Architecture Design - HiVR

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

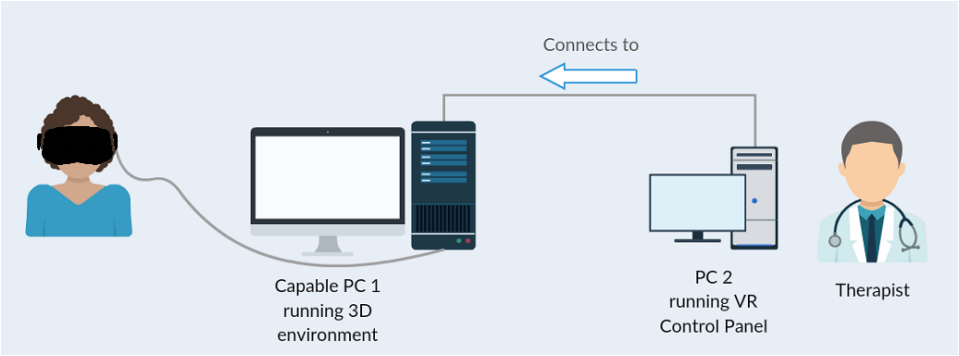
June 16, 2016

Contents

1 Introduction

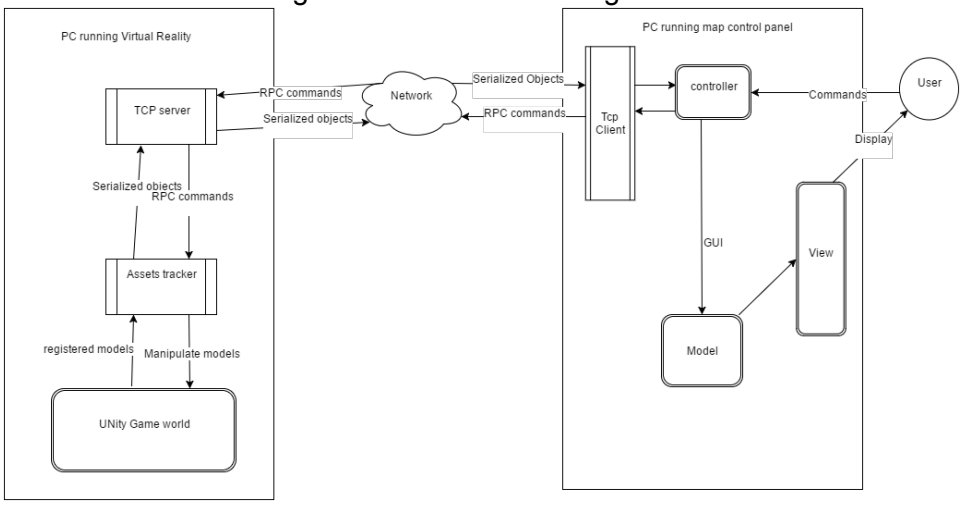
This document provides an overview of the system which will be built during the context project Health Informatics for CleVR.
The software is targeted towards medical specialists that wish to use the CleVR therapy software suite. The main focus is to make an interface as user friendly as possible without limiting treatment options. The interface will provide an overview of where the patient is situated in the VR world and also provide many tools that can aid the therapy sessions to the specialist's liking. figure 1 and two give a better overview of the architecture of our software.

