MDR Report - Team 10

Austin Fernalld, CSE, Connor Mclaughlin, EE, Alex Smith, CSE, Alex Bonstrom, CSE

FIVR: Finally Interacting with the Virtual Realm

***Abstract*—Virtual reality gaming is a rapidly growing market all over the world. While many advancements have been made in recent years to improve the visual and audio quality of VR environments, there haven't been many developments to improve physical interactions in virtual environments. At FIVR, we are in the process of designing a virtual reality controller that is able to provide feedback based upon a user's hand and grip position, allowing for a more lifelike gaming experience than ever before when interacting with in-game objects.**

I. INTRODUCTION

THE technology upon which today's gaming systems are built is improving at a rapid pace, and due to this progress, the world has games that are more interactive and immersive than ever. Video game graphics in recent years have gone from 1080p to 4K resolution, providing an incredible visual experience. On top of that, audio has gone from mono to immersive surround sound, but the controllers used to play these games have not advanced at the same rate. Most gaming controllers today follow the same format they did 30 years ago: a plastic handheld device with a multitude of buttons and joysticks.

The gaming world has put forth one major trend in recent years: virtual reality. It is now possible to enter the world of the games that have been created, but the 'reality' aspect begins and ends with vision. The controllers used to manipulate the environment are still based primarily on button input. After an initial push into movement-sensing controllers with the wildly successful Nintendo Wii, controller innovation was largely ignored for some time. Recently, Oculus have released their Touch controllers for their Rift VR system, which claims sub-millimeter 6 DOF (degrees-of-freedom) tracking [1]. Still, there is the issue of feedback. With these controllers the user is holding onto a plastic stick with buttons, and only given feedback in the form of vibration.

Our proposed solution is to design a mechanical controller that has a great deal of freedom of movement while allowing the controller to provide force-based feedback to the user depending on which object they are attempting to interact with in the virtual world. We have decided that our solution must determine the hand and individual finger positions of a user's hand, map this into the virtual world, and provide feedback at a multitude of different grip positions with latency quick enough to maintain the immersive feel of virtual reality, and weight that is less than or equal to three current gaming controllers (which are rather light). Power is not a major consideration of our project, as we are using a tethered design to minimize the latency. Further detail on our proposed system specifications is provided in Table 1.

Specification	Value
Weight	<1.8lbs
Finger Position Accuracy	+/- 5mm
(X, Y, Z) Position Accuracy	+/- 7.5cm
Distinguishable Finger Locations	10 positions/finger
Latency	<100ms

Table 1. System Specifications

II. DESIGN

A. *Overview*

With FIVR (Finger Interaction to Virtual Reality), we are attempting to solve the problems of controllers from the past with a new design approach. Our controller will determine the exact position of a user's hand in free space, the current position of each of the user's fingers, and provide resistance when the user attempts to grip down on an object inside of a virtual reality environment.

Some technologies which will be utilized to achieve this goal include the OpenCV image processing library, Python and C++ programs, a microcontroller for reading in and managing all data types, a modern cellular telephone for displaying the virtual reality environment, and a host computer as the main hub for interaction between our multitude of system components. For the feedback portion of our design we will be utilizing micro servos to stop the lightly-pressurized springs which are moved by each finger, and a time of flight sensor to measure the current compression of each spring, which corresponds to the current position of each finger.

The main four blocks in the diagram include the controller, the computer, the webcam, and the phone. The controller houses one position sensor and one micro servo for each finger, the gyroscope and accelerometer (integrated in one unit) for sensing rotation of the hand, and additionally the tracking object is attached at the base of the controller near the palm of the hand. Our fast microcontroller and wired communications channels will allow for low latency between action and response. The micro servo at the base of each finger will allow for multiple locking positions to simulate grip feedback. A minimum of 10 positions on the servo to be represented in the VR system. Resistance springs will be in place to give the fingers on the controller a natural push back feel.

