Michael Camara

Honor Code Pledge: This work is mine unless otherwise cited

CMPSC 382

Due Date: 11/30/15

Final Project: Progress Report

In order to build an Android game for this final project, I first had to familiarize myself with new libraries, new syntax, and a new IDE for Android development. I initially followed instructions outlined in *The Beginner's Guide to Android Game Development* by James Cho, which guided me through the convoluted process of creating an application from scratch. I installed an Eclipse plugin for the Android software development kit (SDK), which created a number of necessary directories, libraries, and other files for my project. I then proceeded to study the XML files that Android uses when running an application, which help ensure the interoperability of the application on different devices, in addition to controlling various aesthetic options. Next, I followed some lengthy, thorough examples that Cho uses in his book for converting a game from Java on a personal computer to the Android platform on a mobile device. This allowed me to understand some of the key differences between the two and some of the important classes and methods that I would need to utilize in the future.

Ultimately, familiarizing myself with the Android platform has occupied most of my time for this project so far; but I have managed to create a functional application with many of the key elements I want to implement. While Cho creates an extensive framework for his game, I realized that I could cherry-pick some of the most important elements in order to simplify the overall design of my system, and to avoid inadvertently copying any of the code he provided. In general, Android uses “activities” to separate different windows of an application, similar to how

one might use a JFrame for a window using the Java2D library. These activities are further governed by XML files that specify the layout of the window and any widgets contained therein. One can then put any number of “views” inside of an activity, perhaps similar to a JPanel from Java2D. These views can belong to many different subclasses with special purposes, but they generally handle the lower-level details of program execution while the activity handles the overall look-and-feel of the window itself. By using these two key classes, activities and views, I can abstract away many of the complicated details of Android development and focus on the actual implementation of this project. This should further afford me room for growth in the future if I decide to extend this project beyond the scope of this class.

Although the names and organization of my project will likely change in some way, these are the primary components I'm working on. First, I have a MainActivity class and a SecondActivity class, corresponding to a title screen and game screen, respectively. Their main purpose is simply to set up the window and to allow the user to switch between them. These classes rely on XML files to determine their layout, although there are still some modifications that I will need to make. I then have a CustomView class that extends the standard Android view and handles user input. This class contains a method called onDraw(), which will behave similarly to the draw() method in Processing, executing continually throughout the program once control has been passed to this view. It additionally has an onTouchEvent() method that will retrieve the X and Y coordinates wherever a user has touched the screen.

While the previous classes provide the overall framework for the game, the classes in my models package comprise the visible components of the game itself. The actual game will proceed as follows: the user controls a horse on-screen that moves wherever the user touches. While the screen is being touched, the horse will shoot lasers from left to right. Additionally,

colored orbs will periodically appear, moving right to left, and the user is supposed to shoot all of the orbs before they either touch the horse or hit the left side of the screen. Additional features may be added, but this is the basic scope of the game. It then follows that I created classes to encompass each important element, including: Horse, Laser, Orb, and Background.

I made the Horse class to mimic the horse image I created in a previous lab, although it involved a bit of refactoring with the Android API. Drawing anything in Android typically uses two core classes: Paint and Canvas. Paint controls the color, style, and format of the "brush" being used, while Canvas is used to actually display something on the user's screen using the properties specified in Paint. Additionally, in order to create a custom shape, Android uses the Path class to indicate specific vertices to connect in various ways (e.g. using Bezier curves, simple arcs, or straight lines). By replacing the beginShape() syntax used in Processing with different Path objects and small changes, I was able to roughly recreate the horse image, which serves as the focal point of the game itself. Using a translation transform in the GameView class, the horse will appear wherever the user touches the screen and follow along as the user drags it anywhere.

Next I created a Laser class, which creates bright red rectangles that originate from the location of the horse's eyes and gradually move to the right. The Orb class similarly creates circles that originate at a random Y position on the right edge of the screen, gradually moving left. Additionally, I created a MyColors utility class that provides static colors I created using HSV values. The Orb class uses this utility class to create highly saturated orbs that will stand-out from the background. The Background class then uses a gray color to provide sufficient contrast to all the elements on top of it. Finally, there is a CollisionDetector class that is used to determine when an orb has contacted a laser or the edge of the screen. When either event is

detected, the orb will “burst,” creating a large, desaturated circle below it on the background. This might be equated to a colorful water balloon bursting and then creating a splash pattern below it. This creates a dynamic and colorful background that is still easily distinguishable from the foreground elements due to the large difference in saturation values.

There are still a number of elements I hope to add to the program, using some of the techniques and theories we've learned in class. For instance, I hope to implement a “power-up” that uses some preattentive process to grab the user's attention (e.g. it could be a different shape, move in a different direction and flicker). I also hope to display some on-screen statistics as the user plays, perhaps mimicking a brushing and linking interaction. I may also try to incorporate more image transformations, although the constrained resources of a mobile device might limit what I'm able to achieve while maintaining acceptable performance. Regardless, I believe that I've made considerable progress since starting this project, and I believe I should have enough time to implement and polish these features before the due date.