates. For example, an eight hour task could take eight hours programming plus four hours of researching everything and finding the best way of doing it.
Our biggest failure with regards to SCRUM was being too perfectionist. Because we didn't know the framework, we tried to find the best way to implement a task, even keeping in mind tasks for following weeks, to minimize technical debt. This caused us to work on many tasks at once, without actually completing something. In the end, we think this illustrated the need for using SCRUM correctly, because it became extremely inefficient especially combined with our lack of working together.

4 Developed functionalities of the HiVR control panel

4.1 HiVR control panel

Overall we managed to successfully finish 2 out of the 3 phases required, only missing the last part of interactivity in which we instruct tasks to actors and in turn send these tasks to the VR world. The displaying of the static and dynamic map was successfully completed. This means in practice that the user can see what is happening in the virtual world realtime on the map.

4.2 Unity performance

The performance hit on Unity was very minimal. We were quite happy with this. The load stays in the very low percentages, we are talking values closer to 0 than anything else. The performance impact of the netwerk code was barely noticable in the Unity Profiler, and absolutely not noticeable for

the Patient and the Therapist. This was an important requirement from CleVR and we successfully completed it.

4.3 Slight lag in the HiVR control panel

While updating the new position of an actor there is a slight stutter, although very minimal, of the actors moving around in a timed rate. Although not easy this can fortunately be addressed; We would have to make a static image of all the static objects and not render the actors one by one, but do all the changes per frame, overriding the internal function WPF uses. Keep in mind that we are rendering the entire map at any time an object is retrieved now, so it would help if we focused on the area close to the patient because this way we would have to render a lot less. However we still think it is very important to optimize the whole map and not limit it only to areas around the patient. After all managing the virtual world and the patient at the same time requires a bigger overview of the world than just a small area around the patient. With our current code and performance it would be possible to create a minimap in a corner so that the therapist can see the patient up close but still have the big picture.

4.4 Disappointing graphics

A glaring issue in our project was that we only implemented basic shapes as graphics which ended up having a big impact on the experience of users. Fortunately this is not actually a difficult problem to fix. Our code allows for easy refactoring of graphics and shapes. And CleVR would only need to design a high end vector image to fix this problem.

4.5 Networking

The connect screen looks clean and the design is simple but effective. However, it assumes that the user knows about IP-addresses/hostnames and ports, as it asks for that information. This could be streamlined by making the program remember the previous successful connection, so that once the right parameters are provided, someone with less knowledge about these matters can still use the program to connect to Unity. The connection screen automatically fills in the default port and if you run the Unity environment locally you can just use localhost to connect.
The connection process in itself has also turned out to be of good quality. Very stable and with no noticeable data loss. In non of our test have we experience a connection drop. We do have a disconnect option in the file menu of the control panel just in case one would wish to reconnect again.

4.6 Serialized Objects

We are quite happy with our success in converting the Unity objects to XAML. The objects from the Unity environment are stripped from unnecessary information and are then serialized to a byte stream. This effectively converts the Unity objects and their state to a byte stream that can be received and deserialized by the HiVR control panel to recreate the exact same objects that are in the Unity environment. This makes a very lightweight connection protocol that does not use much bandwidth and processing power. This was also done with the help and feedback from CleVR.

5 Interaction Design

5.1 Context inquiry

The students that worked on this context project have not been on site to overview any sessions at a clinic that makes use of CLeVR's products. All our information on the needed requirements comes from stakeholders that do not directly use the software daily. The biggest source of our understanding of the context project and requirements were the periodical meetings with CLeVR in which they explained the main purposes and ways of treatment currently available. We picked up the idea to depict the location of the patient with an arrow to display the viewing direction in one of those meetings. We also learned a lot about how to make the interaction intuitive to the therapist, by making use of a GUI that limits the clicks needed to send a command to two clicks by using a circular menu.

5.2 Claim analysis

We envision our product to be used in one of the following ways:

1. The therapist will start up the application, which is already installed for him/her. He or she will be greeted with a welcome screen containing the connection information. The user enters the appropriate IP-address or it is automatically saved from a previous session. After connecting to the running Unity environment a screen will be shown showing an overview of the map. The therapist will look at the map to see where the patient is in the Virtual Reality world, and may adjust some parts of the treatment on this information. When the treatment is over, the therapist will click the disconnect menu item in the menu and optionally close the application by clicking on the red cross in the top-right corner.