Block Diagram

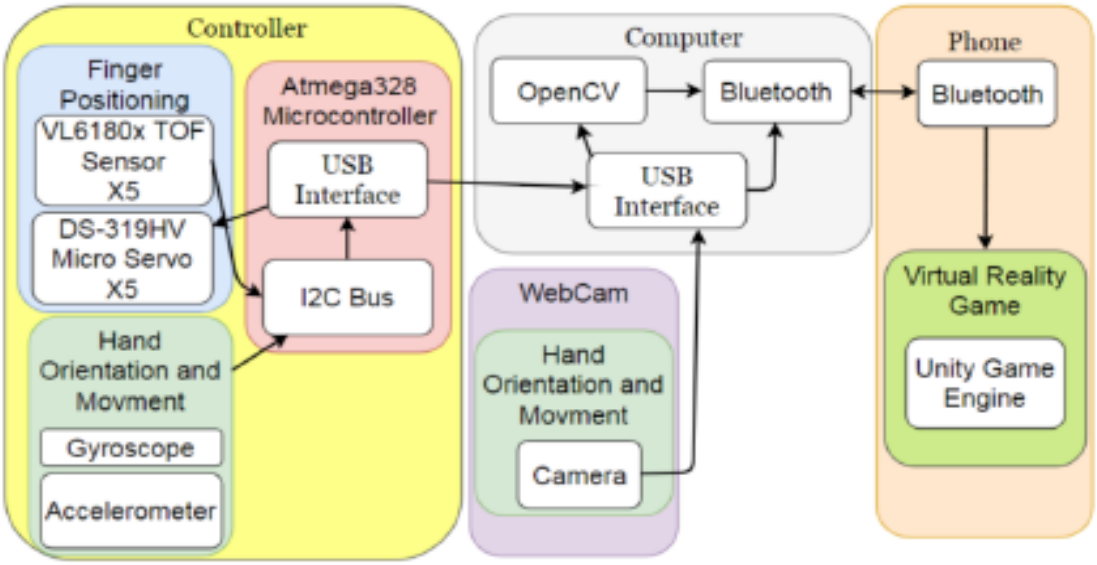

Figure 1. Block Diagram of FIVR System

After research into the average weights of video game console controllers and the longevity of playtime we selected our controller to be less than 1.8 lbs. [2] Offloading the power supply to the computer allows the weight of the controller to be minimal, and the accuracy of our sensors and information collection will ensure that finger position and hand position are read in without error. With complex dimensions and shapes needed to mirror, each finger will operate on its separate servo to provide a more accurate representation of objects. When choosing a measure of accuracy and the capabilities of the TOF sensors we selected a finger accuracy to about +/-5mm. As small as a margin as that is we provided believe it is highly possible in our system.

The computer will act as the main communication hub between all components of the project. The webcam will read in live data to determine the user's hand's current location relative to the body. The phone will run the Unity game engine and display the virtual reality environment interacting with the controller motion.

B. Hand Orientation and Movement

This block will provide hand position data in the form of (x, y) coordinates and a radius value that will then be passed through the microcontroller to the Unity game engine [3]. In order to track the hand position, we will be using a bright yellow Aero-Strike softball [4], the color of which has unique HSV (Hue, Saturation and Value) values to ensure that our program will be able to discern the correct object to track by its color and shape. The center of this object is tracked via (x, y) coordinates and the boundaries of the color are used to determine the current value of the objects radius, in terms of pixels. This object is also extremely lightweight and has virtually no air resistance.

A webcam will take in data from a fixed distance away from the user, and the frames from this webcam will be passed through an OpenCV-Python script which takes existing OpenCV frameworks such as motion detection and tracking and utilizes them in a Python script. The (x,y) coordinates of the center of the softball, along with the length of its radius, will be updated over thirty times per second and sent to Unity using a Python script. This data will be passed through the USB interface to the computer in real-time, where this information will be used to determine Unity VR coordinates. Additional data provided from an accompanying Bosch 9DOF sensor [5], which includes a 14-bit accelerometer and 16-bit gyroscope, will be interfaced via I2C communication and sent into Unity to determine the user's current hand orientation.

In order to map the controller's shifts in movement, an equation which correlates the current position in free space to the new current position for the VR environment is needed. Preliminary tests for equation development were conducted using a set distance of 8 feet from the webcam to the user. Each x-coordinate pixel changes maps .236cm/pixel and each y-coordinate pixel change maps to .229cm/pixel. There are 1000 possible x-values and 750 possible y-values, allowing for a wide range of possible controller positions. Due to fisheye lens distortion, the readings will be less accurate at the edges of the frame and the user is therefore encouraged to remain as close to the center of the webcam's field of vision as possible.

