Performance Evaluation of Target Selection Tasks from Older Adults on Touch Screen Devices

Final Report

Jiamin He

Dec.12, 2016

School of Information Studies, McGill University

Table of contents

Project Overview

Project Overview

Project Name

Performance Evaluation of Target Selection Tasks from Older Adults on Touch Screen Devices

Description

To understand the target selection behavior on touch-screen interfaces, including the impact of finger size, angle, pressure, velocity and the three-dimensional motion trajectory

Expectation

Inform the design of new touch-based interaction techniques and increased accessibility for a wide range of users

Supervisor & Author

Karyn Moffatt, Afroza Sultana & Jiamin He

Duration

Jul 31 - Nov 15, 2016

Codes,Explanations & Layouts

Codes,Explanations & Layouts

- Login Page
- Finger Calibration Task
- Two-D Finger Calibration Task
- Two-D Fitts Task

Login Page (App)

In the LogIn page, participants are required to enter their Participant ID, which will be included in their FFitts App Results and their Leap Motion Results.
In the App:(LoginScreen.java)

```
login = (Button) findViewById(R.id.btnLogin);
pidInput = (EditText) findViewById(R.id.inpPID);
login.setOnClickListener(new View.OnClickListener() {};
```

The Onclick Listener is set to respond to the successful input.

```
createFile();createExternalFile();
createBlockFile();createPIdFile();createScoreFile();
```

Login (LeapMotion)

In Leap Motion(FingerXYZ.java), when a participant starts to record the data, he/she needs to provide their participant ID, which will be included in their Leap Motion Results.
With the same participant ID, the results can be easily synchronized and analyzed.

```
Scanner sc = new Scanner(System.in);
System.out.println("Participant ID:");
String s=sc.nextLine();
fileName = "test_results"+ "/"+ s +"_Frame.csv";
```

Login (LeapMotion)

Leap Motion creates data file and record data on frame:

```
FileOutputStream outPath = new FileOutputStream(...);
OutputStreamWriter out = new OutputStreamWriter(outPath)
out.write(...);
```

Login Page

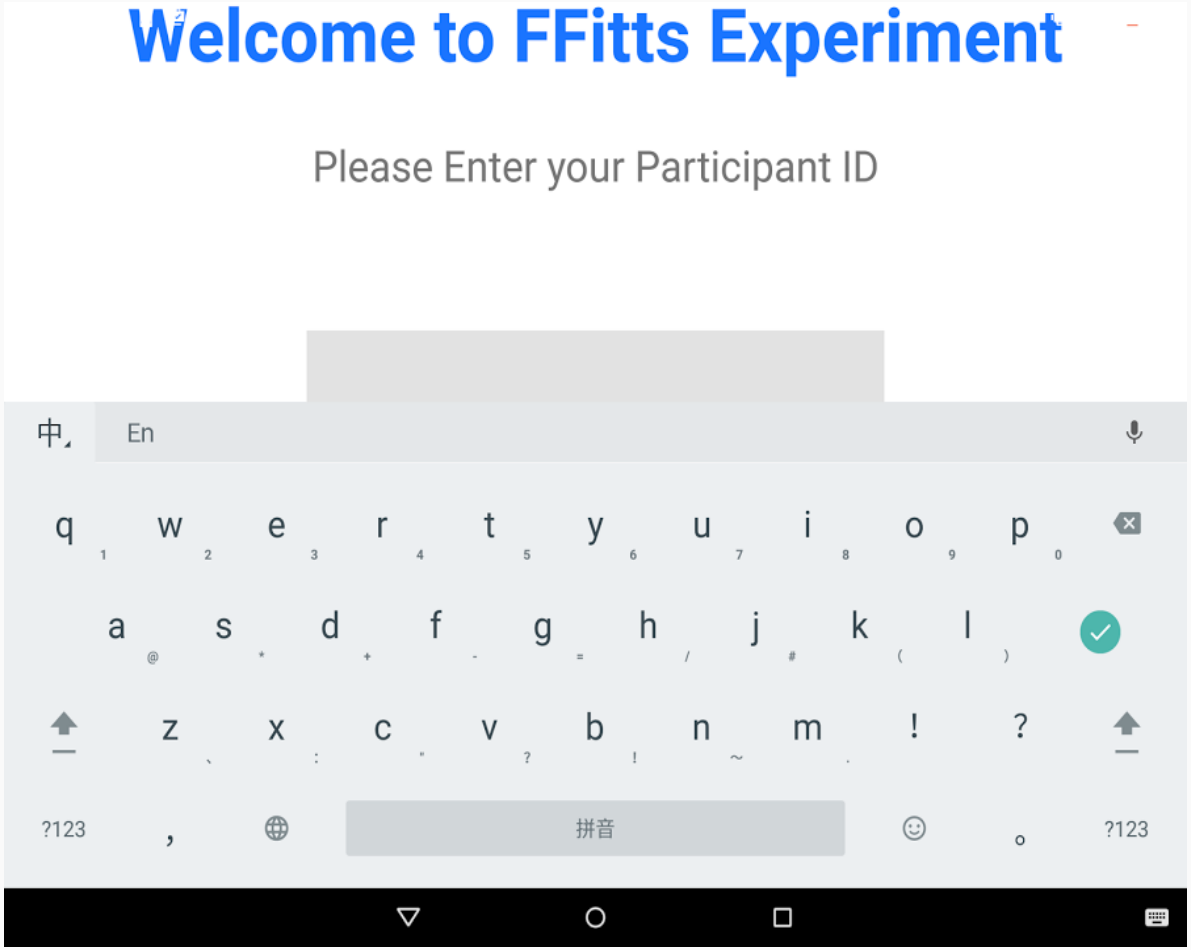

Figure 1: The welcome page in FFitts App

Codes,Explanations & Layouts

- Login Page
- Finger Calibration Task
- Two-D Finger Calibration Task
- Two-D Fitts Task

Finger Calibration Guidance Page

FingerCalibInstructions.java

set an onTouch listener for the start button, in order to start the finger calibration task

```
start.setOnTouchListener(new View.OnTouchListener(){};
```

Finger Calibration Guidance Page

when the start button is clicked, write the starTime to the file

```
"PId_" + pid + "_FingerCalibData_Internal.csv"
```

which is important when analyzed together with the leap motion data. The written startTime will help to extract data to avoid the meaningless data during the login part.

```
startTimeStamp = System.currentTimeMillis();
out.write();
```

