Mobile Virtual Reality featuring a six degrees of freedom interaction paradigm in a virtual museum application

M. Papaefthymiou$^{1,2}$, K. Plelis$^{1,2}$, D. Mavromatis$^2$, G. Papagiannakis$^{1,2}$

$^1$Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science
100 N. Plastira Str., 70013, Heraklion, Crete
http://www.ics.forth.gr

$^2$University of Crete, Computer Science Department,
Voutes Campus, 70013, Heraklion, Greece

Technical Report Number 462
23rd December 2015

Keywords: mobile virtual reality, HMD cardboard, virtual museum

Abstract

From the storyteller on a Greek stage to the latest in curved-screen TVs and IMAX theaters, one of the main objectives of education and entertainment has always been related to immersion: putting the audience “inside” the drama, or educational concept transporting them to another time and place and empowering them to visualize this concept from different perspectives. To date, that has been an “over there” experience, leaving a physical gap that reminds our senses that we’re not truly immersed. But that’s beginning to change, thanks to the advent of modern Virtual Reality (VR) and Augmented Reality (AR) technologies, otherwise termed together as Mixed Reality (MR). Just within the last two years, there has been a great leap forward in immersion and VR/AR techniques and h/w enabling technologies. Along the way we’ve discovered that VR/AR is not just for entertainment—gaming and movies, for example—but that it also has broad commercial applications for the likes of virtual travel and very recently smart education on both mobile and desktop devices. Mobile Virtual Reality (VR) is becoming an increasingly “hot” topic nowadays, with the development and introduction of various VR headsets by a few different vendors such as the Google-Cardboard HMD. One of their shortcomings is the absence of traditional navigation and interaction controls. In this work we employ a secondary mobile device for six degrees of freedom interaction (translation and rotation) using the screen as well as the embedded mobile sensors in a virtual museum application.

This work was supported by the FORTH - Institute of Computer Science.
I. INTRODUCTION

VR headsets aim to increase the immersion of the user, attempting to make the borders between the real and the virtual world very thin. This could be used either in gaming, or in other cases, such as navigating a real space that has been modelled and converted into a 3D application, medical or education applications etc.

Google Cardboard is a virtual reality (VR) platform developed by Google for use with a fold-out cardboard mount for a mobile phone. It’s a low cost device which enables everyone to jump inside the VR world, but there is a problem. Cardboard only offers the user the ability to navigate by head tracking mechanisms and to interact with only one magnet trigger, removing the feel of emerging inside the VR world.

Therefore we decided to implement a way to add more functionality to the Google Cardboard by using a secondary device to navigate and interact inside a scene. The secondary device can be used as a controller to point inside the scenery and also as a way to move. In order to achieve that immersion, at least three things are required:

Head-tracking
Head-tracking is essential for VR applications. Initially, we started with 3 degrees of freedom (3DOF), allowing to map real-world head rotation into virtual-world camera rotation. Oculus then improved on that, using IR LED sensors, thus enabling 6DOF head-tracking, which means that also real-world head movements are translated into virtual-world camera movement. Low latency is a key requirement, to prevent negative impact on user experience.

Stereoscopic 3D rendering
The screen is split (vertically) in two halves, each one displaying a different frame. In fact, the screen output is the result of two cameras with different position and angles, aiming to emulate the eyes and the human stereoscopic vision. Those two frames are each projected to the appropriate eye using two lenses. The quality of the display, such as the resolution, response time and color reproduction, all matter to achieve a good result. For example, a low resolution screen reduces the immersion as the user can observe what is called the “screen-door” effect, the distinctive borders of each pixel. Another worse example, is a display with a slow response time or a low motion resolution. This will result in motion blur as the user navigates the virtual space or even a noticeable latency between the user’s movements and the movement in the virtual space. This takes away from the experience and at worst, makes the user feel motion sickness.

Intuitive, seamless controls
Another very important aspect of VR, which we are going to discuss more in the following sections.

II. OUR MOBILE VR METHODOLOGY

A. Mobile Virtual Reality with Google Cardboard

Google Cardboard is a cheap alternative to push people and developers into VR. It’s a hard-paper mount for mobile phones of certain size that utilizes the device screen and sensors to perform stereoscopic 3D rendering (using the built-in lenses) and 3DOF head-tracking. As a result, there’s no need to use any other devices or computers. Cardboard also offers a magnetic trigger, which works as a button. The Google Cardboard API tirelessly implements stereoscopic 3D rendering and head-tracking, while also providing helpful event listeners (e.g. for the magnetic trigger). It is also available as a Unity plugin.