2. The therapist will start up the application, which is already installed for him/her. He or she will be greeted with a welcome screen containing the connection information. The user enters the appropriate IP-address. After connecting to the running Unity environment a screen will be shown showing an overview of the map. The therapist will look at the map to see where the patient is in the virtual world, and may base some parts of the treatment on this information.

However, the therapist does not understand what all the symbols and shapes on the map mean. The therapist grows frustrated with the program and does not use it anymore.

3. The therapist will start up the application, which is already installed for him/her. He or she will be greeted with a welcome screen containing the connection information, without an ip-address already filled in for them. The therapist might not know what to fill in as connection information and fill in non-sense, hoping this might make the application work. The application will give an error message upon processing this non-sense input and lets the user try again.

4. The therapist will start up the application, which is already installed for him/her. He or she will be greeted with a welcome screen containing the connection information. The user enters the appropriate IP-address. After connecting to the running Unity environment a screen will be shown showing an overview of the map. The therapist will zoom in and out with J and K and pan on the map with the scroll-bars. The therapist will look at the map to see where the patient is in the Virtual Reality world, and may adjust some parts of the treatment on this information. Halfway through the treatment the therapist will decide to load another Scene. The therapist may try to restart a scene in Unity and notice that the map doesn't work as expected, as it still shows the old one. The therapist will then need to restart the HiVR control panel.

Three things are most noticeable in the above scenarios. The first is that users might have problems with knowing which IP to connect to. The second is that users might try to connect with a new Unity Scene while still connected to our application. The third is that the user might be put off because the map might not be self-explanatory enough for them.
Of those three it is the most important feature that we upgrade the graphics. This will make the use of the map more intuitive because the depiction will look more realistic and intuitive. It will also give therapist something pretty to look at and make a good impression when the product is sold. A negative impact this feature might bring is that the map may become more tiring to look at. However, we think this negative impact is almost irrelevant.
The second most important feature is making sure that the last successful connection saves the IP-address and the port number, so that after the first use the right connection information is automatically filled in. This will prevent therapist from having to worry about it and manually entering the information everytime an new scene must be loaded. The trade-off is that ip-adresses kan change if the network has DHCP enabled, which automatically makes saving incorrect connection information misleading to the user. We however think implementing this feature is well worth it, because we expect the two devices to be in the same room and often directly connected with static IP adresses.
The last feature to consider is to automatically disconnect whenever it is detected that the Unity server has stopped, and return back to the connection screen. This will make it clearer to the therapist that something has gone wrong with the connection. Then the therapist can take appropriate actions. However, if this automatic disconnect occurs faulty it might become frustrating for the therapist, because he or she must restart the scene to see everything again. All in all, we think this idea is a good idea, but it will still need refinement to work out the details.

5.3 Testing Procedures

We first let the user test the software without explaining what they where looking at, only a reference to a virtual reality control panel. We did this to test the ease of use of the program for an user with standard computer knowledge. Only a small number quickly picked up what the map represented.

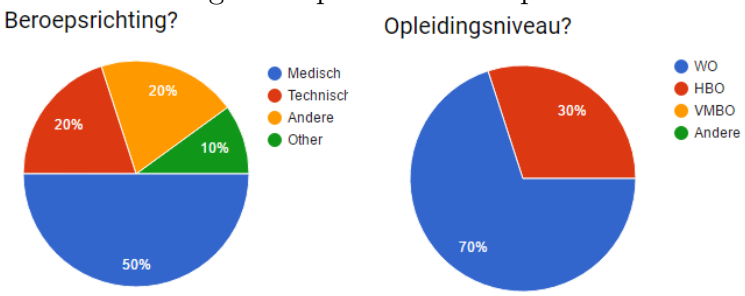
This was done on the last version of the software, and as we only use vector shapes instead of vector images it is of course expected that many users wont understand immediately what they are seeing.
We measured the time that it took them to recognize that the map was a top-down representation of the Unity world. 70% took less than 30 seconds, 30% took longer than that.
We tested multiple people in the medical field, which are the intended users for this software, as well as people in technical fields and other studies. We let more medical users use our software because we believe it may give us better indication and feedback we need for our product.

5.4 Results

We had in total 10 academic testers 50% of these worked in the medical field. 20% technical and 30% other. We got suggestions such as:

- "More options on the panel for doctors, such as a toolbar and icons for connecting disconnecting or making screenshots."
- "Better icons to represent objects on the map, the actors are clear but the rest is unclear what they represent."
- "The point of the software is clear enough, but it is lacking in presentation."

