INCORPORATING HAPTIC FEATURES INTO PHYSICS-BASED SIMULATION

An Undergraduate Honors Research Thesis

by

RUKAI ZHAO

Submitted to the Computer Science Honors program at
Texas A&M University
in partial fulfillment of the requirements for the designation as a

COMPUTER SCIENCE HONORS UNDERGRADUATE

Approved by Research Advisor: Dr. Shinjiro Sueda

May 2019

Major: Computer Science
Minor: Math

TABLE OF CONTENTS

ABSTRACT

Incorporating Haptic Features Into Physics-Based Simulation

Rukai Zhao
Department of Computer Science & Engineering
Texas A&M University

Research Advisor: Dr. Shinjiro Sueda
Department of Computer Science & Engineering
Texas A&M University

In our graphic lab, we have developed many physics-based animations focusing on muscles and we hope to create an interactive interface with tactile feedback so that the users can not only see those physical features but also experience the forces in the muscle line. They will be able to touch on the surface of muscles and feel the muscle texture and they will also be able to drag the muscle line and feel the tension and forces. This is especially important for co-contraction of two opposing muscles, since co-contractions do not produce any motion but changes the stiffness of the joint. Therefore, we used the Geomagic Touch ^{TM} haptic device for generating the haptic feedbacks and to incorporate OpenHaptics for haptic programming. The first attempt was to embed the haptic code into the simulation base code. However, there were many stability issues due to the different versions of drivers and library files supported by different aspects of the hardware. Therefore, we implemented a network version which used TCP sockets to build the connection between the simulation code and the haptic code. Using this framework, the simulation code and the haptics code can run on two separate computers or on the same computer using two different processes. In order to make sure the data transfer process was fast and stable, we also added the multithreading feature into our codes. This will also help us to integrate haptic feature into Unity simulation code in the future. In the first attempt, the

haptic feedback was identical to the correct graphic scene. In the second attempt, we were able to feel the spring force from haptic client thread. We believe those programs can be used as entertainment and education purposes which can help the users better learn about muscles. Our future work will be focusing on combining our haptic client with a Unity simulation server.

ACKNOWLEDGEMENTS

I would like to thank my Research Advisor, Dr. Shinjiro Sueda, for his patient guidance and support throughout the research program. I also want to thank my lab mates who have helped me a lot for this research program.

CHAPTER I

INTRODUCTION

Nowadays, haptics technologies are commonly used in various researches especially in the field of computer graphics. The word "Haptic" is originally coming from the Greek word "haptesthai" which means "relating to the sense of touch" [5], so haptic technologies primarily deal with the touch sensory. Computer haptic is designed to integrate the senses of touch into a virtual environment using a physical 3D touch device. It helps to increase the technical system's realism and performance by allowing the users to interact with the machine [2]. Haptic sensing technology can gather the force information to help the operator navigate through deformable cloth and visualize the simulation [4]. It can also be used in education purpose like teaching students about bovine abdominal anatomy [6]. Our research will be focusing on incorporating haptic features into physics-based simulation.

Figure 1. Geomagic Touch™ Haptic Device

Previous work which motivated us is Anatomy builder VR, which is a virtual reality system that helps people learn canine anatomy [3]. We think if we can add haptic features into this VR application, the user can better experience the spatial visualization through the tactile feedback. They will be able to touch the muscle surface and feel the friction and resilience. They can also drag the muscle line to feel the force that is sometimes not well captured by the eyes. Having those features, the learners can have a deep impression on those anatomy knowledges.

In this research, we were dealing with physics-based simulation like particle system and spring forces. Our application will be able to feel the spring force and to experience the stiffness and friction on the entities' surface. We have implemented two haptic programs, one run on one process while the other one was able to run on separate processes. The haptic device we are using is the Geomagic Touch™ which provides us the environment to interact with the graphic window and to touch and drag the objects in the window (Figure 1). The language we are using for haptic programming is OpenHaptics. It is an API that implemented in C++ programming languages.

CHAPTER II

METHODS

We have developed two haptic programs. For the first program, we have written a single application that has a haptics thread and a display thread. The physics calculation has been done in the haptics thread. Because of the stability issues due to the different versions of driver and library files supported by different aspects of the hardware, we implemented the second program which we put physics simulation code on the server side and haptic code on the client side and used TCP Winsocket to connect these two processes.