B. The main problem addressed

An important and open topic regarding VR is the controls and controllers. Controls are a major part of VR, as they also alter the perceived level of immersion. Currently, a few approaches are being used:
Head-tracking only
Limits the user to being a spectator, or perform minor interaction with the virtual world

Keyboard
A lot of buttons but still not ideal, as it limits players from moving in the real-world space

Gamepad
Better than a keyboard, as they are wireless and easy to use, but still not designed with VR in mind.

Custom controllers
Such as “Motion Leap VR”, “Oculus Touch” etc. which are built with VR in mind, and provide VR-oriented controls. Currently is one of the best solutions for VR interaction.

Cardboard suffers in that regard more than the PC-powered VR headsets, as the mobile device is inside the Cardboard, making it impossible to use the touchscreen or any hardware buttons. As a result, the only inputs available are the head-tracking and the magnetic trigger.

III. OUR MOBILE VR CONTROLLERS
We considered the following controllers for our Cardboard-based mobile VR application:

RGB Camera
By processing the camera images, we can potentially detect hand gestures or movement in real-world space and act accordingly. We argue that this is not easy to implement, not accurate enough, has a high latency due to the processing needed and also because of that will have a significant impact on the framerate of the stereoscopic 3D rendering, as mobile devices are still magnitudes slower than high-end computers.

Keyboard or Gamepad
As we already mentioned, those controllers are limiting and not really designed for VR apps. In addition, mobile devices usually do not offer keyboard or gamepad support using the manufacturer’s firmware, as the required drivers are not installed and the USB port is not setup to work as a USB host. Elevated rights (root) and possibly custom firmware may be needed to enable these.

Secondary, controller mobile device
We chose to use a second mobile device, which will be the controller. This will be easier to implement and it will allow us to design the controller closely to the needs of our app. Modern mobile devices also offer a wide range of sensors which can be used, along with the touch-screen controls. The Cardboard device will communicate with the controller device using a wireless networking protocol, which means the latency should be well within acceptable values. Finally, this approach has no impact on the performance of the device that does the stereoscopic 3D rendering.
IV. IMPLEMENTATION

Our implementation is based on a typical client-server model. The controller device is the server, while the Cardboard device is the client. Both devices run Java code that uses the native Android APIs, with one difference. The controller device runs a proper native Android app, while the Cardboard device invokes the Java code via Unity C# scripts through its Java Native Interface (JNI) helper methods. That means our Android code is injected in the Unity app as a native plugin. While JNI calls are more expensive compared to simple calls, they are not at all reducing the rendering performance.

We had 2 options when deciding which Wireless protocol to use, Bluetooth or WiFi. We chose WiFi, as it enables connections with much higher throughput available, lower latency and longer range. In addition, all mobile devices have WiFi support nowadays.

Normally, WiFi networks require a “hotspot” to connect to in order to communicate, meaning the need for extra equipment. We overcame this hurdle, by using WiFi Direct, which creates a ad-hoc network, allowing direct peer-to-peer (P2P) connectivity.

When the controller app is ran, a WiFi P2P service is registered with a specific name. Then, the Unity app (Cardboard device) performs service discovery and attempts to connect to any service discovered that matches the expected name. Finally, a one-time prompt is shown to allow the connection, and the API provides the IP address of the WiFi P2P group owner, enabling us to setup a typical TCP socket.

We offer 3 ways to navigate in the virtual-space:

- **Simple button**
  A simple button that moves forward, relative to where the camera is looking, towards the Z-axis, while it’s being held.

- **Joystick**
  On-screen analog joystick that enables 2-axis (X and Z) navigation.

- **Step sensor**
Implemented as proof-of-concept, while the user is physically walking, each step detected is translated to 0.5 seconds of forward Z-axis movement (similar to the button). The accuracy is not perfect and the user needs to be aware of the surroundings.

We also use the volume up and down buttons for some extra functionality.

Lastly, we utilize the Rotation Vector Sensor data. This is in fact a sensor fusion, meaning a virtual sensor that combines data from multiple real sensors (such as the gyroscope, accelerometer, magnetic sensor etc.) in order to provide more meaningful and accurate results. The Rotation Vector Sensor provides the absolute rotation of the device in quaternions. The axis defined for Android are different to Unity, as well as the 0 points. We also calculate and transmit the Euler values.

The Cardboard SDK inside Unity gives us a prefab which contains the Stereo Controller. By doing some modifications on the default script (controller.cs) we added a secondary helping camera on the head of the Player which will use as a pointer inside the scenery.

This second camera, which is disabled, points on a direction based of the rotation vector on the secondary device. We use a disabled camera to take advantage of the Unity function ScreenPointToRay which let us draw a ray from the camera to the direction pointing and find the first object we collide with by doing a simple ray cast.

