Chapter 1

Application User Interface

In this chapter some of the useful UI features of the application will be discussed.

1.1 Main Layout

The application has three main scenes. The first one is the home screen which shows buttons to take the user to the other two scenes; "Enter Test Mode" button takes the user to the scene which is used for collecting data, and "Analyze Data" button takes the user to a scene that allows him/her to view the collected data. The home screen also contains two radio buttons to allow the user to select which hand (left or right) he/she will be testing with the gestures. Figure 1.1 shows the layout of the home screen.

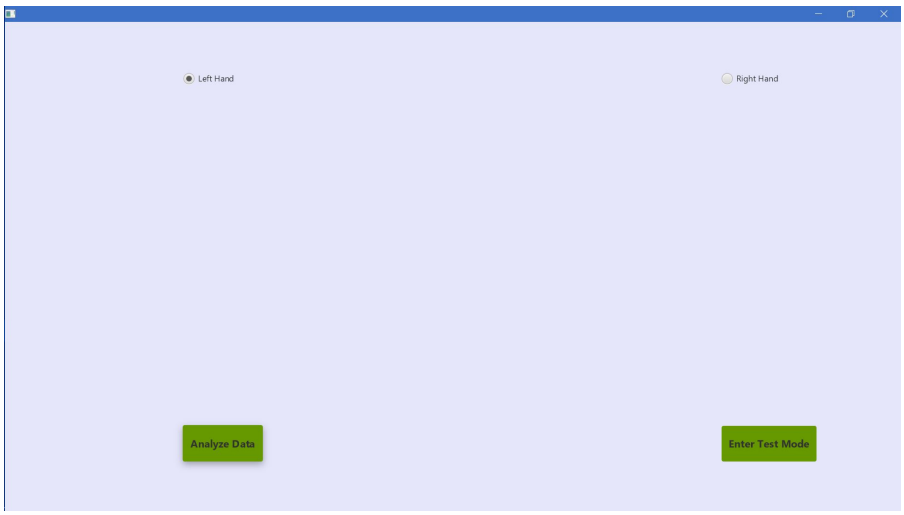

FIGURE 1.1: The scene that the user sees initially when they load the application.

1.1.1 Creating or Loading User

The user can click on the "Enter Test Mode" to go to the scene where data will be collected. However, before the user can go to the data collection scene, they must first select a previously saved user or create a new user. All of the hand gesture data collected will be stored in an appropriately named folder for the user. Therefore, when the user clicks on the "Enter Test Mode" the first thing that comes up is a small pop-up screen that asks whether the user would like to create a new user or select an older user; see Figure 1.2 and Figure 1.3.

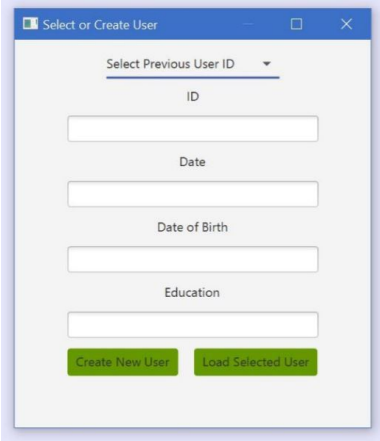

FIGURE 1.2: Pop-up window showing options to create or select user.

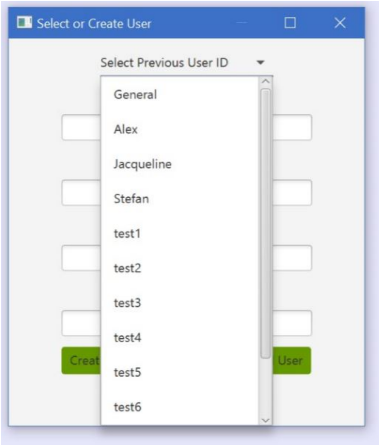

FIGURE 1.3: Pop-up window showing the previously created users.

The users shows in the dropdown menu are loaded from a CSV file which is used to record them. Whenever a user is created, a new entry is added to the CSV file for that user. As can be expected these operations are handled by using a convient User class which encapsulates all the data associated with such an object.

1.1.2 Saving User Data

After having created or selected a user, the user is taken to the screen where he/she can start to proceed to practicing the ten gestures shown for whichever left/right hand was selected on the home screen; see Figure 1.4. On this data collection screen there are five buttons: the Next and Previous buttons cycle through the gestures, the Save button saves the currently being displayed user hand, the "End Testing" button goes back to the home screen, and the Rotate button which rotates the user and target hand a full 360 degrees slowly.

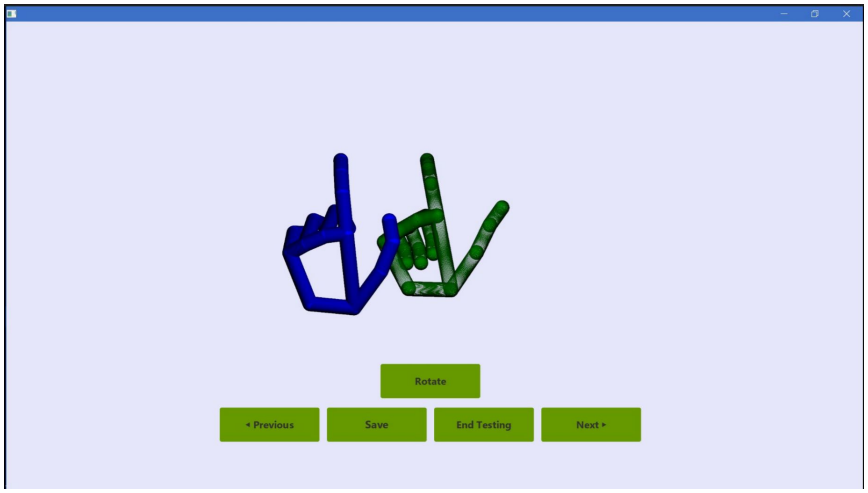

FIGURE 1.4: This scene is where the user can complete the shown gestures on the screen and the clinicians can collect the user's gesture data.