Finger Calibration Guidance Page

Also, we read the current block from the file, and update the block so that it can successfully move on to the next block.

```
getBlock();
 block++;
updateBlock();
```

Finger Calibration Guidance Page

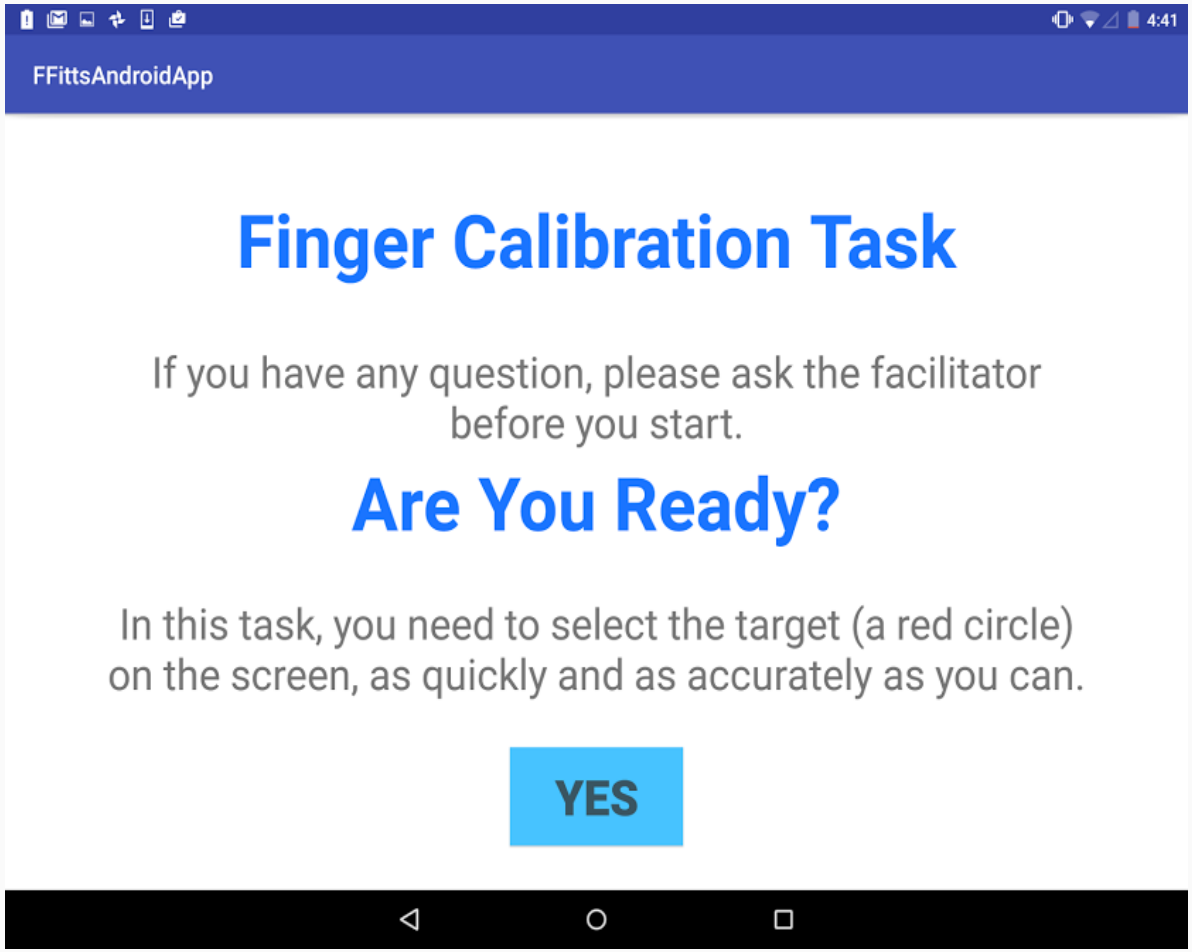

Figure 2: The guidance for the finger calibration task

Finger Calibration Task

FingerCalibTask.java

max_trial and max_block determines how many trials and blocks one should take.
targetWidths determines the width of the displayed target, in the finger calibration task, all the targets shares a certain width.[e.g., 91]
The target is regarded as an object of the class imageview, we set an onTouchListener for it to record the touchDown data and liftUp data.

Finger Calibration Task Page

```
if (motionEvent.getAction() == MotionEvent.ACTION_DOWN)
```

when the touchDown motion is detected, record the touchDown X/Y, the pressure and the timestamp.

```
touchDownX = motionEvent.getX();
touchDownY = motionEvent.getY();
pressure = motionEvent.getPressure();
touchDownTimeStamp = System.currentTimeMillis();
```

Finger Calibration Task Page

```
if (motionEvent.getAction() == MotionEvent.ACTION_UP) {}
```

When the liftUp motion is detected, record similar data.
Also, check if the participant hit the target successfully or miss the target.

```
private boolean isSelectionInsideTarget(...) {
if (distance(liftUpX, liftUpY, targetX, targetY)
<= targetWidth / 2)
return true;}
```

Finger Calibration Task Page

if succeed, play the right sound, ”bling”, to give the participant some positive feedback.
And increment his/her score, and mark the select as 1.
(”Select” is used to represent a successful trial (succeed in the first trial) or unsuccessful one.)

```
private SoundPool sp;
sp = new SoundPool(10, AudioManager.STREAM_ALARM, 5);
//SoundPool(maxStreams, streamType, srcQuality)
right = sp.load(this, R.raw.right, 1);
sp.play(right, 1, 1, 0, 0, 1);
```