1. Single Haptic Application (1st Haptic Program)

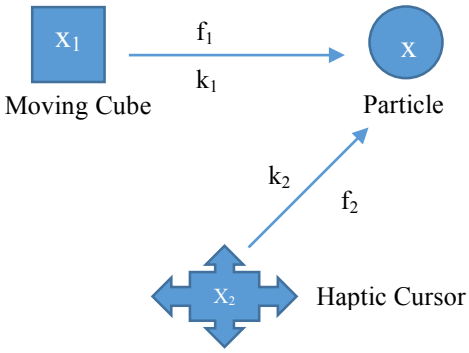

Figure 1. Embedded Haptic Program Prototype

In the first program, we had set up a scene that used implicit integration to compute the particle position from two points, a moving cube and haptic cursors (Figure 1). We used Visual Studio 2010 for supporting the haptic language and Geomagic Touch™ haptic device for feeling the force feedback. With this program, we were supposed to feel spring force between the haptic cursor and the particle. There were two major parts that we encountered in this program: physics-based simulation and HLAPI: Haptic Library API. The difficulty of this task was to integrate the

graphics frame and haptic frame so that we can see the correct graphic scene with the correct haptic feedback.

1.1 Physics-based Simulation

We first used explicit integration for computing the particle position x and the force value between the particle and cursor which will be used for generating haptic feedbacks. We used this function for getting the overall force for Particle x:

$$\vec{f} = k_1(\overrightarrow{x_1} - \vec{x}) + k_2(\overrightarrow{x_2} - \vec{x})$$

$\vec{f}$ is the net force for particle x, k_1 is the spring constant between the moving cube and the particle, k_2 is the spring constant between haptic cursor and particle, $\overrightarrow{x_1}$ is the position of the moving cube, $\overrightarrow{x_2}$ is the position of the haptic cursor and $\vec{x}$ is the position of the particle.

We used these two functions for updating the particle position and velocity:

$$\overrightarrow{v_1} = \overrightarrow{v_0} + \frac{\Delta t}{m} \cdot \vec{f}$$

$$\overrightarrow{x_1} = \overrightarrow{x_0} + h \cdot \overrightarrow{v_1}$$

$\overrightarrow{v_1}$ is the current velocity of the particle, $\overrightarrow{v_0}$ is the previous velocity of the particle, Δt is the elapsed time, m is the mass of the particle, $\overrightarrow{x_1}$ is the current position, $\overrightarrow{x_0}$ is the previous position and h is the time step.

However, the explicit integration was not stable, so we replaced that with implicit integration. The velocity function turned to:

$$\overrightarrow{v_1} = (M - h^2 k)^{-1}(M\overrightarrow{v_0} + h\overrightarrow{f_0})$$

$$k = -(k_1 + k_2)I$$

M is for the particle mass matrix and I is for the identity matrix.

1.2 HLAPI: Haptic Library API

The haptic programming language used on the Geomagic Touch ™ is OpenHaptics. There are three APIs: QuickHaptics micro API, Haptic Device API (HDAPI), Haptic Library API. QuickHaptics is a micro API that has built-in geometry functions with default settings, so the users can set up the haptic easily with fewer lines of code. HDAPI is a low-level C API which enables the programmers to directly manipulate the force rendering and the threading of the program. HLAPI is the middle ground that is designed to be used with OpenGL code and it simplifies the synchronization of haptic and graphic threads [1]. Since HLAPI is a high-level C API for haptic rendering and it can be embedded into the OpenGL code, we used this API for most of the haptic codes.

1.2.1 Custom Force Effect

Using HLAPI, there is a servo loop thread in the program that calculates the forces each time and sends back to the haptic device. This loop runs 30 times faster than the graphic rendering loop and it processes at approximately 1 KHz. Therefore, in our first program, we put all the simulation calculation inside the compute force callback function to make custom effects. The graphic loop will read the particle position each time from the haptic loop which was stored in a global variable. The graphic loop was then able to render the correct position of the particle on the screen and feeling the force from the haptic device. This process is shown in Figure 2. The three major custom force effect callback functions we used were: HL_EFFECT_START, HL_EFFECT_STOP and HL_EFFECT_COMPUTE_FORCE.

