Hunt the Trojans

by Jonathan Ong, Ran Gross, and Quanjie Geng

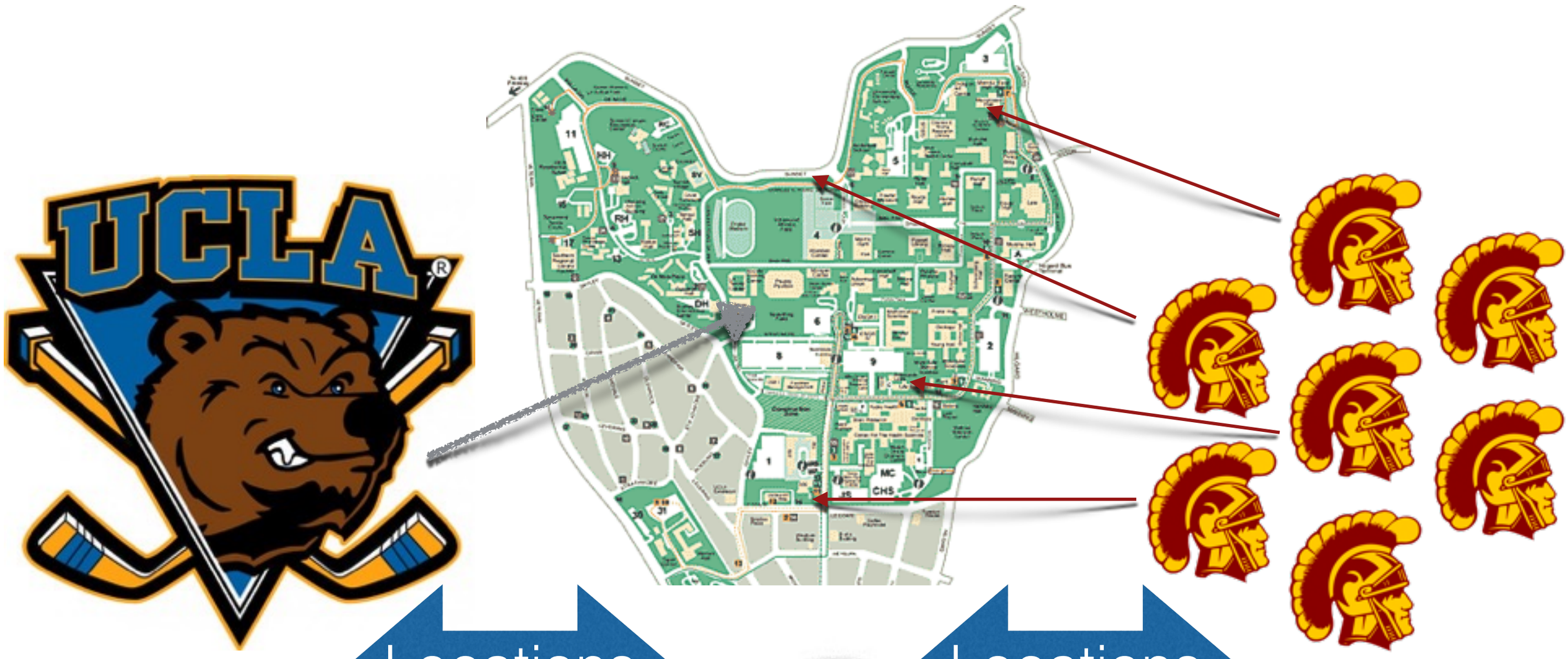
Summary

- An interactive cat and mouse Android app
- Equipments
  - Server (Parse)
  - Android devices
- Communication Methods
  - Wi-Fi, Cellular Data, Bluetooth

Game Logic

- A player can either create a new game or join an existing game.
- The player created the game is the cat and the players who joined the game are the mice
- Cat and mice can request for the locations of everyone that is near them within a certain range
- Locations are served once per 60 seconds for minute ping which both cat and mice have.
- cat has a super ping that allows a request for locations once every 30 seconds.
- A mouse is captured by the cat if Bluetooth connection between their devices is established.
- If the cat captures all the mice within the time limit, the cat wins. Otherwise the mice win.