Figure 1: Interface setup



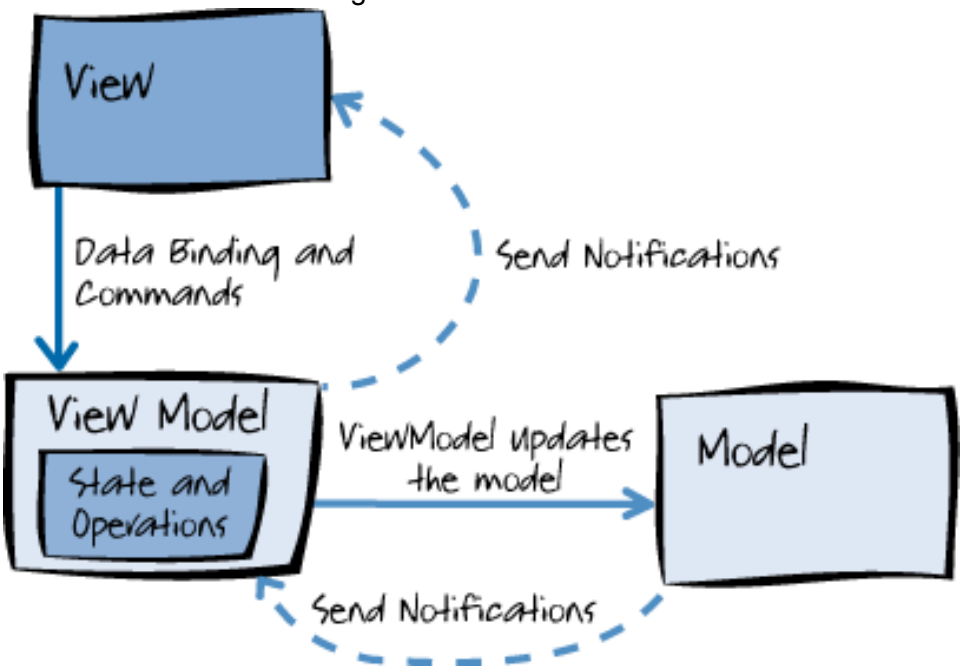
The specialist will be able to both see what the patient is seeing via Computer number 1 and also have a map overview of the VR world in Computer number 2.

Figure 2: Architecture of figure 1.



A more in-dept view of our design implementation.

Figure 3: XAML window



Representation of an XAML window that we use to write WPF with.

The architecture of the system will be explained in the form of high level components, sub-components and sub-systems. This document is a work-in-progress, it will be updated during the course of the project, each sprint. A final deliverable will be presented at the time the code is done.

1.1 Design goals

Throughout the project we will maintain design goals named; ”Performance”, ”Ease of use” and ”Extensibility”. They’ll be explained in each subsection respectively.

1.1.1 Performance

The existing technology runs at an frame rate of 90 frames per second. Which means each frame has just over 11ms to render. We, as the development team, should limit the time needed for each frame to update the map/world as much as possible to keep the program running smoothly. A hard requirement is that we shouldn’t use more then 1ms of processing time each frame. This is of vital importance to the treatment of the patients because any delay in performance may break immersion and cause nausea.

To meet the requirement set above we implemented a tracking system to keep references to all relevant models. By doing this we don’t have to search through the whole Unity scene each time we want an update. We also don’t extract information each frame, instead we use the fixedUpdate() call to fetch information 30 times a frame. That way we are not bound by the frame rate of the VR-device. The information is afterwards being handled by a asynchronous network processor. Another preference is the use of a canvas in WPF since without this the layout manager may not be able handle too much information.

1.1.2 Ease of use

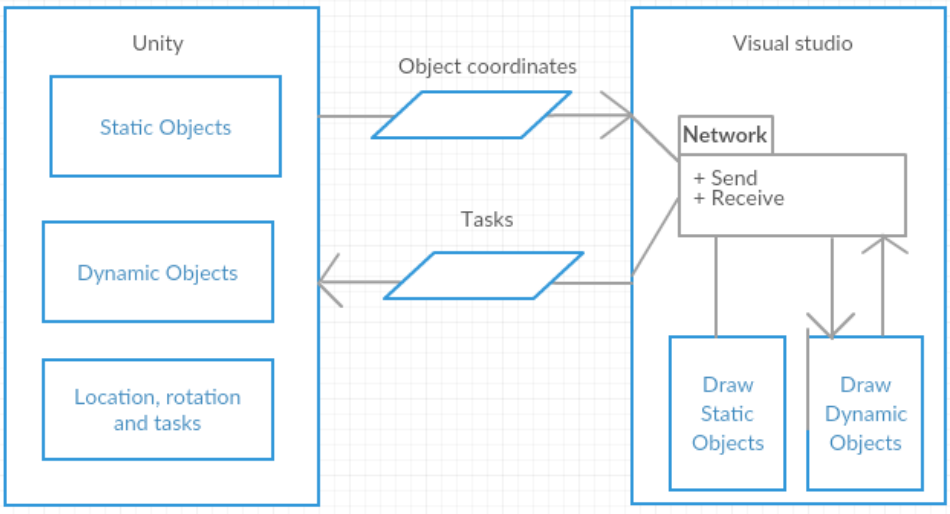
The interactive map should require little maintenance from the therapist during treatment. The focus of the therapist while giving treatment should be on the patient, not troubleshooting with the tools. It is important for the therapist to quickly see up-to-date information on the location of objects and actors in the Unity world, as well as the moods of actors, the field of vision of the patient and more, to quickly determine triggers for the patients. It should

also be quick and simple for the therapist to manipulate the environment according to those triggers.