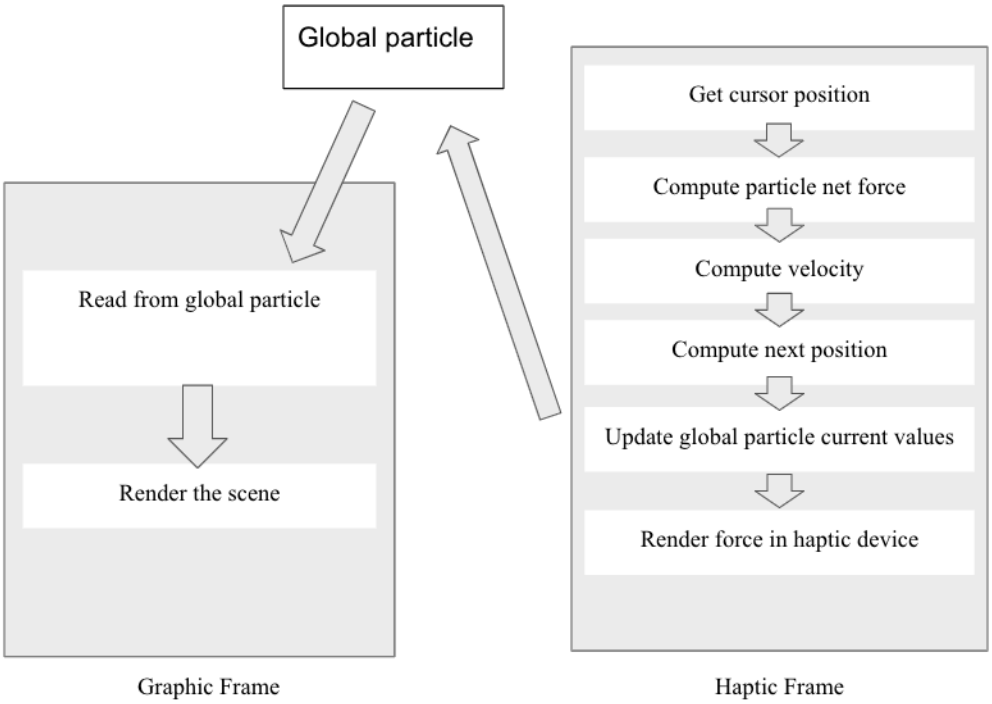

Figure 2. Graphic Frame and Haptic Frame Design

1.2.2 Mapping from Haptic Workspace coordinates to Graphic World Coordinates

One thing that needs to be taken care of when reading the position from the haptic loop is that the position is in haptic workspace coordinates, not in graphic world coordinates. So when transferring the position from haptic workspace coordinates to graphic world coordinates, we applied the inverse matrix of the model to workspace transform which was generated using the in-built function. Figure 3 points out the change.

