Cancer AR Report

COSC 447 - Software Maintenance

Irving K. Barber Faculty of Science
Department of Computer Science
University of British Columbia

Author: Parsa Rajabi - April 2021

Supervisors:
Dr. Patricia Lasserre
Dr. Khalad Hasan

Table of Content

Introduction

CancerAR is a mobile Augmented Reality (AR) application aimed at assisting the doctor-patient relationship during virtual doctor appointments. Through this application, doctors can use the 3D organ model to provide a visual component while advising patients about their health conditions. CancerAR provides both users full access to control the 3D organ (rotate, scale, move) while also being able to annotate it using the built in annotator.

Goals

The following goals were set at the beginning of the semester:

1. Report on Vuforia, Leap motion and the previous app
    a. Move and maintain code in a github repository
    b. Explore technologies used in application
2. Demo on the working prototype with Vuforia image tracking
    a. Setup project in unity and document the process
3. Preliminary demo on touch gestures for 3D manipulations
    a. Remove buttons on the bottom of screen
    b. Read various papers on 3D manipulation touch gestures
    c. Create touch gestures for application based on readings
4. Demo on touch gestures for 3D manipulations
    a. Document new touch gestures
    b. Replace status with x/y/z axis image
5. Demo on the remote multiplayer supports for the app
    a. Setup multiplayer functionality with one-directional access
6. Demo on multiplayer features: player visualization
    a. Complete multiplayer functionality with bi-directional access
7. Preliminary demo on single player features: Annotation
    a. Setup annotation for single player
8. Final Demo on multiplayer features: Annotation

    a. Complete annotation for multiplayer
9. Project presentation
    a. Present work completed at Undergraduate Research Conference

Aside from all the technical goals of this project, it was critical to create and maintain documentation as part of development to ensure future developers can get setup and contribute to the project with the least amount of effort.

Task Accomplished

Setup Github Repository

One of the initial goals of this project was to move the code from a shared Google Drive folder into a github repository. By doing so, the project would benefit from version controlling and continuous integration and development. The github repository can be found here: https://github.com/Parsa-Rajabi/UBC-Lungs-VR-AR-App

Setup Project + Created Documentation

After moving the project into a github repository, it was time to set up and run the project. Unfortunately due to lack of documentation, this step took longer than expected, however, eventually the project was set up on running. To help future developers, a "Getting Started" guide was created on the github repository's README file which explains what tools to install and how to set up and run the project.

Touch Gestures

Once the project was set up and running, the first feature was to remove the buttons alongside of the screen and integrate a new touch gesture into the application. Given the complexity of this feature, it was broken down into smaller components:

1. Remove buttons one by one
2. Read and evaluate various papers on 3D manipulation
3. Create a set of touch gestures based on the current research studies.

Upon completion, the following touch gestures were embedded into the application (figure 1).

	Moving	Scaling	Rotating
1 Tap	Drag	Pinch	X - Axis
2 Taps	-	-	Y - Axis
3 Taps	-	-	Z - Axis

Figure 1: Summary of Touch Gestures Implemented - source

Given the 3D aspect of the models, each touch gesture was mapped to a specific number of taps. Based on the number of taps, a new set of features would be accessible. The application first starts up in the **X axis** mode, allowing the user to move the model using 1 finger drag and scale it using a pinch (2 fingers) motion. In addition, the user would be able to rotate the model using a circular motion in the **X axis**. Once the user double taps the model, they will be in the **Y axis** mode where moving and scaling are disbaled and they can only rotate the model using the circular motion in the **Y axis** direction. Lastly, the user can triple tap on the model to access the **Z axis** rotation which can be done using the circle motion described

before. At any point, the user will have the ability to tap once on the model to be taken back to the **X axis** mode where they will regain access to moving and scaling the model.

Developer Note: These touch gestures are controlled via Lean Touch library. To learn more about how to access and implement Lean Touch, view the README file for the current Lean Touch version imbedded within the application

Multiplayer - Model

Upon successfully implementing the new touch gestures, the next important feature was to enable a multiplayer environment where a user can view and control (move, scale and rotate) the model and the other can follow along. Given the complexity of this feature, the steps were broken down as the following:

1. Investigate and select a multiplayer asset that is up-to-date and reputable
2. Implement a multiplayer asset with single directional access: one user is the host and the other user is a viewer with no controlling access
3. Implement multiplayer asset with bi-drecitional access: both users are able to control the model (move, scale, rotate)

This feature works as the following:
User A:

Upon starting the application, they are prompted to enter a room name. After typing a room name and pressing on the "Create / Join Room" button, they will be taken to a screen to select which model they would like to view. After selecting a model, they're taken into the application where they can view the model.

