@Bristol VR Project Plan

High level description of the system and the needs it satisfies.

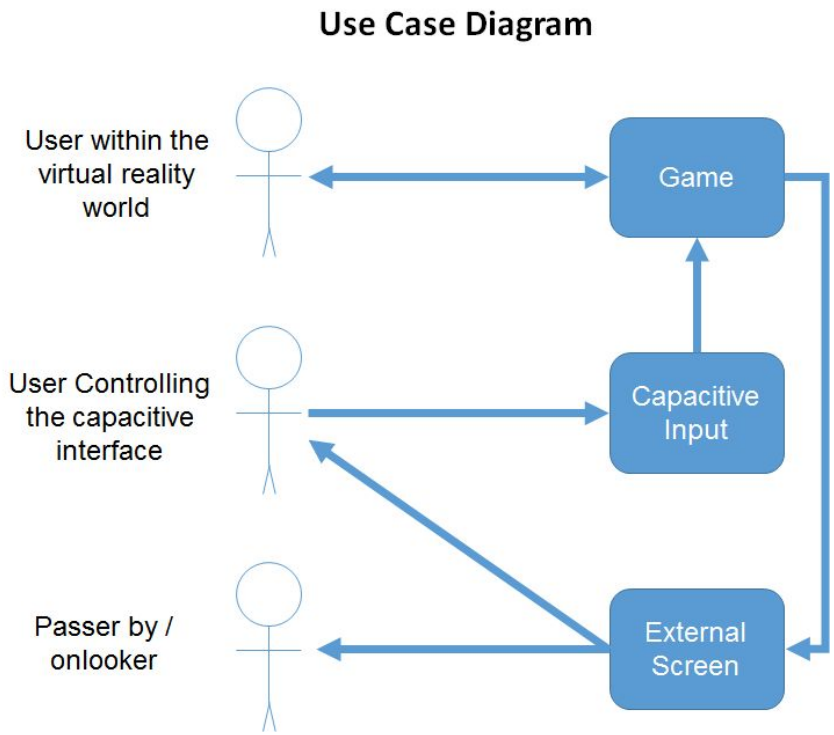
@Bristol have space where they want an interactive exhibit that demonstrates a new technology to both adults and children. Anna, the Creative Director of @Bristol, was excited about our suggestion of exploring virtual reality - using physical and novel methods of interacting with their virtual environment.

They wanted an experience that engages more than one person, to make it suitable for large groups and to entice passing public into the museum space. Anna mentioned that the set up should be both robust and intuitive.

We will create an interactive virtual reality game for Google Cardboard. The users will co-operate with each other within the set-up. One or more users will enter the virtual world by putting on the Google Cardboard, while others interact outside the virtual world through a capacitive input device to either collaborate, compete or aid the navigation of the user(s) in the game. There will be an monitor within view of the capacitive input device to allow the external users or passers by, to see what effect their actions are having on the user, in the game world.

Use Case Diagram

We made the following Use Case Diagram to show the different ways that ours users will react with our system. We explain beneath the image the purpose of the arrows.



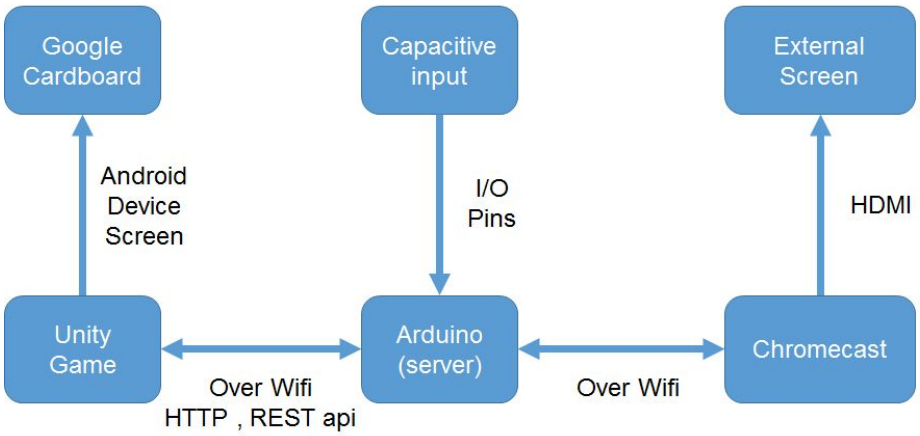
We have three different expected type of user: The user who is using the mobile device and cardboard to enter the game world, the user(s) that are controlling that capacitive input devices and the general public/onlookers that are near the exhibit.

The arrows in the diagram show which users interact with which parts of the system and what output is presented to which users. For example the user controlling the capacitive interface interacts with the game via the capacitive input device while the external screen provides them with the output from the system.