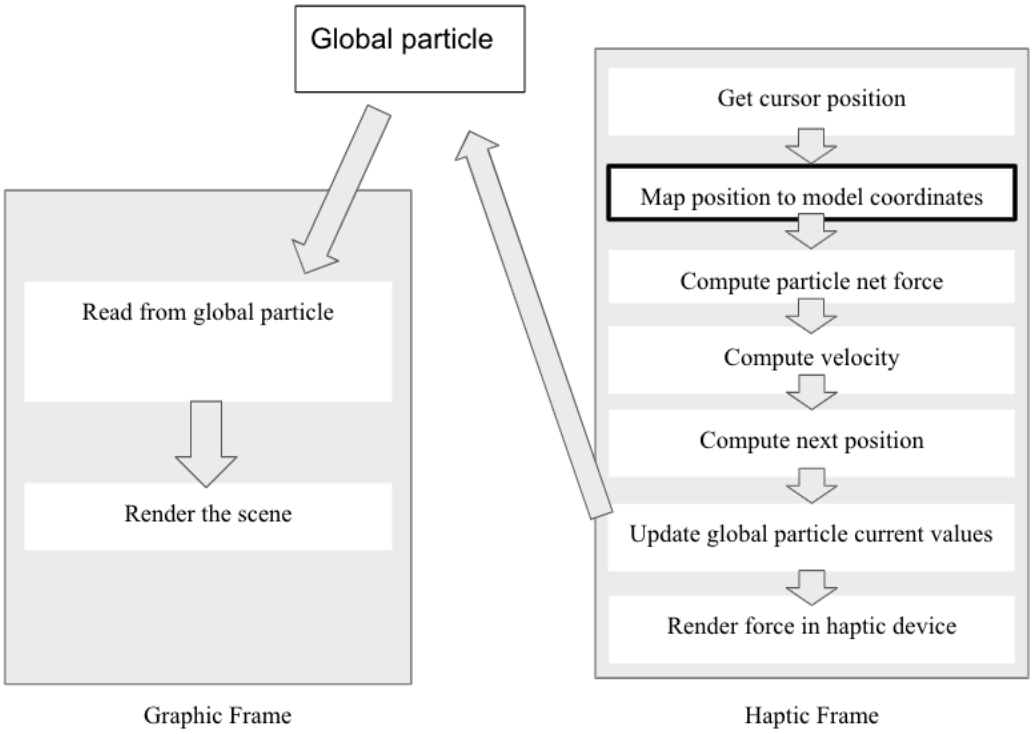

Figure 3. Graphic Frame and Haptic Frame Design with Coordinates Mapping

1.3 Pseudocode

Function: Haptic Frame
Input Parameters: cache and particle_data
Output: particle_data and result_force [3]
(1) h ← 0.01
(2) cursorPosition ← cursor current position from cache
(3) proxyPos ← matrix workspacetomodel * cursorPosition
(4) cube_force ← k_{cube} * (cube_position – particle_position)
(5) haptic_force ← k_{haptic} * (proxyPos – particle_position)
(6) net_force ← haptic_force + cube_force
(7) particle_velocity ← (particle_mass – $h^2k)^{-1}$ (particle_mass * particle_velocity + h * net_force)
(8) particle_position ← particle_mass + h*particle_velocity
(9) result_force [3] ← haptic_force for haptic force rendering

Function: Graphic Frame

Input Parameters: particle_data
Output Parameters: cube_position
(1) t ← current time
(3) radian ← t /180 * π/50
(3) **If** cube is moving = true **then** cube_position ← 2 * cos (radian), 2 * sin (radian), 0
(4) **Else** cube_position ← 0, 0, 0
(5) **HLBeginFrame**
(6) **glPushMatrix**
(7) Translate to cube_position
(8) Draw cube onto the screen
(9) **glPopMatrix**
(10) **glPushMatrix**
(11) Translate to particle_position
(12) Draw current particle onto the screen
(13) **glPopMatrix**
(14) **glBegin**
(15) Draw line between cube and particle
(16) Draw line between particle and cursor
(17) **glEnd**
(18) Draw_cursor ( )
(19) **HLEndFrame**

2. Socket Haptic Program (2nd Haptic program)

The second program was implemented on separate processes, unlike the first program which runs on the single console. The design of this program was that the physics simulation run on one process and the haptic rendering run on another process. We established the connection between these two processes. We also used Visual Studio 2010 for supporting the haptic language and Geomagic Touch™ haptic device for feeling the force feedback. There were three parts that we encountered in this application: physics-based simulation, TCP socket programming and multithreading. The difficulty was that we need to ensure the connection was

stable and the data transferred was fast enough so that we could feel consistent force from the haptic device.

2.1 Physics-based Simulation

In the second program, we used the spring force function to calculate the force based on the position sent by the haptic server:

$$\vec{f} = -k \cdot \vec{x}$$

The reason why we used spring function was that it was easy for us to test the haptic feedback.

2.2 TCP Socket Programming

Since we wanted to let our simulation code and haptic rendering code to run on separate processes and to transfer data over the connection, we decided to use TCP socket programming. we used winsock library in the Visual Studio and TCP protocol which guaranteed stable and ordered packets sending.