Finger Calibration Task Page

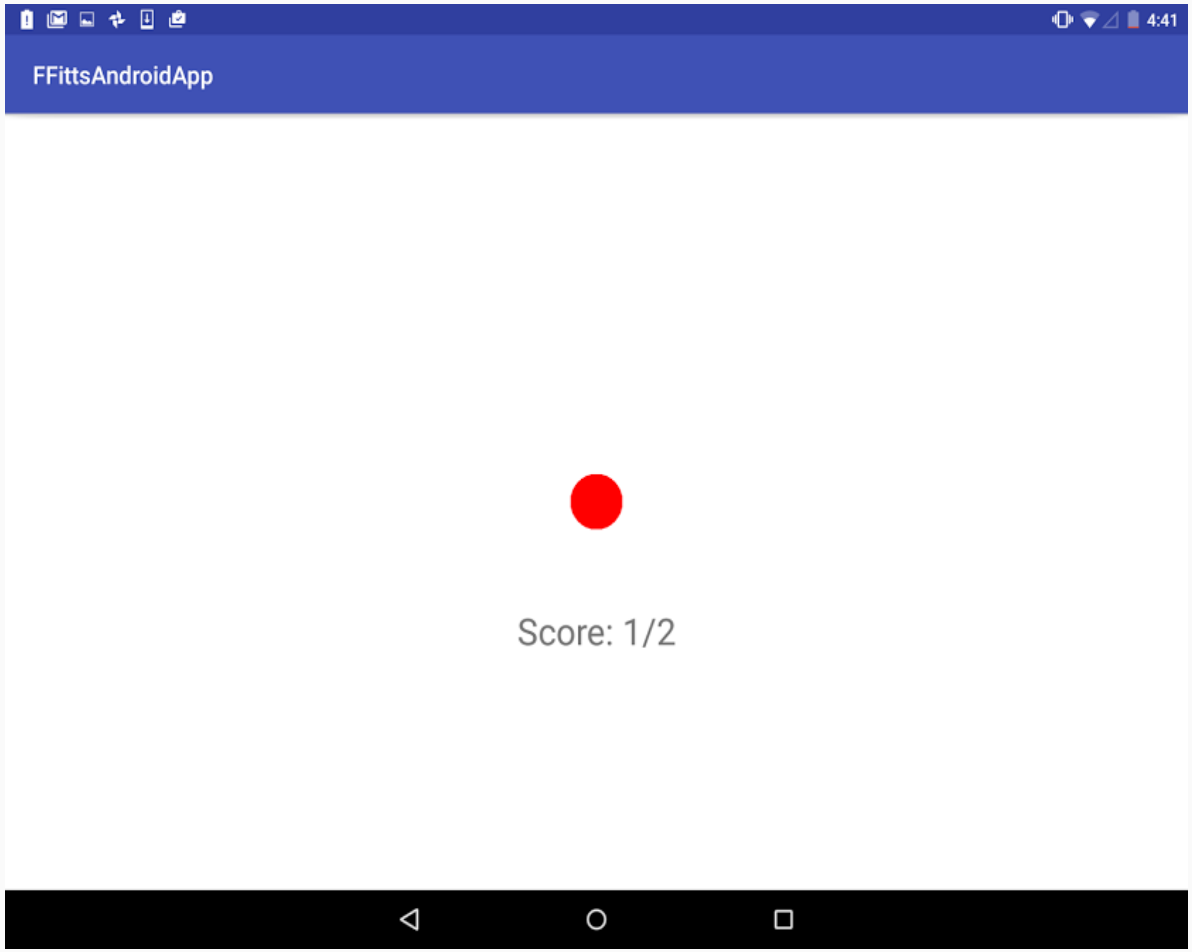

Figure 3: The finger calibration task

Finger Calibration Task Page

When the first trial fails, mark select as 0 and play the wrong sound.

```
private void doAfterTouch() {};
```

```
if (trial < max_trial)
```

update the real-time score, draw a new target, write the relevant data in this trial to the internal and external file.

```
if (trial ≥ max_trial)
```

write the final score for the current block into score.txt.

Finger Calibration Task Page

And then start next block or next task[show the task score and email the result.]

```
private void emailResult() {
 Mail email = new Mail();
email.addAttachment();
};
```

Finger Calibration Task Page

And then start next block (NextBlockFingerCalib.java)
or next task [show the task score and email the result.]
(FingerCalibToTwoDCalib.java)

```
private void emailResult() {
 Mail email = new Mail();
email.addAttachment();
};
```