1.1.3 Extensibility

The software is to be implemented with future growth in consideration. The representatives of CleVR expressed the need for the software to be free of licensed libraries and also that it may be upgrade-able in the future for example: the use of static bitmap images is not preferred, this ensure that the program can be re purposed for other resolutions we are to use vector images instead. In order to handle objects from unity we implemented a Control system in visual studio that read both the coordinates of the objects in unity. This system uses a list known data-templates for each of those objects in unity. This means for each object in unity in the map application there exists a class and a control class. A class to be able to draw it on the canvas, and a control class to implement its behavior or procedures. Please look at the following figure 4.

Figure 4: How the canvas draws objects



The only thing the map is reading from unity is coordinates, the map contains the same database of objects as unity and it scales everything accordingly to the display. From the map tasks are supposed to be sent to unity, which in return we get a reaction by reading coordinates and other necessary data from unity.

2 Introduction

This section discusses the architecture of the system. We will briefly describe what each sub-system does, what the inter-dependencies are and how they work together. Research is currently being conducted and the following section will be updated accordingly.

2.1 Subsystem decomposition

We are planning to build 3 components:

- The graphical interface itself including, layout of the map and inside the static objects like car's buildings benches, also dynamic objects like actors and the patient. Two examples:

Figure 5: GUI map

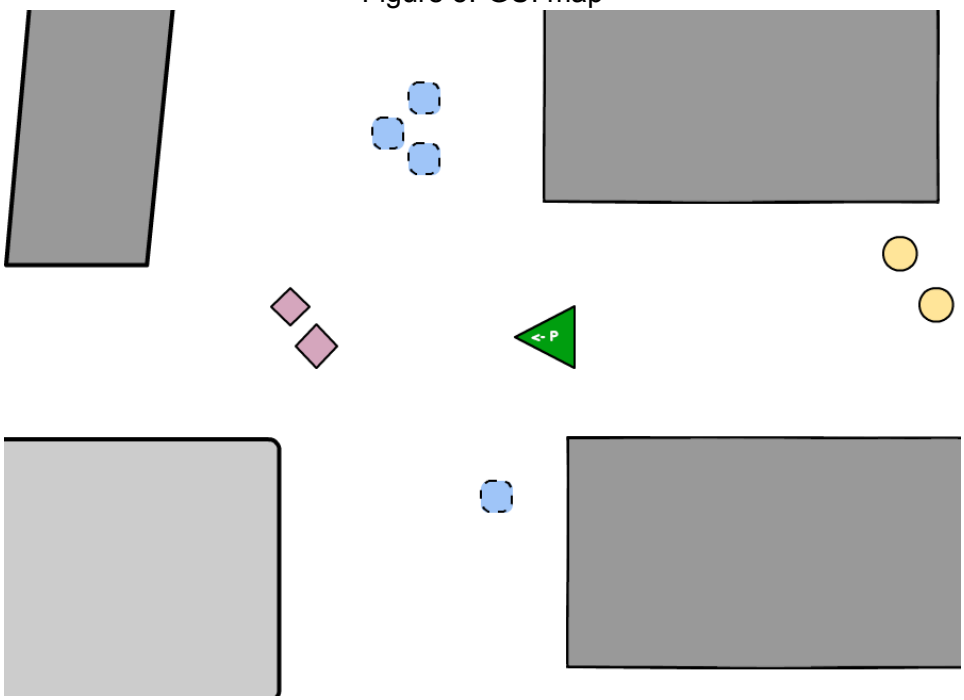
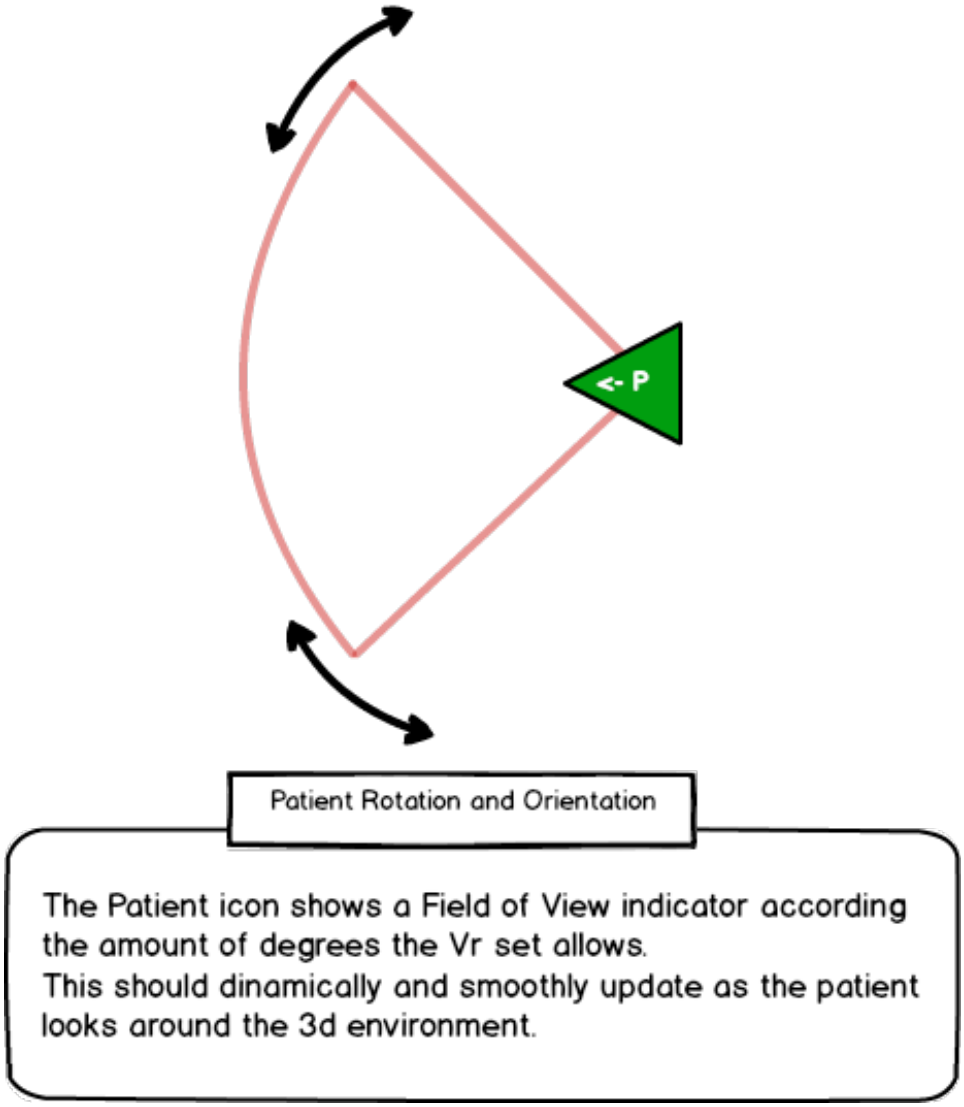


Figure 6: Proposed patient orientation design



- The needed plugin/protocol with Unity to let the GUI communicate with the Unity world.
- The needed network protocol to connect host and client computers

2.2 Hardware/software mapping

The tools we will use are the standard input devices for any computer, namely the keyboard and mouse. An optional hardware that we might implement is touchscreen integration. This is yet to be decided. Furthermore, the system will be running in 2 computers, on for the VR simulation and one to function as control panel for the therapist, these will be connected via a local network.

2.3 Persistent data management (file/ database, database design)

The interact-able map being developed is a viewer for the world defined by Unity. Everything the viewer processes will be given to it at run-time, and as the work being done by Unity is out-of-scope for this context project. This project will therefore not include persistent data management.

2.4 Concurrency

Each frame has just more than 1 ms to render. We should limit the time needed for each frame to update the map/world as much as possible to keep the program running smoothly. We discussed a window of 1 ms to send and receive all our commands. We still have to determine our lag limit, but ideally something low enough that the user itself wont notice.

3 Glossary

CleVR VR development team at Yes!Delft, focused on virtual reality therapy solutions. `http://clevr.net/`. 3

Unity Unity is a game development platform, can be used to make 3D environments. `http://unity3d.com/unity`. 5