There are also some keyboard shortcuts that were coded that function in place of some of the buttons. For example, left and right arrows are mapped to the Previous and Next buttons, while the Enter key is mapped to causing the Save Gesture dialog window to pop up just as the Save button does. The user's hand will appear in full blue color on the screen, whereas the target hand the user is trying to imitate will appear in a dark green mesh material. The user's hand will be saved correctly regardless of whether it is exactly covering the target hand. The target hand is just there to give the user indication of the gesture he/she is doing. Figure 1.5 shows the Save Gesture dialog which allows the clinicians to type some comments and mark the result of the user's attempt in replicating the shown gesture.

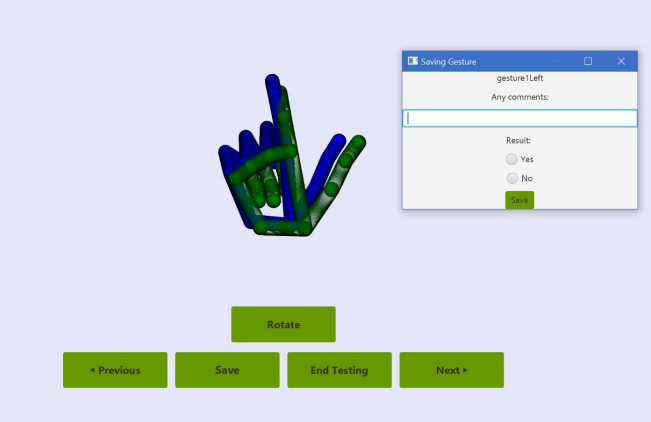

FIGURE 1.5: This dialog lets the clinicians grade and comment on the user's gesture before saving it.

By default the result is saved to be "Yes" which indicates the user successfully completed the gesture. In the code, there is an object called HandInfo which represents the data being saved including the full file path of where the Leap Motion Hand object will be serialized to, and the comments and the result of the gesture attempt.

1.1.3 Visual Rotation of Gesture

The Rotate button causes the 3D camera that is being used in the application to spin around. The user and target hands themselves do not move at all, it is the camera that orbits around them while being focused on them. A picture taken while the rotation was happening is shown in Figure 1.6.

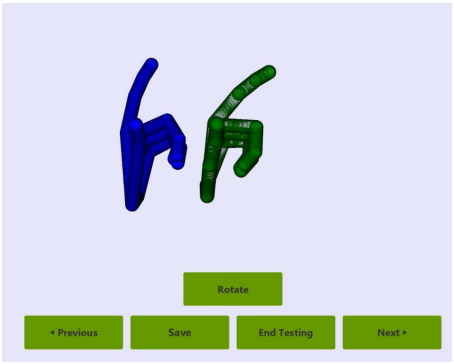

FIGURE 1.6: This figure shows instance of the rotation in action. The hands are shown at 90 degree angle because of the camera rotating around them.

The way this is achieved via code is shown in Figure 1.7. In order for the user to observe the rotation happen in real time on the screen, a Timeline object was used to set up an animation which can update the observable angleProperty() of a rotation transform object attached to the camera in advance. The timeline was set to last for seven seconds and the starting and ending angles were 0 and 360 degrees respectively. The code which actually initializes and adds the Rotate transform to the 3D perspective camera of the application is shown in Figure 1.8.

```
1  //set up rotation timeline
2  Timeline timeline = new Timeline(
3     new KeyFrame(Duration.seconds(0), new KeyValue(rotateAroundY.angleProperty(),
          0)),
4     new KeyFrame(Duration.seconds(7), new KeyValue(rotateAroundY.angleProperty(),
          -360)));
5  timeline.setCycleCount(1);
6  ...
7  //rotate button plays animation
8  rotateButton = new Button("Rotate") {
9     @Override
10    public void fire() {
11    timeline.play();}
12 };
```

FIGURE 1.7: This code shows the Timeline object that was used to animate the movement of the 3D camera around the y-axis.

Since the "rotateAround" transform object is added first to the camera's transforms, it will be the last transform to be executed. This is exactly what we want so the camera will remain focused on the hands as it rotates.

```
// The 3D camera
PerspectiveCamera camera = new PerspectiveCamera(true);
// The rotation transform to be updated later
rotateAroundY = new Rotate(0, Rotate.Y_AXIS);
// rotation transform is added first. It will be executed last
camera.getTransforms().addAll(rotateAroundY, new Translate(0, -5, -50), new
    Rotate(-10, Rotate.X_AXIS));
```

FIGURE 1.8: This code shows the rotate transform that's added to the camera. This transform gets updated by the animation timeline object.

The hands themselves are not affected at all by the rotation and in fact during the seven seconds in which the camera is being rotated around the y-axis, the user can continue trying to imitate the gesture. However, during testing of the application it was found out that most users perfer to have to rotation happen in seperately initially so they can just observe. Only after the rotation finishes is when they would start trying to perform the gesture.

1.2 Tabular Display of Data

editable picture. show code about how table was constructed. maybe even discuss scenebuilder.

1.3 Writing and Reading from CSV

1.4 Artifacts and Distribution

maybe this section can all be collapsed into one. (IDE Build Process and Batch Script)

1.4.1 Leap App Store

1.4.2 IDE Build Process and Batch Script