Finger Calibration Task Result

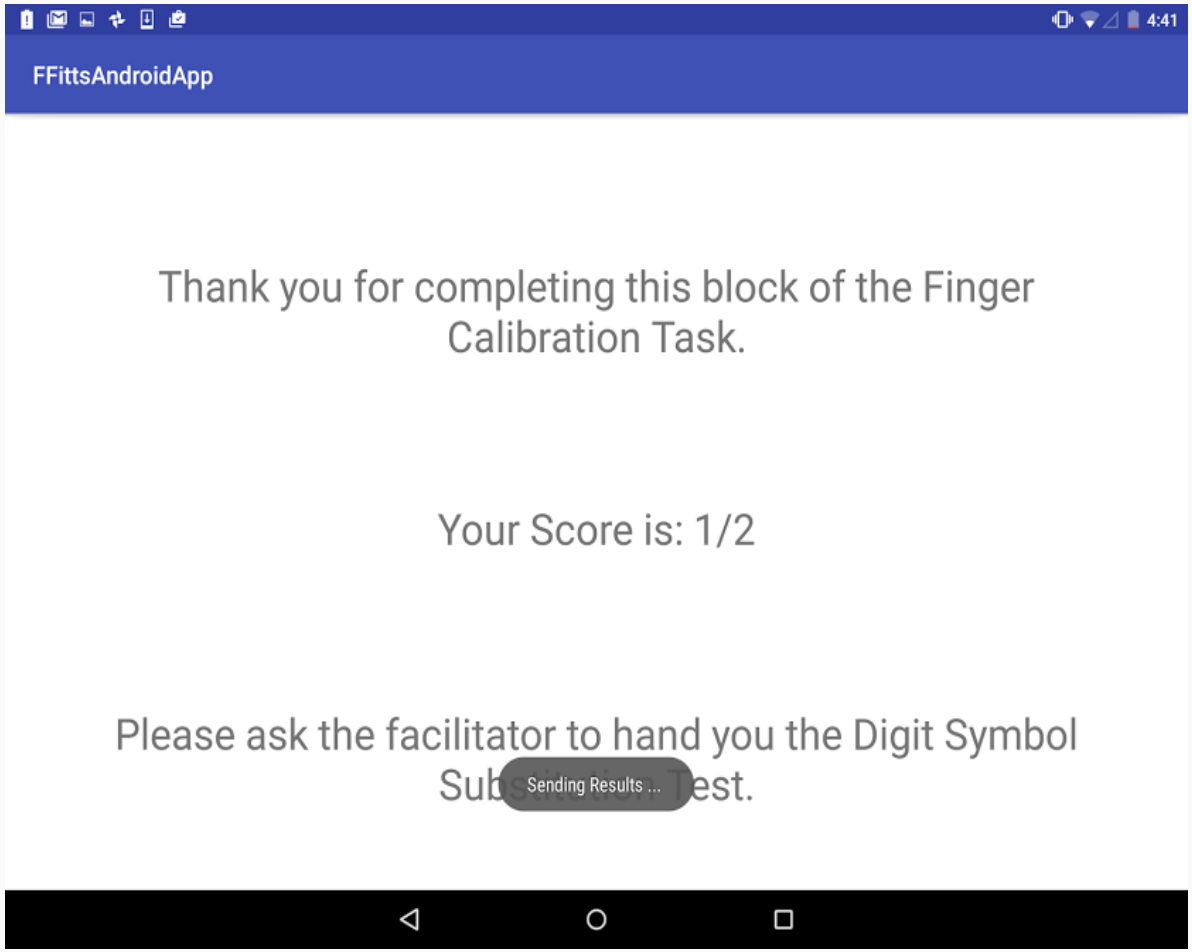

Figure 4: showing the participant's score after finger calibration task

Codes,Explanations & Layouts

- Login Page
- Finger Calibration Task
- Two-D Finger Calibration Task
- Two-D Fitts Task

Two-D Finger Calibration Guidance

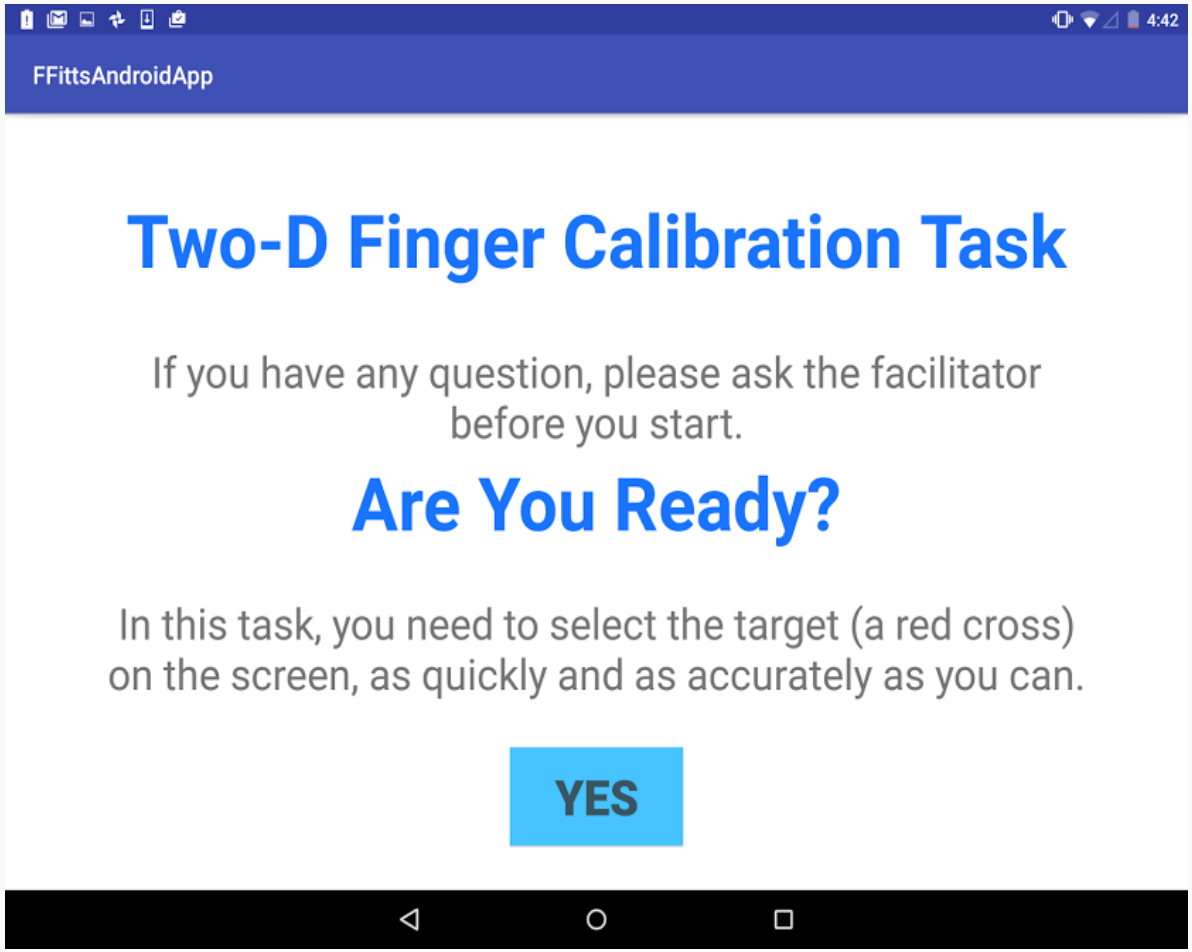

Figure 5: The Guidance for the two-D finger calibration task

Two-D Finger Calibration Task

TwoDCalibTask.java

To record the start time –>

```
WriteStartTime();
```

2 different widths for the targets–>

```
double[] targetWidths = {270, 378};
```

This is used to generate random coordinates combination for different trials–>

```
double[][] targetCor =
{{0,0},{0,-1},{0,1},{-1,0},{1,0}};
if (i < 3) {
targetY += targetCor[i][1] * targetWidths[0];} else {
targetX += targetCor[i][0] * targetWidths[1];}
```

Two-D Finger Calibration Task

when touch down, check if the hit is successful,

```
if (isSelectionHit(touchDownX, touchDownY)) {};
```

when succeed, start the timer, (play the sound: one, two, three)

```
mp = MediaPlayer.create();
mp.start();
```

when the participant hit the target and hold on for three seconds, that is a total success.

```
 mp.setOnCompletionListener();
 onCompletion(MediaPlayer mp){
select = 1;
sp.play(right, 1, 1, 0, 0, 1)};
```

Two-D Finger Calibration Task

when lift up, calculate the RelativeLiftUpXfromTarget, RelativeLiftUpYfromTarget, RelativeTouchDownXfromTarget, RelativeTouchDownYfromTarget. These calculation helps the data analysis.

```
RelativeLiftUpXfromTarget = liftUpX - targetX;
```

if doesn't hold on for three seconds, (just one or two seconds), it does not provide enough precision for data analysis. So it is regarded as a failed trial. Mark select = 0.

```
mp.pause();
sp.play(wrong, 1, 1, 0, 0, 1);
```

Two-D Finger Calibration Task

only after a total success (three seconds), a participant can move on to the next cross calibration.

```
 if (!(i < 4)) {
// different i corresponds to different cross locations
mp.release();
startActivity();} else {
doAfterTouch();
i++;
drawTarget();}
```

when finish all different cross location calibration, we will move on to the next part. TwoDCalibToTwoDTask.java and TwoDInstructions.java.