The user experience will be the most crucial component of testing this block's effectiveness and accuracy. If there is a noticeable difference between where a test user believes their hand should appear in the VR game and where it is actually displayed, this will interfere with the overall success of the project.

C. Unity Game Engine

To build our virtual reality environment, we are using the Unity game engine. Unity is the industry standard video game

engine [6], and it has all the tools necessary for our project. The project will be developed for smartphones, specifically the Apple iPhone. Because Unity does not come standard with iOS development packages, we will be using the Google VR SDK for Unity [7], which is under the Apache 2.0 License. The package requires at least an iPhone 5 or higher that is running iOS 8 or higher. For development and testing purposes, a team member has volunteered their iPhone 8 that is currently running iOS version 11.1.2. The phone will receive controller input signals from the computer through Wi-Fi signals.

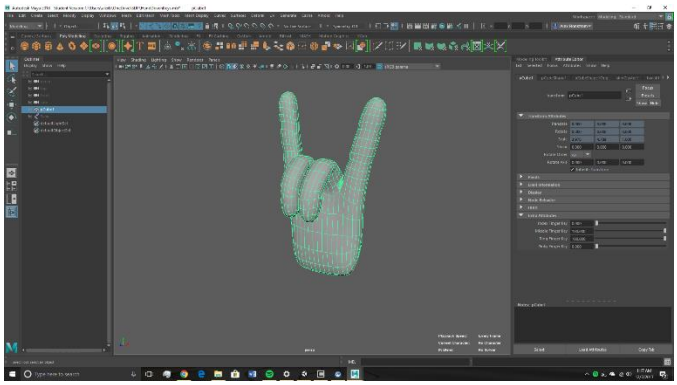

Figure 2. AutoDesk Maya Screenshot

Configuring the Unity environment requires a combination of scene and object modeling as well as scripting. The scene and object modeling is done through the Unity editor. Unity offers many tutorials online, and we used these to determine how to proceed with our scene and object development. Scripts in Unity allow any in-game object to be controlled through any input. All scripts in Unity must be written in C#. While none of us have direct experience with C#, it is an object-oriented language similar to Java, which we do have experience with. While the Unity engine has everything needed for realizing a full virtual reality environment, it is not well-suited for creating custom objects for the game. To model all the components for the game, we will be using the AutoDesk Maya modeling tool [8]. Fig. 2 depicts a simulation of individual finger articulation within Maya. Maya is a great tool to use with Unity because Unity accepts Filmbox (FBX) files that Maya projects can be exported as. In order to conduct testing we will be using tools which Unity provides for free in their online marketplace. Since testing is very difficult to do for modeling, we will only be running unit tests for the C# scripts created in Unity.

D. Finger Positioning

The controller block is responsible for sensing the player's hand position in addition to providing haptic feedback by restricting the player from closing their hand. Finger sensing is done with the use of five small, low powered, time-of-flight (TOF) sensors called the VL6180x [9]. Their job is to measure the distance between the finger and palm. The TOF sensors communicate with our Atmega328 microcontroller through the I2C interface. The Atmega328 transmits the distance values to a computer via serial port which then gets sent to the phone by Wi-Fi. The in-game finger model moves in response to these distance changes.

Finger sensing is accomplished with the VL6180x sensor made by STMicroelectronics. However, our project uses five breakout boards made by Pololu which incorporates the VL6180x and can be seen in figure 3 [10]. This breakout board was chosen over the standalone sensor because of the ease of system integration. The VL6180x sensor by default is a surface mount component that is 4.8 x 2.8 x 1.0 mm in dimension [11]. The process of using a reflow oven on such a small component was not worth the $25 saved by forgoing the Pololu breakout board. Another reason for sensor selection was that the breakout board by Pololu featured a voltage regulator which made integrating the 2.7V device on our 5V power supply plug-and-play [10]. All that was required to add the VL6180xs to our breadboard prototype was to solder 7 pin straight headers.