User B:

Upon starting the application, they are prompted to enter a room name. After typing in the **same room name** as User A and pressing on the "Create / Join Room" button, they will be taken to a screen to select which model they would like to view. After selecting the **same model** as User A, they're taken into the application where they can view the same model as User A.

At this point, both users are in the same network "room" and are sharing the same view of the model. To take ownership of the model, users can tap anywhere on the screen and they will become the primary controller of the model while the other user becomes a "viewer". At any point, the viewer can tap on their screen to take over and start controlling the model.

Developer Note: Photon's Ownership has multiple settings and this process can be changed to UserA is the owner and for UserB to become the new owner, they have to send a **request** to UserA and if they accept, UserB will become the owner. To learn more on how to implement this, view Photon's Ownership Documentation

Multiplayer - Annotation

Another aspect of this application is providing a tool for users to annotate on their screens. Using the annotation feature, doctors and patients can draw on the screen to highlight or point out specific areas of the model. Similar to to the multiplayer feature, the annotation aspect of the application also required to be broken down into more manageable components:

1. Investigate and attempt to expand the already existing word-annotation to multiplayer access
2. If existing word-annotation does not work, develop a new single player annotation tool to draw on the screen
3. Expand the single player annotation to multiplayer (allow both users to view the other users' annotations)

Here is a quick recap of how the annotation feature works:
After a user has created / joined a room, selected a model, they're taken to the main screen of the application. On the top left of the screen, the user can select "Annotate" and after doing so, the application switches to annotation mode. Here is how the application behaves while in "annotation mode":

- The model's position is locked
- All model controls are disabled (moving, scaling, rotating)
- The x/y/z status icon is hidden
- The button at the top left (previously said "annotate") changes labels to "Done"
- The user can annotate the model using their finger

The annotate mode is active until the user selects the "Done" button at the top left, which then disables annotation mode and the application goes back to default behavior:

- The model controls are re-enabled
- The x/y/z status is visible
- The button at the top left changes back to "annotate"

Developer Note:
All the logic for this feature can be found in the `Annotate.cs` script. The script uses a prefab and creates a **clone** of it every time the user draws a new line. Currently, the lines are set to disappear after **3 seconds**, however, this timer can be changed in unity > Annotate (game object) inspector. To make it so the lines do not disappear, you'd have to set the duration to **infinity**. Please do keep in mind that if the duration is set to infinity, you'd have to implement a new feature to **undo** and/or **delete** all lines on the screen.

Another item of importance is how Photon network is treating the annotation. Given the **Photon instance** feature, this relies heavily on the users' network connection to the internet and more importantly to **Photon Servers.** The higher the ping, the higher the latency.

Bonus: Refactored Code

As part of this project, most of the initial code was either

a) Removed
b) Refactored

Given the new complex features of the project, the initial code made it difficult to access and update variables throughout the application. Although the new code base does not have the best structure, it still provides the ability for future developers to build and expand on existing futures.

Bonus: Help Panel

As part of the touch gesture feature of the project, it was important to provide some sort of help guide for users in regards to how they can control the model. The following help screen was created which is shown to users after they select a model to view:

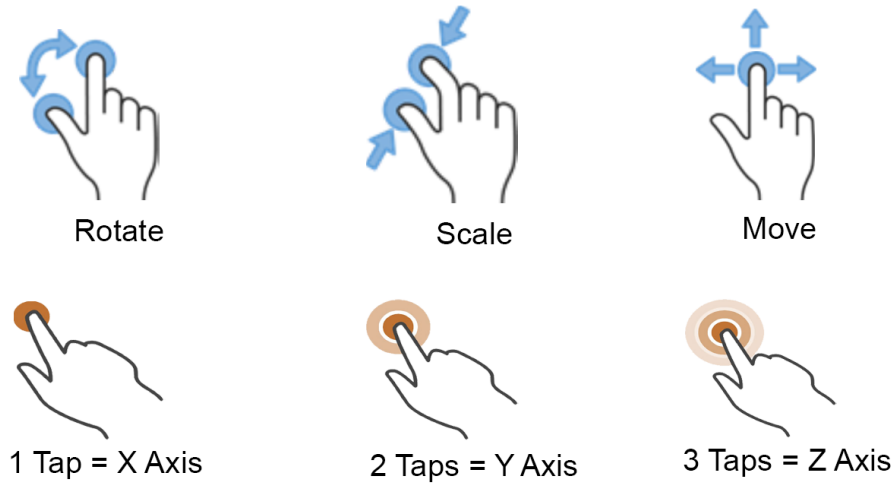

Figure 2: Summary of Touch Gestures Implemented - source

Users can also view this help screen by selecting the "Help" button on the bottom right side of the screen.