Two-D Finger Calibration Task Page

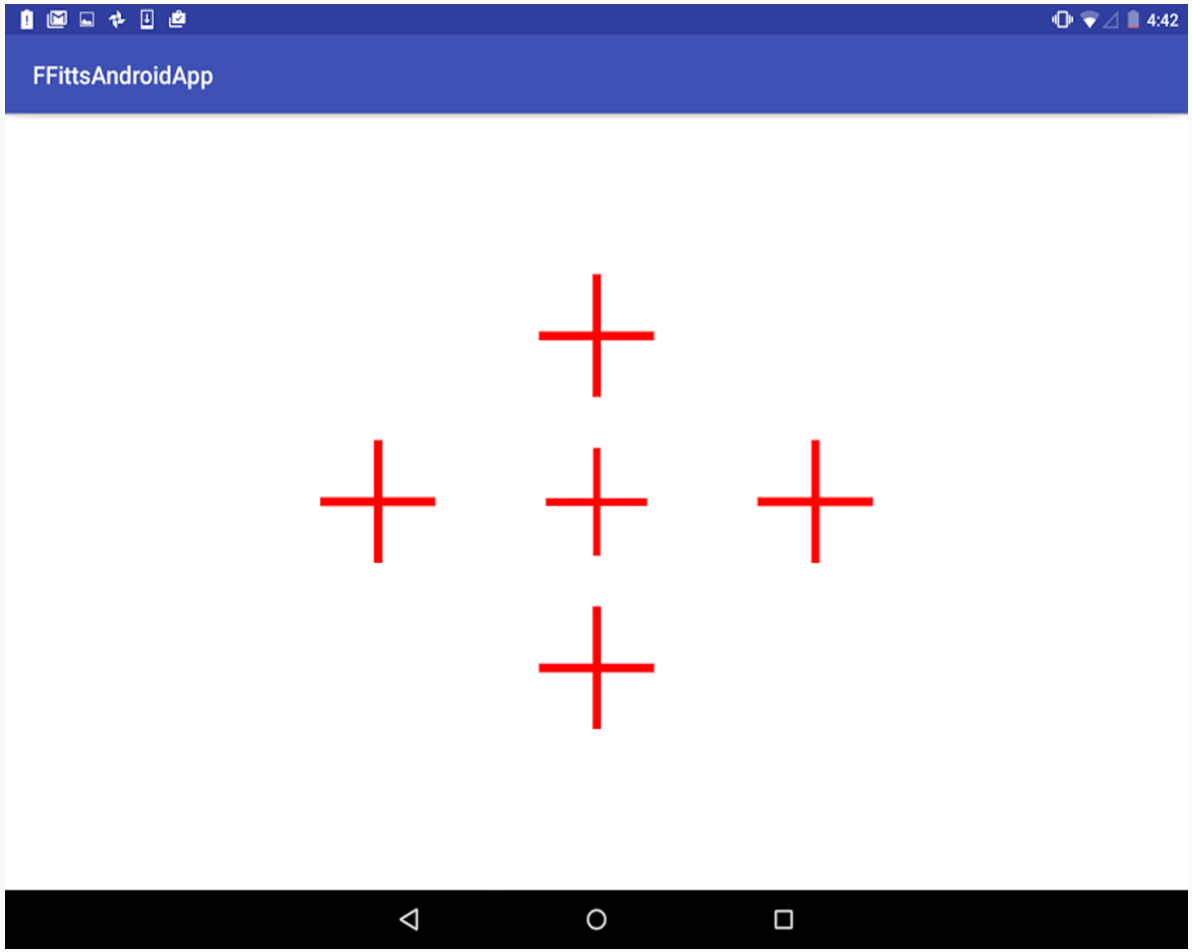

Figure 6: The Two-D Finger Calibration Task

Codes,Explanations & Layouts

- Login Page
- Finger Calibration Task
- Two-D Finger Calibration Task
- Two-D Fitts Task

