VR Jedi Knight

Citizens For Ted Cruz
Zoë Naidoo (uznaidoo@uchicago.edu)
Zakir Gowani (zgowani@uchicago.edu)
Andoni Garcia (andoni@uchicago.edu)
Inspiration
The Problem

Google Cardboard is Limited

Although Google Cardboard provides a framework for thousands of uses utilizing its technology, as it currently stands, users can only interact with the device in 2 ways: by moving their head (and body) or by pressing the single button.
Solution

Enhancing the VR Experience utilizing a second mobile phone

Networking a second mobile phone for input opens up a wider range of possibilities for the Google Cardboard framework
VR Jedi Knight

This game allows for players to utilize a second mobile phone networked to their Google Cardboard device, in order to wield lightsabers. Players will rotate and slash their saber using the phone held in their hand to destroy enemy orbs that appear and shoot at the player in their Virtual Reality headset.
What’s Been Done?

- There are currently Bluetooth Controllers on the market for games that require hands-free play (e.g. Zombiestan).
- Many people have used external phones to represent the camera view of their VR/AR game, however, few have used an external phone as a game object.
- There appear to be limited Virtual Reality Lightsaber games in the app store:
  - LightSaber Escape enables the viewer to wield a lightsaber tool on their phone, but is limited to desktop view on Google Chrome (not an immersive VR experience).
  - Disney also released a Virtual Reality Star Wars game, however this game appears to be more of an “exploration” game than a fighting game and does not involve an external sensor.
How Did We Innovate?

We brought a second device into the VR/AR arena by networking it to the Google Cardboard viewer. The player uses this device exclusively to interact with in-game objects and interface with the game in ways that were not previously possible.
How It Works: Networking

One device chooses to become the server -- the Google Cardboard viewer by default. Then, the device dynamically displays its IP/Port or GUID identifier.

The second device becomes the client -- the lightsaber by default, and upon the user entering a valid server identifier, creates a connection.

From there, both phones are ‘paired’ and share a reliable connection.

The client continually sends its updated gyroscope input to the server via Remote Procedure Calls.

The server then renders this information and updates the game state.
How It Works: Lightsaber Movement

We initially tried to use accelerometer data, but found that it measured acceleration based on gravity. This made it terribly difficult to reliably trace a device’s movement.

As such, we instead resorted to using the device’s gyroscopic attitude. This data was much cleaner to work with in terms of tracing movement, but came in a Quaternion system rather than the more intuitive Vector3 coordinate system.

Because Quaternion systems rely only on rotations, this implicitly reduced a lot of visual lag and the jittery nature of accelerometer data, but was very tricky to manipulate to give accurate 3D-space renderings.

Ultimately we had the client simply send its exact attitude data, and perform the computation and game state rendering on the server.
How It Works: Enemy Bots

We created the enemy bots to randomly appear throughout a region that was accessible by a stationary player only using their lightsaber.

These bots always spawn aimed at the player, and proceed to shoot at him on a fixed interval.

As the player survives longer and longer, the bots shooting speed quickens, decreasing the player’s health faster!

Finally, when a player runs out of health, the game is over. He receives a score equal to the number of bots he destroyed before he ran out of health.
Difficulties

The Quaternion coordinate system was incredibly difficult to work with, as it did not correspond well at all to a 3D spatial rendering. Manipulating rotations and movements were tough and subpar.

Connecting the lightsaber physics with the 3D avatar’s movement was hard. The movements do not appear natural in accordance with human musculo-skeletal structure (i.e. the player’s arm does not appear to be dragged by the saber and the arm is often contorted in unnatural shapes).

Networking in unity was not the easiest endeavour. It is built for multi-client games, and not ideally suited for client-server games, where the client is talking to the server rather than other clients.

Version control, specifically git large file store.
Future Additions/Changes

Before this game could be released for production, we would definitely need to work extensively on better lightsaber movement including spatial tracking, rapid acceleration handling, and recalibration. Gyroscopic data is good for phone orientation, but is only a small part of full movement.

Adding in player movement would be a cool (but equally as difficult) project. Registering the headset’s lateral movement and corresponding it to the lightsaber’s would open up the game field a lot.

Additional gameplay including multiple droids attackers, an advanced mode where you had to deflect the bullets into droids at range, etc.
Demo Scenario

- Activate JediKnight App on both devices
- Select “Server” and “Client” on each device respectively
- Enter GUID on “Server” phone in the “Client” Window
- Ready to go!
- Place the “Server” phone into a Google Cardboard headset
- Manipulate the “Client” phone to control the on-screen lightsaber visible in the Cardboard headset
- Aim to capture as many enemies as possible before your health runs out.
Demo + Q&A