Team member list: Craig Hunter, Mitchell Smith, James Wang

Pokemon name: Furret

Project/App Title: Pi Pi Revolution

Project Pitch (one paragraph describing what the app does): Our app is a rhythm game. After connecting to the Pi, you can select a song to start playing. The Android device will play music, and the LED strip on the Pi will light up to music. The lights will appear at one end and move down to the other one. Once they reach the end, press the matching color on the tablet to be awarded points and increase your combo! Post your results to Twitter! (@pipirevolution)

List of key features:

Letting you learn to groove!

Festive spirit!

Exciting and addictive gameplay!

Social networking!

Basic instructions on usage:

1. Select Settings, and input the Raspberry Pi URL/IP address. Click done.
2. Select Play. Click a song.
3. Press the matching colored button when a light reaches the end of the strip
4. Post to Twitter, retry, or pick a new song!

How you use the lights: Uses Raspberry Pi LED strip to show visually what buttons to press at what time.

How you use the sensors: Shake protection feature protects your game in volatile environments (customizable) by using the accelerometer to detect motion

What your third party web service is and how you use it: Posting scores to Twitter.

Any special info we need to run the app (username/passwords, etc.):

Links to your final APK: [INSERT LINK]

Lessons learned (at least half a page):

We learned a lot about Android development. One of the first things we noticed was that we couldn’t run our logic code in the same thread as UI. This caused the graphics to lag significantly, rendering our prototype unplayable. We came to appreciate threading after that. After spending 2 hours trying to put 3 buttons in a line with padding on the outside, we came to appreciate Android styles and layouts, such as when to use Relative and Linear layouts to shape our View to look like we wanted it to.

Since we were making a rhythm game, the ability to sync notes and music was crucial. Having things on different threads happen at the same time proved more difficult than initially anticipated. In order to increase performance to make sure the music was in sync with the notes, we had to decrease unnecessary CPU utilization. By minimizing new objects created in loops, we were able to decrease CPU cycles used on memory allocation and freeing.

Because we had several settings that the user would change, we had to figure out how to move data across Activities. We made heavy use of the SharedPreferences manager, and occasionally packaged information with the launching of our Activity in Intents.