Two-D Fitts Task Guidance

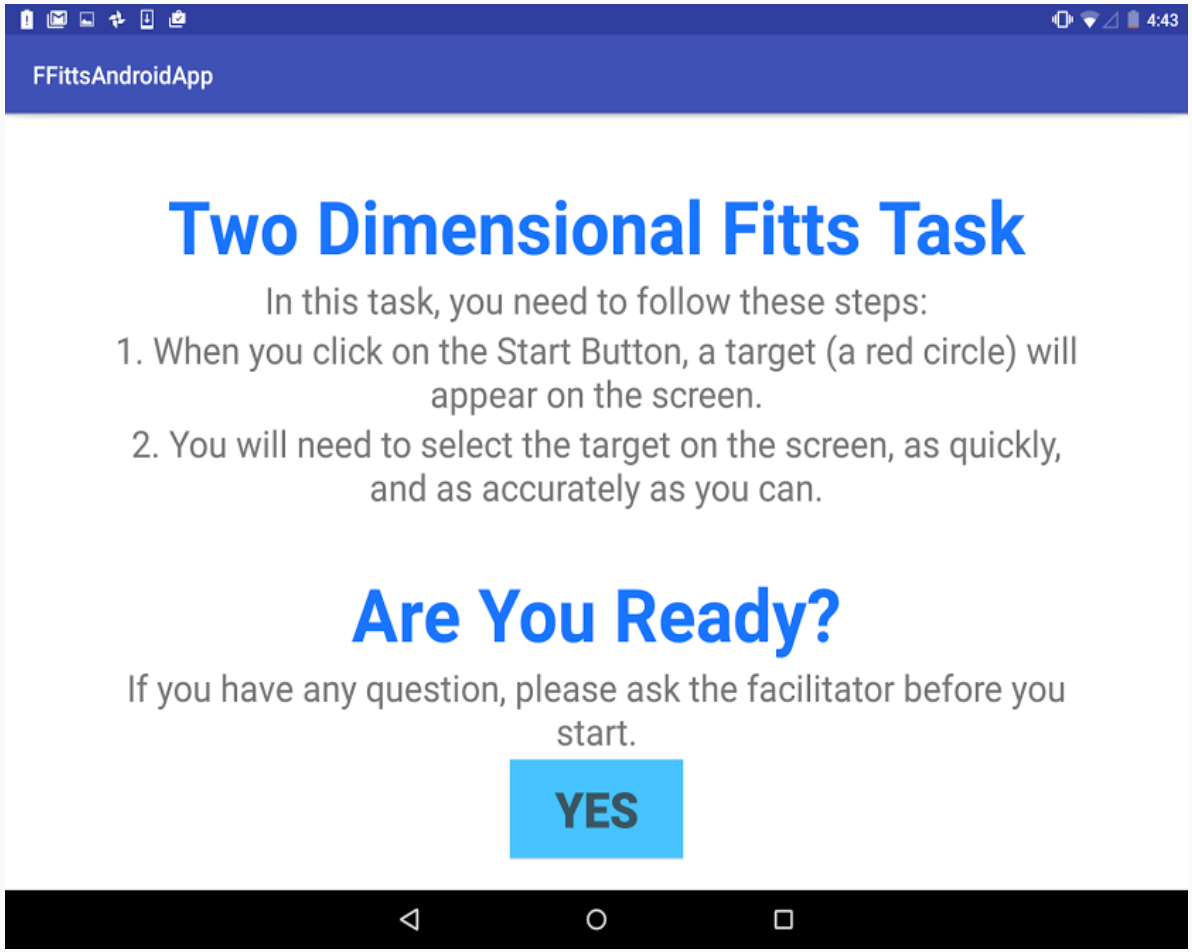

Figure 7: The guidance for the Two-D Fitts Task

Two-D Fitts Task

Ensure the random combination of different targets –> targetAngles, targetDistances and targetWidths.

```
targetAngles = {0,45,90,135,180,225,270,315};
targetDistances = {270, 378};
targetWidths = {86, 110, 158};
```

The start button is regarded as an imageButton.

```
ImageButton btnStart;
btnStart.setOnTouchListener();
```

Two-D Fitts Task

There are different kinds of errors during the analysis part –> error, SlipError, NarrowSlipError, ModerateSlipError, LargeSlipError, VeryLargeSlipError, MissError, NearMissError, NotSoNearMissError, AccidentalTap, OtherError, AccidentalHit.
We also have entry, firstreEntry, TRE.
They are all initialized to 0.
When the start button is clicked,

```
calculateStartCenter();
calculateScreenProperties();
drawTarget();
```

The start button for each trial

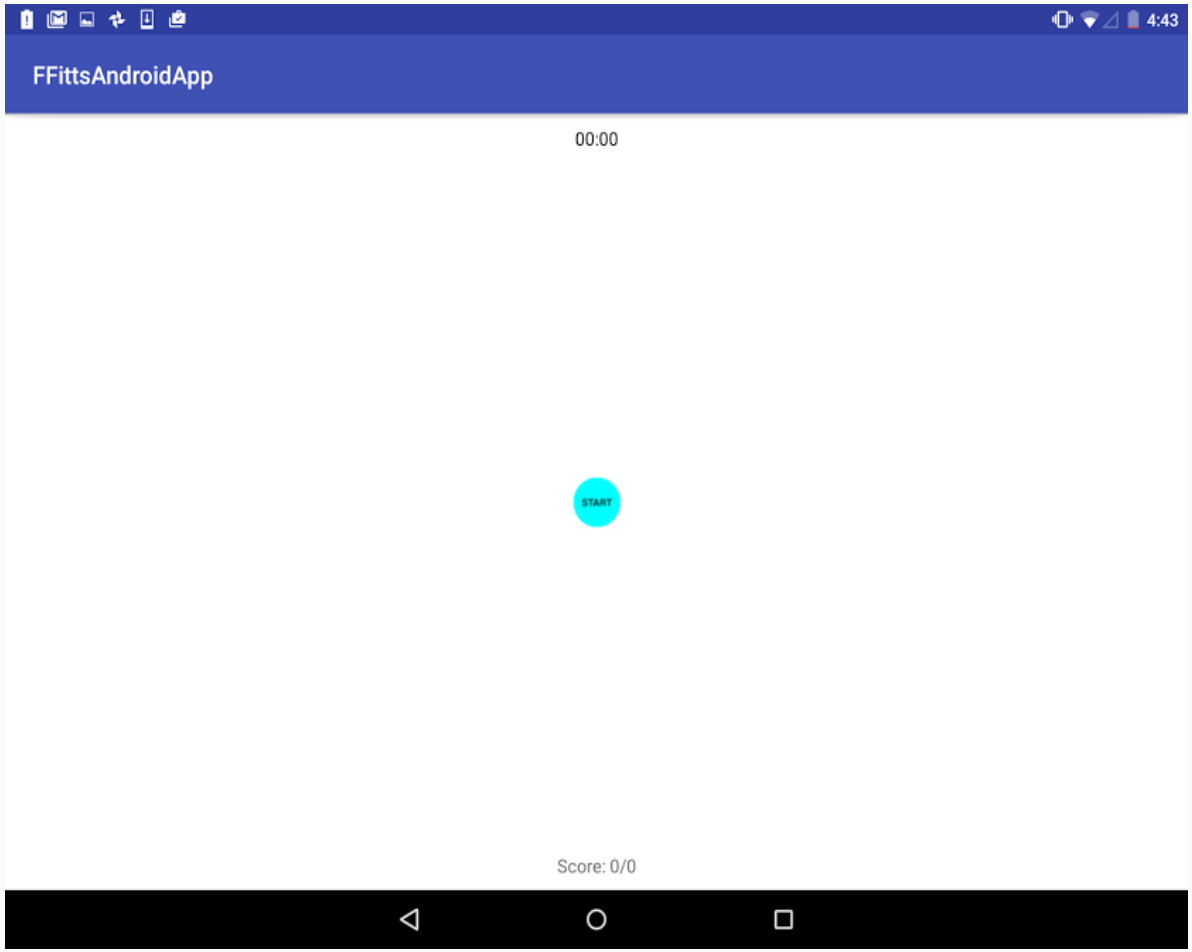

Figure 8: The start button

Two-D Fitts Task

During the trial to hit the target, detect the submovements of the participant.

```
if (event.getAction() == MotionEvent.ACTION_MOVE){};
```