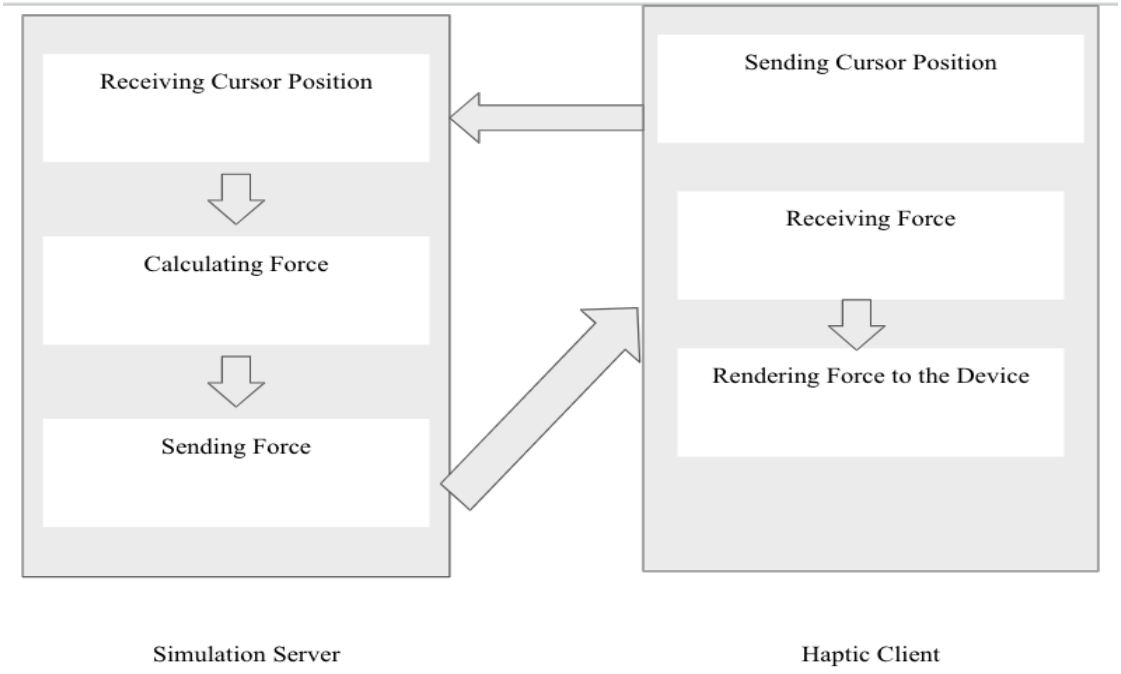

Figure 4. Simulation Server and Haptic Client Data Transfer

From Figure 4, we packed the three floats which represented in X, Y and Z directions into the buffer and send it to another process. Server was responsible for receiving the position from the client and calculating the force and sending it back to the client. Client was responsible for sending the haptic cursor positions and receiving the result force and rendering force in the haptic device.

2.3 Multithreading

At first, we didn't use multithreading for our second program. Instead, data transfer was done by server wait until received the position from the client and then calculated the force based on that position. There were delays between each force values received by the haptic client, so that the haptic feedback was discontinuous. In order to minimize the delays, we used multithreading in both client and server, so the client will keep sending positions to the server

and the server will keep sending calculated forces back to the client. The result haptic feedback was very consistent and we can feel the cursor was pulling toward the center of the device.

2.4 Pseudocode

Server Program:

Function: Receive position thread
Input Parameters: data Output Parameters: result_force [3] (1) **While** true (2) Receive from client and store in buffer (3) Read position float x, y, z from the buffer (4) result_force [3] ← k * position (5) **End**

Function: Send force thread
Input Parameters: data Output Parameters: buffer (1) **While** true (2) Pack force in a buffer (3) Print send force value (4) Send to the client (5) **End**

Client Program:

Function: Send position thread
Input Parameters: data Output Parameters: None (1) **While** true (2) Read position from global_position (3) Pack position in a buffer (4) Send to the server (5) **End**

Function: Haptic Frame
Input Parameters: cache and particle data Output Parameters: result_force [3] (1) global position ← current cursor position from cache (2) Receive calculated force from client (3) Scan buffer and save force to result_force [3] (4) Render force to haptic device

CHAPTER III

RESULTS

1. Single Haptic Application (1st Haptic Program)

For the first model, the method we used to test our model was to feel whether the force was identical to the graphic scene. The forces that we supposed to feel in each of the scenario according to the Figure 5 and 6. We wanted to have smooth forces without sudden pulses.

