Scott Crawford

Odyssey Project: Android Emulator Documentation

**GitHub Repository:**

<https://github.com/Scott-Crawford/Odyssey/tree/master>

**Current Functionality:**

In order to run the android emulator, you will need a physical android device, a set of Odyssey emulator controllers as would be used for the desktop version of the emulator, and an adapter from USB to either USB-C or USB-mini, depending on your android device. This emulator has only been tested on a Pixel 1 device with the USB to USB-C adapter that came with the Pixel device.

At this point, the android app is running as a full screen mode and accepting serial + keyboard input from the controllers to move the player 1 and player 2 ImageViews. While no controller is connected, the two views will be placed at coordinates (200, 300) and (425, 300), and a toast will show up saying that no USB is connected. Once a USB device has been connected, a toast will notify the user of that and the android device will request user permission before accepting the serial input, at which point the views will be set to the locations determined by the positions of the knobs. If the controllers are ever disconnected from the device, a toast will notify them and the views will be set back to their neutral coordinates.

Moving the players’ horizontal and vertical knobs will move their corresponding views along the screen, although some jumpiness is visible, probably due to bad threading, which can be improved in the future. The controllers send several lines of values separated by commas, which are then separated into a string array split by the new line character. The first line is then split by the comma, and stored in another string array, which is then passed to a function for moving the views. Currently, only 4 of the 8 values of the string array are being parsed, each one corresponding to an x or y value for a player. Once those values are parsed, setX and setY methods are called for the views. The app is currently showing toasts of the serial input values every time it receives an message from the UsbService. These can easily be commented out in the myHandler code.

The keyboard presses on the controller are currently being accepted, though at this point they are simply displaying toasts of what that button press should do on the key up action, such as the reset button sending a toast saying “Reset.” Since all six buttons are related to either the ball or the cards, no functionality for them as been added beyond the toasts.

A ball view has been created, but the view’s visibility has been set to invisible for this version of the app. Additionally, the ball’s English value is being read in from the device, but not currently used in any way.

**Libraries Used:**

In order to read the serial input from the controllers, the app is using the UsbSerial library (<https://github.com/felHR85/UsbSerial>) as created by Felipe Herranz (felHR85), which is licensed under the MIT license.

In order to make this version of the app, I used his example of creating and using a UsbService from the library, as outlined in his SerialPortExample repo (<https://github.com/felHR85/SerialPortExample>). The service that the app is currently using provides a lot of extra functionality that can be removed later, such as the ability to send a message to the USB device.

**Methods and Variables:**

player1View – ImageView for player 1. Appears as a 50dp by 50dp white square.

player2View – ImageView for player 2. Appears as a 50dp by 50dp white square.

ballView – ImageView for the ball, currently set to invisible. Appears as a 25dp by 25dp white square.

p1H, p1V, p2H, p2V – Integer values saving the horizontal and vertical values of players 1 and 2.

handleMessage() – Receives a message from the UsbService, which is then broken down into useable values and calls movePlayers(). Also presents a toast saying the received serial input.

movePlayers() – parses the values received by handleMessage() and uses those values to set the x and y coordinates of players 1 and 2.

onKeyUp() – Automatically called whenever a keyboard event occurs, which is pressing any button on the controller. Currently uses a switch case to find the button pressed, then presents a toast saying what that button should do.

**Future Work:**

The most obvious bug in the program as it stands is the jumpiness of the player views, which can hopefully be solved with some multithreading. Additionally, if a perfect emulation of the original Odyssey console is our goal, I believe we may also need to allow the views to disappear from the screen completely. Along with that, even though the player views are currently reaching the edges of my android device, it may need to be refactored to work with all devices.

The next goal would be to get the ball working, along with its associated values such as speed and English. After that, I would work on collision detection between the different views. At that point, most of the functionality of the game has been completed, so it’s just a matter of implementing the different game cards and adding inertia to the player views when necessary.

Future testing will also need to include other android devices. Since using the android virtual device is essentially impossible for this project, as it has no way of detecting what USB devices are plugged into the computer, we will need physical devices of the most popular brands to do proper testing.