Diagram of Architecture

We then created the following diagram which outlines all the major components that will be used in our system and how they communicate with each other.

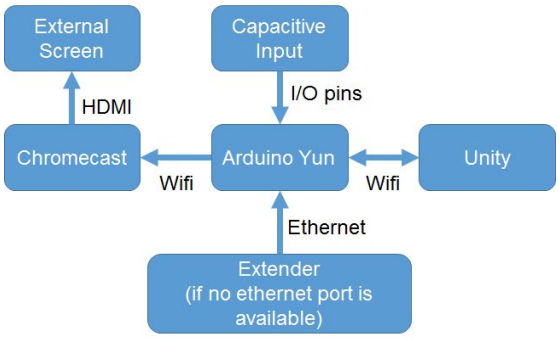
Diagram of Architecture



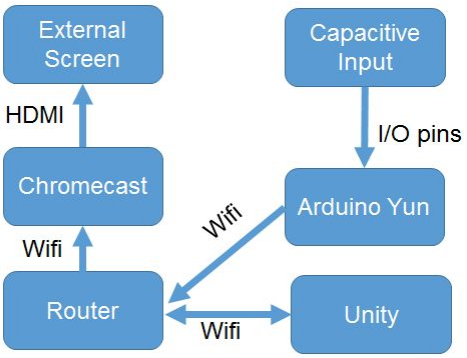
As shown in the diagram above, we have 6 main components in our system. These are the Unity Game (on the android phone), a google cardboard, an arduino Yun (used as a server), chromecast, a capacitive input and an External Screen.

We then discussed how we would specifically allow the game to be shown on the screen whilst still using the arduino as a server to send the capacitive input to the unity game and came up with two ideas.

Idea 1



Idea 2



Idea 1: Our first approach for solving this problem requires the arduino Yun to be set up as a Wifi access point. All devices are connected to the arduino directly using Wifi. Chromecast requires internet connection to function correctly. This will be provided to the network, either directly using an ethernet cable, or using an extender/bridge if no ethernet port is available. The advantage of this approach is that the android phone running the game will have faster access to the data provided by the arduino and thus lower latency. However, this setup could possibly require additional hardware which adds to the cost of the project.

Idea 2: All devices are connected to the same wifi network which should be provided by @Bristol. All communication between Unity, the Arduino Yun and Chromecast will need to pass through @Bristol's router. Although this does not require any additional hardware, it could potentially increase the latency and render the game unplayable.

Languages and Libraries

Unity Scripting Language. Unity provides a host of different languages upon which games can be built. We have considered each in relation to the strengths and weaknesses of the team and concluded that C# fits our goals best.

C# **Pros**: Nearly all documentation has C# examples, close to java/C/C++ which all team members have experience in.
Cons: Verbosity due to being explicitly typed

UnityScript **Pros**: based on javascript which all team members have experience in, most documentation has examples in UnityScript.
Cons: potential for bugs with implicit typing, no generics.

Boo - **Pros**: based on python which some team members have experience in, possibly a more productive language due to its succinctness.
Cons:little to no documentation, team members less familiar with python based languages, potential for bugs with implicit typing.

We have decided to use C for the Arduino server due to the Arduino IDE being built for C and because all team members have strong experience in C from Year 1.

We will use the Google Cardboard SDK as it provides a simple API for VR that is accessible in Unity.
For communication between Unity and the Arduino Yun Server we will use a REST API over HTTP using JSON to output data, as ease of processing and performance are more important to us than readability of output, and XML and HTML are prioritised the other way around.

Tools

Unity Game Engine will allow simple integration with Google Cardboard.
Minimal installation and virtual reality optimisations supplied for us. i.e. motion sickness adjustment, gyroscope smoothing technology and other assorted niceties that make development easier.
Furthermore, most team members have experience with the software.

We will use **Git/Git Large File Storage as well as Trello**.
This will enable the sharing of large unity game projects and trello will provide an intuitive and simplifying method of communication between team members and the team as a whole.

Criteria to measure success

We have explained to the client how we intend to implement a minimum viable product. This framework comes from the lean/agile methodology. We want to develop this and iterate in very small cycles on the interface. Specifically we are working top down - from the VR headset and steadily toward capacitive interaction. That brings me to:

1. When we have at least a MVP that is robust, functional and can be shown and used by the client. We have established this to be a minimal VR interface. Anna, has told us that this would, at the worst case be sufficient.
2. Any iterations beyond that up to a fully fledged capacitive interface implemented as well would be ideal.
3. The above combined with the ability to coordinate and distribute tasks effectively amongst the team with minimal inter-team tension. Should be a robust, structurally elegant model that can be given to @Bristol.

Work Plan/Gantt diagram