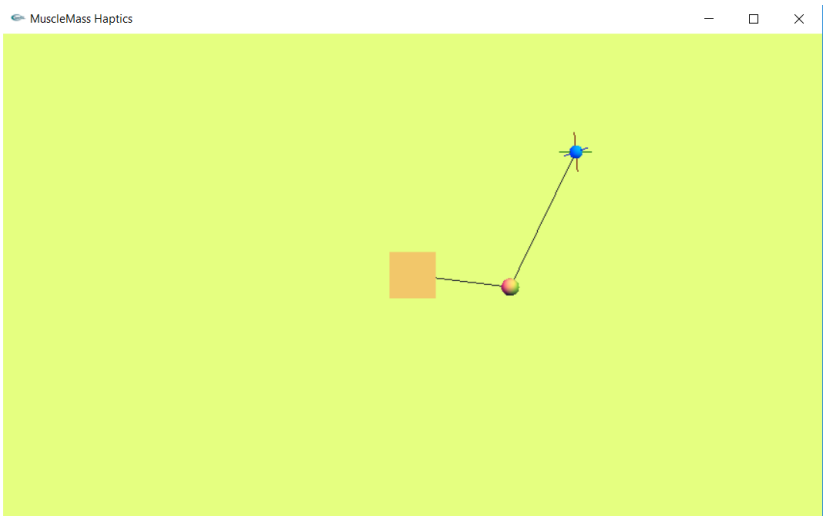

Figure 5. Capture from program 1

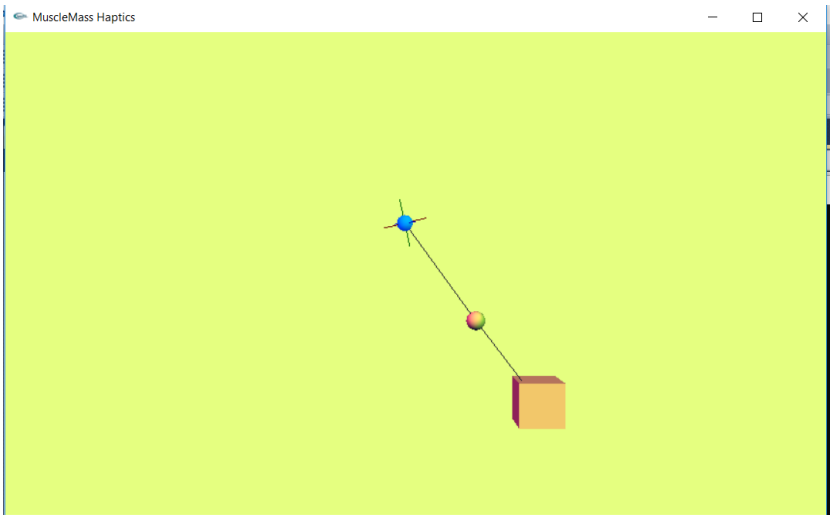

Figure 6. Capture from program 1

According to the scene of Figure 5 and 6, the colorful ball was the particle which we used implicit integration to calculate the position. The blue point was the haptic cursor position and the cube was either fixed at the center of the screen or moving in circular motion.

From Figure 5, when the cube was fixed at the center and the cursor swirled, the particle would also swirl and we felt the particle was swirling from the haptic cursor. When we stopped swirling the cursor suddenly, the force did not stop and we continued to feel the force until the particle was not moving in the graphic scene.

From Figure 6, the cube was moving in circular motion and we tried to fix the cursor at the center of the screen. We felt the cube was dragging the cursor toward itself and the cube was moving in circular motion.

Based on result above, the graphic loop was able to generate correct graphic scene and the forces generated by the haptic device felt identical to the graphic scene. The forces were consistent and smooth without having sudden pulses.

2. Socket Haptic Program (2nd Haptic program)

For the second program, the method we used to test our model was to feel whether the directions of the forces were pointing towards the center of the device. The force should be consistent without having sudden pulses. Figure 7 shows how we manipulated the haptic cursor in three different directions.