Ray cast lets us find all the elements that the laser pointer intersects and by calculating the distance of each collider we can find the closest one. On the closest collider we then draw the laser pointer which in our case is a simple red sphere which activates the collision detection of the collider.

A. Implementation of Collision Detection

Each model of the virtual museum uses a custom script which changes the color of the mesh when a user hovers the model with the laser pointer. The script also contains a string which corresponds to the description of the model that we are going to use on the preview mode and listen to by the TextToSpeech. We used the functions OnCollisionEnter and OnCollisionExit and check if the collider is the laser pointer. If it is true we highlight the model and enable it to move to the preview mode if the user uses the magnet trigger.

B. Implementation of Movement

The secondary device can send command to move inside the scenery on a specific direction or use the step sensor to move forward. To translate the commands to movement we added inside the default script of the Cardboard SDK a simple function that calls Java code to get a float array from the WiFi P2P connection. The data are then used inside the script to translate the position of the player inside the VR environment.

C. Implementation of Preview Mode

We decided to add an additional feature of letting the user inspect the model better and read/hear some description about it.

If the player hovers on a model and uses the magnet trigger, the player and the model are moved to what we call Preview Mode.

When you enter the Preview Mode a TextToSpeech starts to explain some facts about the model and also you have a text which you can scroll through by using the volume buttons of the controller device.

Also, you can rotate around the y-axis of the model by rotating the secondary device to see more details around the model.
D. Challenges addressed

When using quaternions, we came across many problems because the axis on the phone are different than the ones in the Unity platform. Therefore, the quaternions were incorrect and we had to use Euler Angles despite the problems that may cause.

Mobile devices have many restrictions when it comes to lighting and some lights are not working properly when inside the program. In preview mode the lighting is not working correctly on the mobile device.

Rotation vectors generated from the gyroscope of the controller device may differ between different devices so we had to implement a dynamic way to compute-recalibrate the pointer.

Not enough free models to use inside the museum. We ended up only using a free version of the Statue of Liberty and a custom made painting of Guernica made in SketchUp.

No centered-pivots for the models causing rotation at preview mode nearly impossible. Had to improvise and making an empty game object set as parent to the center of the object we wanted to change the pivot. This way changing the position of the parent was moving the child by the pivot. This can cause nausea.

V. Results

The virtual museum was designed inside Google SketchUp and imported inside Unity. Everything except Statue of Liberty had to be made manually and design it inside Unity with the default assets given. The following figures illustrates are results and mobile VR interaction metaphor with a secondary device used as trackpad (three degrees of freedom for translation via the mobile device screen) as well as motion controllers (three degrees of freedom rotation via the device’s gyroscope and accelerometer sensors):
We ended up with a simple virtual museum proof of concept, which only for now holds inside two exhibits (Statue of Liberty and Guernica painting) and a custom made room called preview room which is only visible when the Player wants to inspect an exhibit further and activate the scrolling text and text to speech. We wanted to design a
simple virtual museum with some exhibits and let the player navigate inside freely with the use of the secondary device.

The Player has the ability to use secondary device’s rotation vector as a laser pointer to choose an exhibit and move it to another room called preview room to inspect it further. Inside the preview room the player has the ability to listen about more facts about the exhibit and also read from a scrolling text.

In this project we used the Cardboard SDK and Unity. My colleague established a communication between two devices and designed a simple GUI. The purpose of the secondary device is to send some specific data which we are using inside Unity.

The data that we are sending include a simple move button (on/off) to move the Player inside the scene, rotation vector of the device to know where the Player points with the secondary device and when a volume button is pressed to scroll through text inside the preview mode.

![Figure 3. Our mobile VR virtual museum project inside Unity.](image)

A. 3D models and props

The virtual museum was designed inside Google SketchUp and imported inside Unity.

We ended up with a simple virtual museum-environment which only for now holds inside two exhibits (Statue of Liberty and Guernica painting) and a custom made room called preview room which is only visible when the Player wants to inspect an exhibit further and activate the scrolling text and text to speech.

VI. CONCLUSIONS AND FUTURE WORK

We showed an alternative to expensive custom VR controllers, that is utilizing a second mobile device and building a software controller. This concept can be extended to become a generic software controller for VR apps or customized for the specific needs of each app. The big range of sensors in modern mobile devices makes this approach very appealing, enabling unusual but intuitive control schemes. As part of our future work, we are currently working on being able to retrieve the 3D models needed in the virtual museum application from online open repositories and metaverses, such as European or High Fidelity. Lastly, another interesting aspect would be in creating a network of multiple secondary devices, providing and combining data from them.
VII. REFERENCES

10. High Fidelity: https://metaverse.highfidelity.com