Bonus: Exit Application

One last bonus feature of this application is an exit button located on the bottom left side of the screen. Users can tap on this button to exit the application on their mobile devices.

Challenges and Limitations

Lack of Documentation

As mentioned before, one key factor that caused delays throughout this project was lack of documentation from the previous developer. Given the various different assets and technologies of the project, it was difficult for a new developer to pick up where the previous developer left off as there were no guides or README files available. Fortunately, this won't be an issue for future developers as this report and the README files on the github repositories should serve as a starting point for future development.

Multiplayer - Photon

One of the main aspects of the project is the multiplayer functionality. Based on the current network technologies in the market, Photon was deemed to be the best option for this project. However, there are some key information about Photon (PUN) that are important to note:

1. Pricing
    a. The application is currently using a free tier which provides the following:
        i. Allows up to 20 Connected Users (CCU) at the same time
        ii. Suitable for ~8 K MAU
        iii. 500 Msg/s per Room
        iv. Up to 16 Peers per Room
        v. Includes 60.0 GB Traffic
        vi. CCU Burst not included
    b. For more information about pricing visit their website
2. Server Location
    a. A key factor for multiplayer access is latency. Given our location (Kelowna, BC - Canada), the best public photon servers (with the least latency) are located in San José (USA). For research purposes, the application could be set to Photon's servers in Montreal (Canada), however, the latency does become noticeable for users located on the west coast.
    b. For more information about Photon server locations visit their website
    c. Developer Note: To change the server location on the application, use the Photon Server Setup Wizard and set the "Fixed Region" to the Token value found on Photon's website.

Areas of Improvement

Update Unity Version

Currently this project is running on **unity version 2019.1.10f1**. Unfortunately, this version of unity does not receive updates from unity as it is not part of the Long term Support (LTS) program. It is crucial for this project to be updated to a unity version that is enrolled in the LTS program to ensure it receives required bug-fixes from unity. To view a list of LTS releases, check out unity's release website.

Code Structure

One key area of improvement for this project is the fundamentals of the code structure. Given the time constraint of this project, there was not enough time to refactor the code structure however, this is an area that does require attention in the future.

Code Test

Currently this project has no tests associated with it, however, this is an item that should be prioritized in future development as tests are a key element in software development.

Optimize Build Times

During the development of this application, it became evident that the lengthy build times could become a problem in the future. Currently, the build times are as the following:

- Unity Build iOS Application ~15 minutes
- xCode Build and Run Application on Device ~10 minutes
- Total build times ~25-30 minutes

Here are the specs of the computer that was used for measuring build times:

iMac (Retina 4K, 21.5-inch, 2017)
Processor: 3 GHz Quad-Core Intel Core i5
Ram: 16 GB 2400 MHz DDR4
Graphics: Radeon Pro 555 2 GB

It is recommended to review the current code base and refactor the "expensive" operations that cause such long build times. The build times could also improve if a newer version of unity was used for this project.

Conclusion

This report highlights the various complex features that were developed and implemented from January - April 2021 as part of the COSC 447 course. The major features that were implemented are:

1. New touch gestures
2. Multiplayer
3. New multiplayer annotation

As part of this course, there were regular video updates available to demonstrate the new features. All these videos alongside of specific tasks that were completed can be found on the github repository's document README file.

References

Course Weekly Updates

Course Progress Report:
https://github.com/Parsa-Rajabi/UBC-Lungs-VR-AR-App/tree/dev/docs/2021%20Spring

Photon

Photon Pricing: https://www.photonengine.com/en-US/PUN/Pricing
Photon Server Locations:
https://doc.photonengine.com/en-us/realtime/current/connection-and-authentication/regions
Photon Ownership Documentation:
https://doc.photonengine.com/en-us/pun/v1/demos-and-tutorials/package-demos/ownership-transfer

Unity

Unity Release Page: https://unity3d.com/unity/qa/lts-releases

Lean Touch

Lean Touch README Repository:
https://github.com/Parsa-Rajabi/Lean-Touch-README