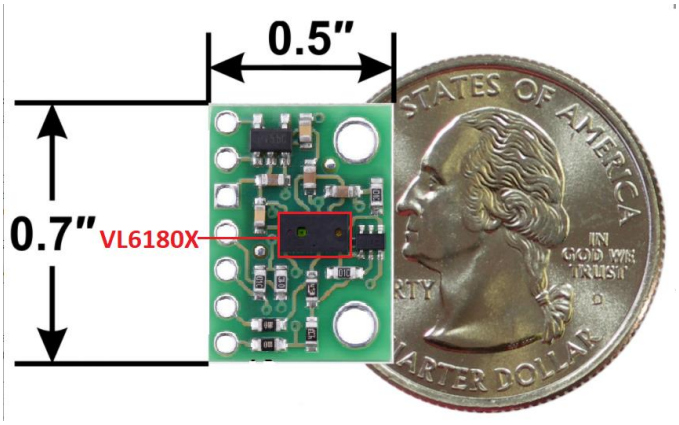

Figure 3. Pololu's breakout board with accompanying VL6180x shown beside a quarter for size reference.

The VL6180x is able to consistently take range measurements from 0-100mm with a resolution of 1 mm [11]. Our team measured the distance our fingers travel from open palm to picking up a shot glass and found that our fingers did not move more than 100mm. Therefore, our finger measurements should never fall outside of the recommended accuracy range. The VL6180x's datasheet reports that measurements can be taken up to 200mm with some loss in accuracy. The sensor is able to take continuous range measurements at a range period equal to about 11.5ms. This was found using formula one shown below:

$$Range\ Period = Pre - calibration + Range\ Convergence + Readout\ Averaging\ Time \quad (1)$$

Pre-calibration is the time needed for the sensor to calibrate it's instruments before each reading which lasts about 3.2ms. Readout averaging is used to reduce measurement noise and is set to 4.3ms by default [11]. Range convergence time is largely influenced by the range and reflectivity of the object it is sensing. The max convergence time at 100mm for an object

with 17% reflectivity is about 3.69ms so we will use 4ms to be safe [11]. This gives us a worst-case readout period of 11.5ms, which translates to a theoretical finger sensing rate of 87Hz. This would be more than enough to give the user the 60 frames per second needed for smooth gameplay. This theoretical sampling rate has not been reached in our lab. In tests conducted on the speed at which the TOF sensors could output a measurement ranged from 10ms to 32ms with the majority landing around 18ms. This gives a finger sensing rate of 55Hz, but this rate is expected to improve as the sensors are optimized for the material and environment. The discrepancies are suspected to originate from the cross-talk between the multiple VL6180xs emitting at the same time. Future tests are planned to try to improve the response time and improve sampling rate.

The sensors were tested to see if they fulfill the first requirement in our system specifications of measuring finger position within 5mm. We tested the accuracy of the sensors by conducting five tests at varying distances with 5 measurements taken from each of the four VL6180x. The arbitrary distances are based off objects in the lab that provided a steady base to lay a low reflectivity object across. This yielded tests that had a target distance of 122mm, 88mm, 64mm, 29mm, and 17mm. The test results are show in Appendix A and show that without calibration the sensors are inside the 5mm threshold when we average out the 5 tests. However, there are some measurements that are 10mm outside the target distance which could create a bad visual effect for the user who will see their fingers jump briefly to another location. This experiment was run without calibrating the TOF sensors which will be an option for our team if the accuracy proves to be a problem.

E. Micro Servo & Feedback

TOF Sensors are positioned to record each finger's distance to the palm to accurately draw the hand in the VR system.

Digital Metal Gear HV Servo (Corona DS-319HV): The micro servo component of FIVR is utilized to provide feedback to the handheld controller by locking the internal gears at a set positional degree given by the microcontroller. The servo operates on a 20ms period, corresponding to a frequency of 50Hz. Communication with the microcontroller is possible by first converting the CLK frequency to the desired 50Hz in the Phase Correct PWM Mode, shown in formula 2 below [12]:

$$f_{OCnxPCPWM} = f_{clkI/O}/(2 * N * TOP) \quad (2)$$

The selected Phase Correct PWM Mode operates at a high-resolution phase and frequency correct waveforms. It is based on a dual-slope operation that begins at BOTTOM(0x00) and starts up-counting until it reaches TOP (value selected in frequency equation). While up-counting and a compare match between the timer and our selected value(OCR1x), the output is cleared. While down-counting and there is a compare match between the timer again then the output is set. The output would look similar to Fig. 4 below[13]:

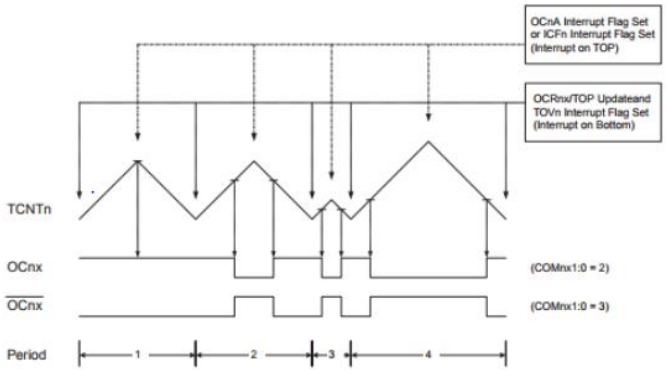

Figure 4:

With full control over the PWM we underwent experiments to test the compare values to the servo's positions. By testing maximum and minimum pulse widths we can record the max and min angles for our micro servo. Implementation with the controller dimensions is necessary to assure the hand movements are correct in each setting.

The value chosen to alter the pulse will be reliant on the VR data and a collision flag has been set. Depending on the dimensions of the VR object the controller is coming into contact with, the value(OCR1x) will be set accordingly to which the servo will lock in a position to mimic the dimensions of the object. With a servo for every finger the dimensions for each could vary depending on the object. For example, the grips of a broom or a baseball are different for each finger specifically so by applying a servo for each would provide a more accurate representation of the objects.

The angle the servo moves will be related to the feedback given to the controller. When receiving the proper information for the dimensions of the object the servos will react accordingly and bend to the proper amount.

F. Computer

The host computer will act as the hub of communication between all devices, taking in all webcam and sensor data, process this information, and output the resulting data to the phone via Wi-Fi. The computer and phone communicate through TCP in a classic client server system, where the computer is the TCP server and the phone is the TCP client.

G. Phone

The phone will run the Unity VR application which will project the location and movement of the hand. Additionally, the phone displays individual motion of each finger and entire hand movement around the body.

III. PROJECT MANAGEMENT

For our MDR deliverables, our original proposal was quite

unspecific, and we did not suggest any form of motion tracking would be functional. We had also originally proposed for one time of flight sensor to be reading in distance data, which was countered by the faculty evaluators as not being sufficient. We then revised our deliverables to include a preliminary form of motion tracking, in addition to four functioning time of flight sensors. We were able to provide all of these deliverables for the MDR presentation. The next step is to take these independent components interact with each other.

Proposed Deliverable Description	Completed?
Take distance measurements from four fingers	✓
Transmit distance measurements from micro controller to PC	✓
(X,Y) coordinates via webcam and OpenCV-Python	✓
Servo movement	✓
Demonstration of Individual VR Finger Interactions (using simulated data)	✓

Table 2: Proposed MDR Deliverables

The team has been working well together thus far, and we have attempted to play to each other's strengths and help each other out as much as possible. There is a considerable amount of work done jointly by Alex S. and Connor who are collaborating on the mechanical design and the interactions of the TOF sensors and servos. Austin and Alex B. also formed a subgroup to work on how Unity will mimic the precise position and orientation of the user's hand based off the webcam and controller sensor data. The spring semester will require a greater amount of cross-collaboration as integration of system components continues.

Each team member took initiative to learn something entirely new for each process of the project. Changing pace from rigid lecture-based structure to an independent type study gave us the opportunity to progress our design and researching skills. Each week new issues arise from the design, but our team members have been able to overcome every obstacle encountered. Alex S. had never worked with time of flight sensors or on an intricate mechanical design, Alex B. had never worked with the Unity game engine, Connor had never worked with micro servos and Austin had never worked with image tracking scripts. For each member there is a new area to gain knowledge in.