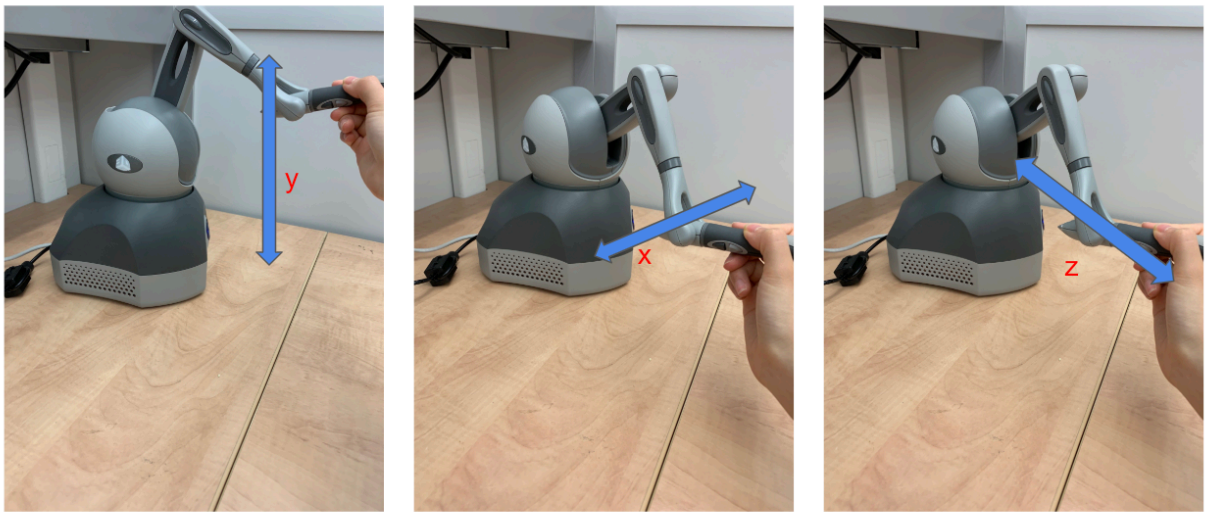

Figure 7. Haptic Device Cursor Moving in x, y, z directions

```
Connected to server
*************************************
This is the DEVELOPER EDITION of OPENHAPTICS, commercial distribution is prohibited
Please contact 3D Systems to obtain a commercial license
*************************************
*************************************
This is the DEVELOPER EDITION of OPENHAPTICS, commercial distribution is prohibited
Please contact 3D Systems to obtain a commercial license
*************************************
Custom effect started
```