Figure 6: presentation report



As expected our software did not score good in our user-tests when it came to identifying objects:

Figure 7: presentation report

This is mainly because us not being able to refine the presentation of the icons to more relateable objects, since we only have basic shapes representing an object. Interesting enough all of the medical testers did find the smoothness of the animation to be adequate, while our more technical testers where more critical. Technical testers also gave back more constructive criticism.
In conclusion: From our tests it was blatantly clear that for such a project, presentation is very highly regarded. Especially if the software is intended to be used by people who are not always adepts with computers or software.

6 Self Evaluation

6.1 What worked

As discussed in chapter 4 we have a working dynamic map, that is missing interactive capabilities with the Unity world. We feel it is unfortunate that we could not also implement these features. We are however happy with what we do have: We were able to successfully build a network that connect to unity, communicates to our XAMLframework and automatically draws every single object on a canvas. All of those things are done with close to no performance hit on the unity client. We are quite happy as-well with the stability of our network. On the group side of thing, we did feel that we had good synergy and understanding of our responsibilities and capabilities. This of course only came forward when we all worked in unison in the same room, mostly at the end of the project. We are less impressed with ourselves looking back on the earlier stages of the project.

6.2 What did not work

We are missing quite a bit of options in our design, simple things as e.g. a reconnect button, error messages in case things go awry and some basic standards that are to be expected by windows users like right-click context menus. Presentation is one aspect that didn't work to our advantage and can

surely be improved on. Working online and keeping contact from a distance most certainly doesn't work. As a last point: trying to be to perfectionist will cost you valuable time and effort.

7 Conclusion

To conclude our report, we have to say that despite our shortcomings and unexpected difficulties we have had a blast developing this piece of software. They say hindsight is 20-20 and it certainly is. We could certainly improve our project by having better graphic design, and adding interactive objects for example. But also by working better as a team and sticking to the SCRUM formula better.

We have come to learn so much about Unity, C#, WPF, MVVM and many more programming concepts and frameworks that if we had to do anything similar again we are certain that we are up to the task. We have learned the importance of teamwork on a level that we have never experienced before.

We feel the HiVR software we have built is a good foundation for further development, should CleVR ever decide to extend our application. We would not care to say that it is ready to ship, our software acts as proof of concept. It is certainly not good-looking but in the end it is what is in the inside that counts.

References

[1] Anderson, Page L.; Price, Matthew; Edwards, Shannan M.; et al. JOURNAL OF CONSULTING AND CLINICAL PSYCHOLOGY Volume: 81 Issue: 5 Pages: 751-760 Published: OCT 2013

[2] Vive. (n.d.). Retrieved May 04, 2016, from https://www.htcvive.com/eu/

[3] Oculus. (n.d.). Retrieved May 04, 2016, from https://www.oculus.com/en-us/

[4] USC University. (n.d.). Retrieved May 04, 2016, from http://ict.usc.edu/prototypes/pts/

[5] Kim, Y.Y., Kim, E.N., Park,M.J., Park, K.S., Ko, H.D., Kim, H.T.: The application of biosignal feedback for reducing cybersickness from exposure to a virtual environment. Presence 17(1), 1–16 (2008)

8 Glossary

CleVR Main stakeholder in the Health Informatics Context Project. CleVR specializes in creating complete and customized Virtual Reality solutions. CleVR delivers interactive custom built VR software and hardware for a wide range of purposes in the Health Care sector (such as fear of

heights, fear of flying, Psychosis and Social Phobia) and training sector, where the user is able to interact with the computer in a natural and intuitive way. http://clevr.net/. 3

HiVR The name of our Context Project group. Also the name of our software application. http://hivr.nl/. 1

IDE An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. https://www.visualstudio.com/. 9

NUnit NUnit is an open source unit testing framework for Microsoft .NET. http://www.nunit.org/. 9

SCRUM Scrum is an iterative and incremental agile software development framework for managing product development. http://www.scrum.nl/site/Wat-is-Scrum-agile-scrum. 3

StyleCop Recommended by CleVR. StyleCop is an open source static code analysis tool from Microsoft that checks C# code for conformance to StyleCop's recommended coding styles and a subset of Microsoft's .NET Framework Design Guidelines. https://stylecop.codeplex.com/. 9

Unity Unity is a cross-platform game engine developed by Unity Technologies and used to develop video games for PC, consoles, mobile devices and websites. Unity is specialized in creating 3D environments and is increasingly popular under game and VR developers. https://unity3d.com/unity. 5

WPF Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications by Microsoft. https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx. 4, 9