1 Introduction

Our project creates a game for Android devices in which the user creates his own virtual biological virus and attempts to spread it in his local area as widely as possible. This spread is implemented as a probabilistic occurrence through Bluetooth and GPS to other local devices, with the likelihood of success coming from a couple of fields which the user can upgrade. These upgrades come at the cost of upgrading other traits, which would improve likelihood of, for example, taking over a host already containing a virus, spreading to a wider geographic region, and surviving as another virus attacks its same host.

This project was decided upon due to interest in working with phone hardware features such as Bluetooth and GPS, and with implementing communications with the central server. It features some opportunities to work on object-oriented programming, but focuses mostly on the more novel sides of app development, namely graphics, phone sensors hardware, and cloud based features. It seems also a project in at least some small demand, as recently the sort of virus-building game has become its own genre; however, none of the apps already thriving on the Google Play Store make use of actual physical proximity and infection methods, or interact with other players.

2 Distribution of Labor

The project we divided into three parts: the front-end, back-end, and the server side. This worked rather well with the relative strengths of our three group-members. The project lead was also an interface designer and the lead processing designer, although we separated the back-end into two components. The GUI designer also did interfacing and documentation. The remaining group member was needed on back-end server-side engineering throughout the course of the project.

3 How it Works

As mentioned above, there is a virus object which infects a host ID corresponding to a game user. The host ID is saved on the server with a list of virus IDs which have infected it, and the device corresponding to a host ID computes locally every 15 minutes how the various viruses which infected it are interacting. This process runs in the background after the app has been closed, and again, the purpose of virus interaction is that the different viruses within a host may kill off each other. The server’s list of host IDs and corresponding infections is used to push to the user statistics concerning how many hosts he has infected.

On the front end, the only complication is the dynamic generation of card views, a new View object defined and created in-app, which display on a local device an overview of each virus which currently infects it. All other display aspects were handled using standard Android display elements.

4 Challenges and Remarks

One anticipated challenge of maintaining organization and efficiency was overcome through good planning of time and distribution of labor, and through effective use of tools. GitHub was used for versioning, and that and use of the powerful features in the Android Studio IDE greatly facilitated design. The most significant obstacle for us was time, as each member of the group was seeing opportunity for features which would even further enrich the app, which could not be implemented in the time each member had available. This, of course, could have been alleviated by beginning work sooner.

The newness of app design was a smaller issue, although certainly present, as we had certain team members entirely or mostly unfamiliar with Java and XML, who nonetheless had to understand those environments; good use of online resources allowed everyone to adapt fairly quickly to unfamiliar circumstances, and sharing knowledge allowed each of us to avoid getting stuck on any one problem for very long.