GANTT CHART OF REMAINING TASKS

	December	January	February	March	April
Alex Bonstrom	Finger Movements Displayed in Unity	Object Interaction in Unity	Bluetooth Integration	Unity Refinement	System Polish / Documentation Finalization
Connor McLaughlin	Micro Servo Implementation (Readings to Computer)	Further Micro Servo Implementation (with Sensors)	Mechanical Integration of Micro Servo & Controller Design	Finalize Data Collection Procedure	
Austin Fernalld	(X,Y) Tracking Implemented through OpenCV / Python	(Z) Tracking with Accelerometer or Gyroscope	Translate Motion Tracking Coordinates to Unity Coordinates	Assist with Unity Implementation	
Alex Smith	Calibrate Continuous Distance Readings	Continue with Distance Reading Accuracy & Timing	Mechanical Design & Integration with Components	Continue Mechanical Design	

Figure 5: Gantt Chart for Remainder of Project

IV. CONCLUSION

Thus far, we have made considerable progress on our project. Currently, the motion tracking brings in the (x,y) coordinates of the object, but cannot accommodate for the z position (depth), or any rotation about the axes of the controller. We have multiple functioning Time of Flight sensors which send data to the microcontroller, which is then forwarded to a serial port in the host computer. Within Unity, we have the ability to articulate finger movements displayed, but can only show responses to simulated data.

Integration between individual components will be key over the next few months of our project's development. We must take the coordinates of the hand position and accurately map that position to coordinates within Unity and be able to detect when there are collisions between the current hand position and an object, which will then trigger the micro servo for feedback and resistance. The physical structure of the controller must also be assembled and functioning. Clearly, there is much to be done for FIVR to be completed by the SDP demo day, but our group is confident we will be able to provide a rough demo of the ideal experience by committing a great deal of our time throughout the spring semester and focusing on cross-collaboration between our group members and their respective technical assignments.

ACKNOWLEDGMENT

We appreciate all of the assistance and advice that our advisor, Professor Kelly, has provided us with. We are also grateful for the constructive feedback provided by Professor Krishna and Professor Goeckel following our PDR and MDR presentations. Professor Hollot and Professor Soules have also contributed greatly to the development with component recommendations.

References

[1] Oculus Rift vs. HTC Vive vs. PlayStation VR. Retrieved February 6th, 2018 from http://www.tomshardware.co.uk/vive-rift-playstation-vr-comparison,review-33556-6.html

[2] Best PC Game Controllers 2018. Retrieved February 6th, 2018 from https://www.tomsguide.com/us/best-pc-game-

controllers,review-2776.html
[3] Unity 3d Game Engine. Retrieved February 5th, 2018 from https://unity3d.com/
[4] Amazon. Retrieved February 5th, 2018 from https://www.amazon.com/Franklin-Sports-Aero-Strike-Plastic-Softballs-Pack/dp/B0064I2WD0
[5] Bosch BNO055. Retrieved February 6th, 2018 from https://www.bosch-sensortec.com/bst/products/all_products/bno055
[6] "This engine is dominating the gaming industry right now", The Next Web. Retrieved February 6th, 2018 from https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/.
[7] Google VR. Retrieved February 5th, 2018 from https://developers.google.com/vr/.
[8] Autodesk Maya. Retrieved February 5th, 2018 from https://www.autodesk.com/products/maya/overview.
[9] "VL6180x", STMicroelectronics, Accessed February 3rd, 2018. http://www.st.com/en/imaging-and-photonics-solutions/vl6180x
[10] "VL6180X Time-of-Flight Distance Sensor Carrier with Voltage Regulator", Pololu, Accessed November 14th, 2017. https://www.pololu.com/product/2489
[11] STMicroelectronics, "VL6180x proximity and ambient light sensing (ALS) module," en.DM00112632 datasheet, Sept. 2013 [Revised Mar. 2016].
[12] Google. Microservo. Retrieved February 5th, 2018 from https://embedds.com/controlling-servo-motor-with-avr/
[13] "8-bit AVR Microcontrollers ATmega328/P Datasheet Summary", Figure 15-8. Phase and Frequency Correct PWM Mode, Timing Diagram, Atmel, Accessed February 1st, 201

MDR Report - Team 10

Alex Bonstrom, CSE, Austin Fernalld, CSE, Connor Mclaughlin, EE, Alex Smith, CSE,

APPENDIX

A. The graph below shows the result of four VL6180x time of flight sensors tested for relative accuracy of a target distance which has a +- 5mm error bar attached.