Figure 8. Haptic Client Console

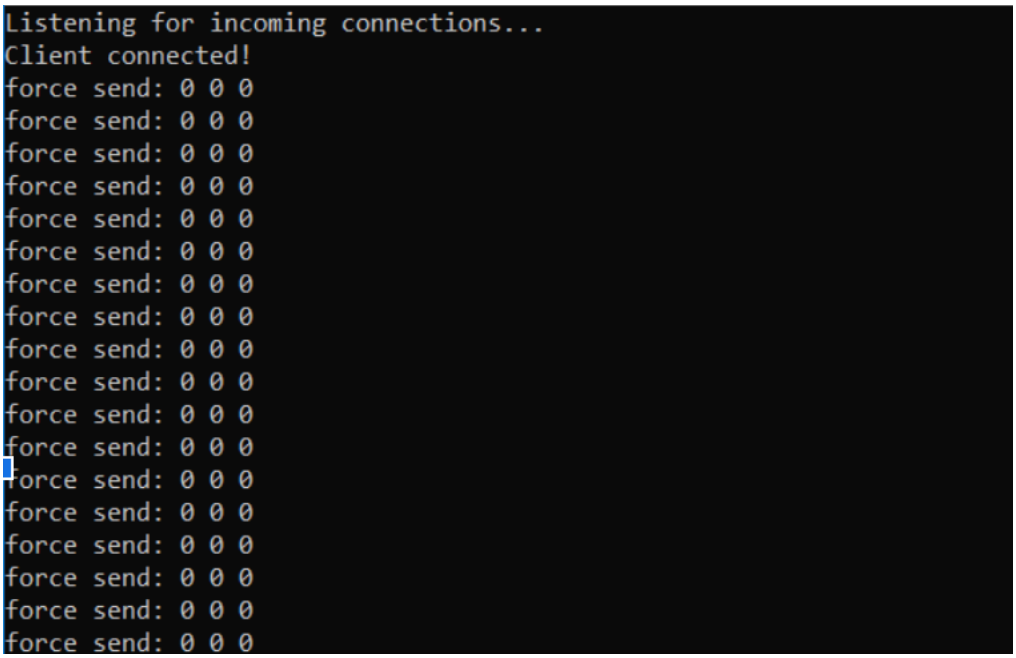

Figure 9. Server Console

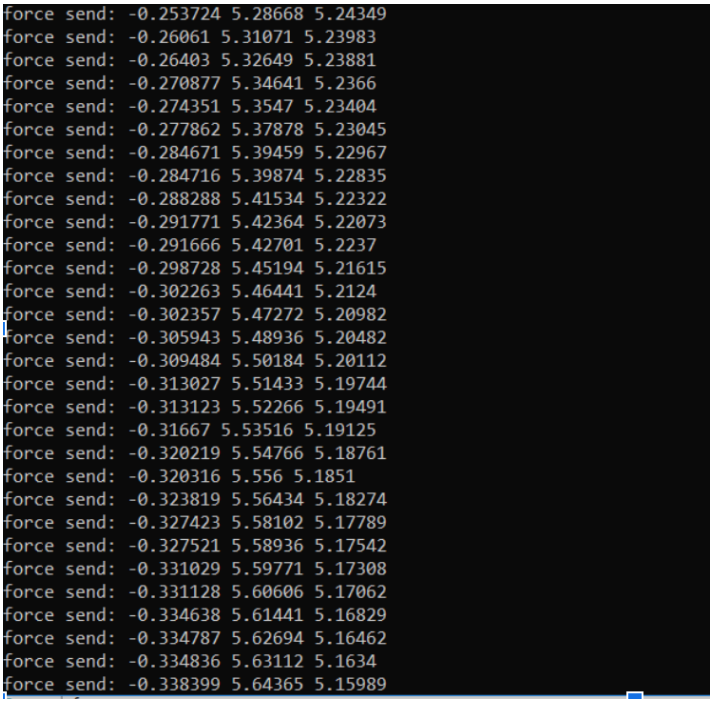

Figure 10. Server Console

Figure 8 shows the haptic client console. After connected to the server, it began the haptic rendering process. Both Figure 9 and 10 show the server console. When starting the

server, the force sending were zero, but when it received the position from the client, it started to print the values onto the screen. For the second program, we can feel the direction of the force was pointing toward the center. The further we moved away the cursor from the center of the haptic device, the stronger force we can feel. The round of sending and receiving can be done in 16kHz. During the testing, the forces were quite consistent without feeling any pulses.

CHAPTER IV

CONCLUSION

For the two programs, we have successfully generated correct force feedback based on the graphic scene. However, it is hard to measure the accuracy of those force feedbacks because of the haptic device in-built feature. We did not build a model that can measure how well the haptic feedbacks mapped to the graphic scene. Also, sometimes the cursor will drop into the surface of the mesh, though we were using the in-built method for the device. Nevertheless, we think our programs will be workable for feeling the muscle simulation. The second program will be very helpful for generating haptic feedback for the muscle simulation code which was performed on a higher version platform or on a different platform, since we can use TCP socket method to transfer the data between platforms.

Future work

Our future work will be focusing on combining our haptic client with a Unity server which is a VR muscle simulation application. We want to be able to feel the muscle and to drag the muscle line in the spatial environment. We also want to set up an application that can test the accuracy of our haptic output which means that we want to build a model that can measure how well the haptic feedbacks mapped to the graphic scene.

REFERENCES

[1] *OpenHaptics Toolkit version 3.4.0 Programmers Guide.* 3D Systems, Inc., 2015. Print.

[2] M. Strese, J. Lee, C. Schuwerk, Q. Han, H. Kim and E. Steinbach, "A haptic texture database for tool-mediated texture recognition and classification," *2014 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE) Proceedings*, Richardson, TX, 2014, pp. 118-123. doi: 10.1109/HAVE.2014.6954342

[3] Jinsil Hwaryoung Seo, Brian Michael Smith, Michael Bruner, Margaret Cook, Jinkyo Suh, Michelle Pine, Erica Malone, Steven Leal, Shinjiro Sueda, and Zhikun Bai. "Anatomy builder VR: comparative anatomy lab promoting spatial visualization through constructionist learning," *SIGGRAPH Asia 2017 VR Showcase (SA '17). ACM,* New York, NY, USA, 2017, Article 1, 2 pages. DOI: https://doi.org/10.1145/3139468.3139470

[4] A. Clegg, W. Yu, Z. Erickson, J. Tan, C. K. Liu and G. Turk, "Learning to navigate cloth using haptics," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, 2017, pp. 2799-2805. doi: 10.1109/IROS.2017.8206110

[5] A. E. Saddik, "The Potential of Haptics Technologies," in *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 1, pp. 10-17, Feb. 2007. doi: 10.1109/MIM.2007.339540

[6] T. kinnison, N. D. Forrest, S. P. Frean, S. Baillie, "Teaching bovine abdominal anatomy: Use of a haptic simulator". *Anatomical Sciences Education,* vol. 2, issue 6, 24 Sept. 2009. https://doi.org/10.1002/ase.109