INTRODUCTION

Assassins is a real-life game played by a group of players (n -> 2) in which each player ("assassin") is assigned another player as a target, and must "kill" that player by tracking him or her down in the real world, and performing some special action which signifies a kill. Because of its status as a social game, Assassins is well-suited to an implementation as a mobile app, since traditional gameplay mechanics can be augmented with the networking and location capabilities of mobile devices in an integrated and entertaining matter.

GAME MECHANICS

Our implementation of Assassins, entitled “Hitman: Android,” consists of three major gameplay mechanics. In general, we initially give each assassin very little information about their target (just the nickname with which they register for the game, which is practically meaningless). Then, through the first two mechanics, we gradually give each assassin more information about their target, forcing them to make inferences to figure out who their target is and how to track him or her down. The mechanics are:

1. Periodic Location Updates: players will receive push notifications about their targets' inexact location, e.g. "your target is near -x building+-", at intervals of several hours. Thus, the assassin must still do some, if not most of the detective work on their own, which ensures that the electronic component of the game does not overshadow the physical component.

2. Picture-taking: To help give the assassin additional clues about who their target is and where they are, we will prompt players to take pictures of people around them when other players (not their targets) are nearby; these pictures will then be sent to the assassins. Since players taking pictures will not know the exact direction to the person they are supposed to be taking a picture of (and since pictures will likely be taken in public places with many people around), assassins will probably receive pictures of multiple people, some of whom (and possibly all of whom) are not their targets. This will require the assassin to do some investigative work on their own, identifying their target on the basis of multiple location notifications and pictures received.

3. "Killing" your target: Originally, we had intended to develop an integrated minigame to be played by the assassin and target in order to determine the winner, i.e. who gets "killed." However, since realized early on that writing a multiplayer, real-time game requires a level of technical complexity which is unrealistic on such a short development schedule, we opted instead to implement a more straightforward "kill code" system similar to what is being used already by the University of Chicago Humans vs. Zombies group. This will be described in more detail in the following section.

TECHNICAL ARCHITECTURE

Basic Architecture
- Client: Players install our Android app on their device, allowing them to create and join games, receive updates about their targets, take pictures of other people's targets, and register their kills.
- Server: The Server is a Django app, written using Postgres with the PostGIS geographic extension, and RabbitMQ through Celery.
- External Services: Foursquare Venues API used for reverse geocoding; Google Cloud Messaging Used for push notification.

Functionality
- Login/Registration: The user registers with a nickname (not their actual name) and a password.
- Creating and Joining Games: Upon logging in, the user sees a list of games which have yet to start, ordered by proximity to their current location (the ordering is not currently implemented, but could be easily). They can click on one of these games to join it, or can create a new game with a name, start-time, and location automatically set by GPS.
- Countdown and Game Start: Upon creating or joining a game, the user is taken to the "show game" screen, which shows a ticking countdown to the game start time. This screen shows the number of players currently in the game, as well as a feed of events. When new players join, an event signifying their joining is added to the feed and the number of players update in real time (enabled by a GCM message sent from the server).
- Target Assignment: At the start time of the game (as specified by the user who created the game), the server assigns each player a target, forming a directed graph involving all players, with one cycle. It sends out GCM messages to all the players, and the client shows the user their target, as well as putting a notification in the Android notification center.
- Collecting Sensor Data: When the game is joined, the client starts a Service which sends the player’s location to the server every 10 minutes, using Android’s fused location provider to use GPS or Network location based on availability.
- Picture Taking: The server prompts a user to take a picture by pushing a GCM message. That user then uploads the pictures, which causes the server to push a notification to the recipient. They can then open up an activity to view them, but only once.
- Kill mechanic: Players must agree on a physical action for killing at the beginning of the game. To register a kill with the system and begin target reassignment, the killed user presses a button which shows them their kill code, and the assassin enters their code.

CONCLUSION / EXPERIENCE GAINED

We've yet to actually “play” this game with people besides ourselves in testing, and doing so would most likely provide insights about how people actually react to such a game. However, we did learn from the process of developing it. On the server side, the main challenges were implementing the algorithms to assign targets and detect situations in which users should be prompted to take photos. On the client side, managing session and game state across multiple activities was a challenge (since this was Pete's first non-trivial android app). Overall, hooked up all the moving parts (PostGis, RabbitMQ, the server's REST API, and the client) took up the most time.

Acknowledgements, License, Source

This project was completed as part of CS 234 Mobile Computing (Winter 2013), taught by Professor Chien, with TA support by Tanakorn L. and Falcon Dai
License: Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
Source code at https://github.com/vilterp/hitman-android
https://github.com/whitehaven/hitman-android-server