By comparing the last X/Y and the current X/Y, We can calculate the TRE of each trial.(by calculating the entry times.)

```
if(isSelectionInsideTarget(CurrentX, CurrentY,0.5))
{if(isSelectionInsideTarget(lastX,lastY,0.5)) {}
else{entry++;}}
```

Two-D Fitts Task

TwoDFittsTask.java

During the trial to hit the target, detect the submovements of the participant.

```
public boolean isSelectionInsideTarget(x,y,z){
if(distance(x, y, targetX, targetY) <= targetWidth*z)
return true;}
```

z is the percentage of the target width. When z=0.5, this method functions the same as a method calculating the distance.
We also have isSelectionOutsideTarget(double x, double y, double z), z has the same meaning with isSelectionInsideTarget.

Two-D Fitts Task

When lift up, we can easily calculate the reEntry for each trial.

```
if(entry>0){
reEntry = entry-1;}
else{reEntry=0;}
```

But as we pay more attention on all first trials because they share the same start point. So we can calculate TRE –>

```
TRE = (float)firstreEntry/attempt;
```

Two-D Fitts Task

When touch down, we also record the touch down timestamp, which is used in the results feedback and the data file.

```
TouchDownTimeStamp = System.currentTimeMillis();
CurrentTimeTouchDownTimeTaken =
TouchDownTimeStamp - startTime;
```

If attempt ==1 –>

```
FirstTouchDownTimeTaken = CurrentTimeTouchDownTimeTaken;
FirstTouchDownTimestamp = TouchDownTimeStamp;
```

For the same reason, we can easily get different first and final touch down time taken for each successful or unsuccessful trial.

Two-D Fitts Task

When touch down, we also record the touch down timestamp, which is used in the results feedback and the data file.

```
TouchDownTimeStamp = System.currentTimeMillis();
CurrentTimeTouchDownTimeTaken =
TouchDownTimeStamp - startTime;
```

If attempt ==1 –>

```
FirstTouchDownTimeTaken = CurrentTimeTouchDownTimeTaken;
FirstTouchDownTimestamp = TouchDownTimeStamp;
```

For the same reason, we can easily get different first and final touch down time taken for each successful or unsuccessful trial.

Two-D Fitts Task

To analyze the errors when lift up.

```
if(isSelectionInsideTarget(touchDownX,touchDownY,0.5) )
```

When z ==0.5 –> know slip error and miss error.
When z ==0.75 –> know narrow slip error and near miss error.
When z ==1 –> know moderate slip error and not so near miss error.
When z ==1.5 –> know large slip error, very large error, other error and accidental tap.

```
FirstTouchDownTimeTaken = CurrentTimeTouchDownTimeTaken;
FirstTouchDownTimestamp = TouchDownTimeStamp;
```

For the same reason, we can easily get different first and final touch down time taken for each successful or unsuccessful trial.

The Target in the task

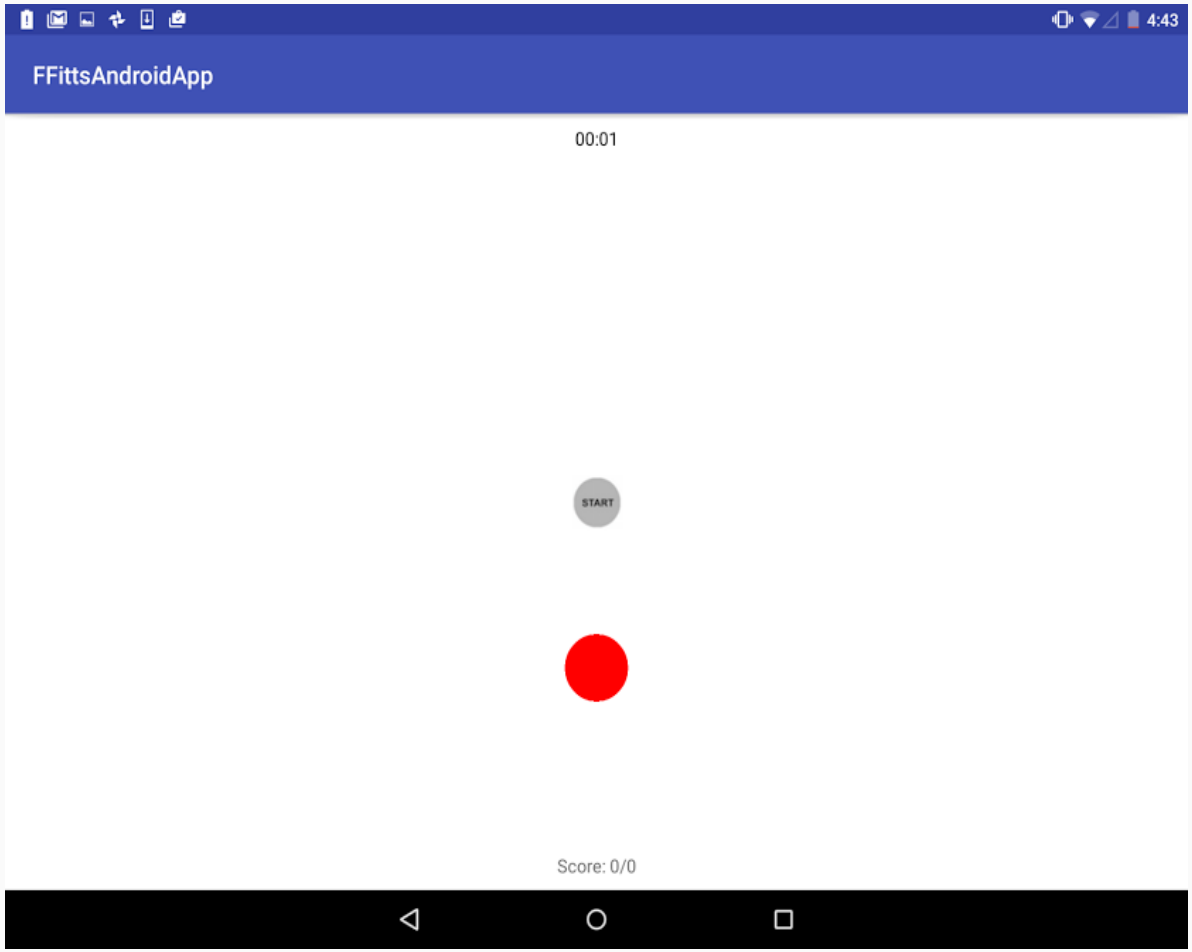

Figure 9: The red target to be hit

Feedback for the task

TwoDFittsTask.java

When touch down, we also record the touch down timestamp, which is used in the results feedback and the data file.

```
if(isSelectionInsideTarget(liftUpX, liftUpY,0.5)){};
touchDownAll += (FinalTouchDownTimeTaken/attempt) ;
liftUpAll += FinalLiftUpTimeTaken/attempt ;
```

Feedback for the task

LastScreen.java

give the participants feedback of their performance.

```
touchDownTime = safeLongToInt(touchDownAll/max_trial);
liftUpTime =safeLongToInt(liftUpAll/max_trial);
touchDownAverage.setText(touchDownTime + "ms");
liftUpAverage.setText(liftUpTime +"ms");
```

Write the end time when finish all tasks, so that it will help to extract data from Leap Motion file.

```
WriteEndTime();
```

Feedback for the task

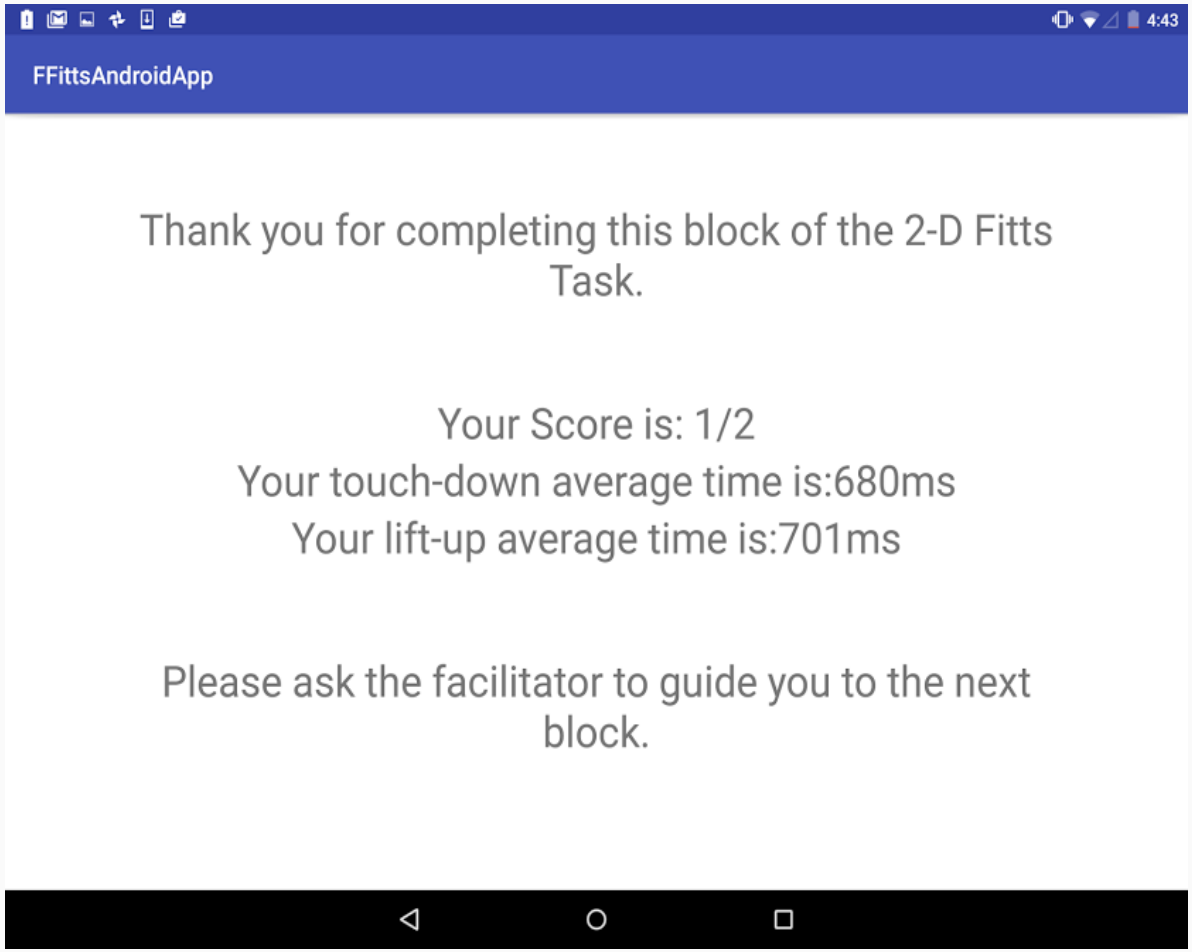

Figure 10: Showing the results, work as a feedback for the participant

Leap Motion Brief Introduction

onFrame(Controller controller)

Destop/FFitts/src/ffitsLeapMotion/FingerXYZ.java

From the frame data, we can get hand data and finger data. During five fingers, we pay more attention to the index.

```
Finger index = frame.fingers().fingerType(
Finger.Type.TYPE_INDEX).get(0);
if(index.isValid()){...};
```

When index is valid, we can extract different vectors, e.g. tipPosition, direction, stabilizedPosition, speed, boneDistal1...

onFrame(Controller controller)

Sometimes it will not respond and not work effecitively. To avoid this situation,

```
System.out.println( "Frame id: " + frame.id());
```

When it works well, we can see each Frame ID on the screen, so that we can confirm it is in the work mode.
To start using it to record the data, a participant should offer his/her ID and it will automatically create a file connected to this ID. –>

```
Scanner sc = new Scanner(System.in);
System.out.println("Participant ID:");
String s=sc.nextLine();
fileName = "test_results"+ "/"+ s +"_Frame.csv";
```

Thank you!