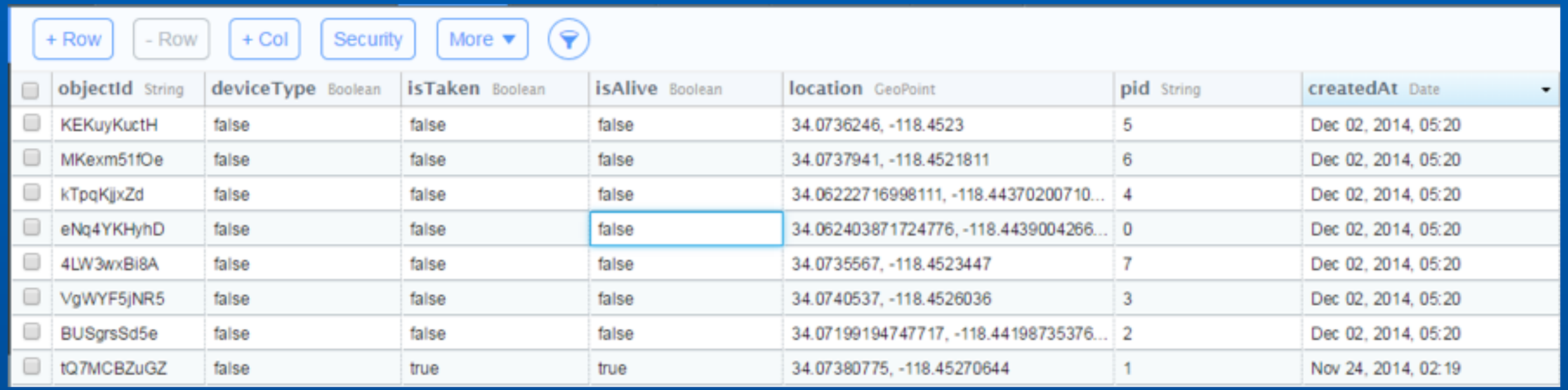
Tracking Locations



Bluetooth Capture

Bluetooth

Parse



+ Row | - Row | + Col | Security | More

objectId String	deviceType Boolean	isTaken Boolean	isAlive Boolean	location GeoPoint	pid String	createdAt Date
KEKuyKuctH	false	false	false	34.0736246, -118.4523	5	Dec 02, 2014, 05:20
MKexm51fOe	false	false	false	34.0737941, -118.4521811	6	Dec 02, 2014, 05:20
kTpqKjjxZd	false	false	false	34.06222716998111, -118.44370200710...	4	Dec 02, 2014, 05:20
eNq4YKHyhD	false	false	false	34.062403871724776, -118.4439004266...	0	Dec 02, 2014, 05:20
4LW3wxBi8A	false	false	false	34.0735567, -118.4523447	7	Dec 02, 2014, 05:20
VgWYF5jNR5	false	false	false	34.0740537, -118.4526036	3	Dec 02, 2014, 05:20
BUSgrsSd5e	false	false	false	34.07199194747717, -118.44198735376...	2	Dec 02, 2014, 05:20
tQ7MCBZuGZ	false	true	true	34.07380775, -118.45270644	1	Nov 24, 2014, 02:19

- The server chosen to store data for the game.
- Data stored includes:
    - ID of each device
    - whether a device ID is taken or not
    - whether the player is dead or alive
    - The locations of each device
- Uses a set of queries to get data that was previously stored on the server to use for the map.

Google Map

- A high level API that let us create a map
  - Move camera across the map
  - Zoom in or out
  - Draw markers
  - Display the device location

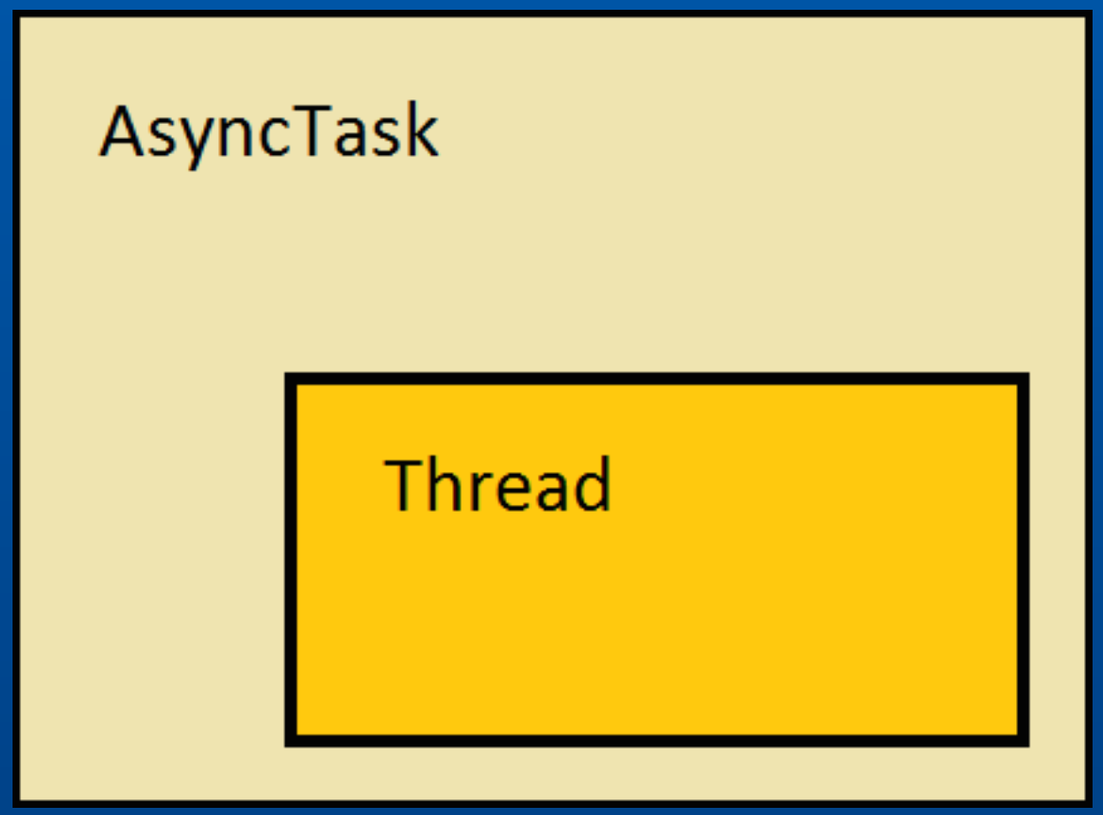
Bluetooth

- Bluetooth is used for a cat to capture a mouse. The cat is the master of the piconet, and all mice are slaves of the piconet.

- This Bluetooth capture is achieved through **AsyncTask**.

AsyncTask

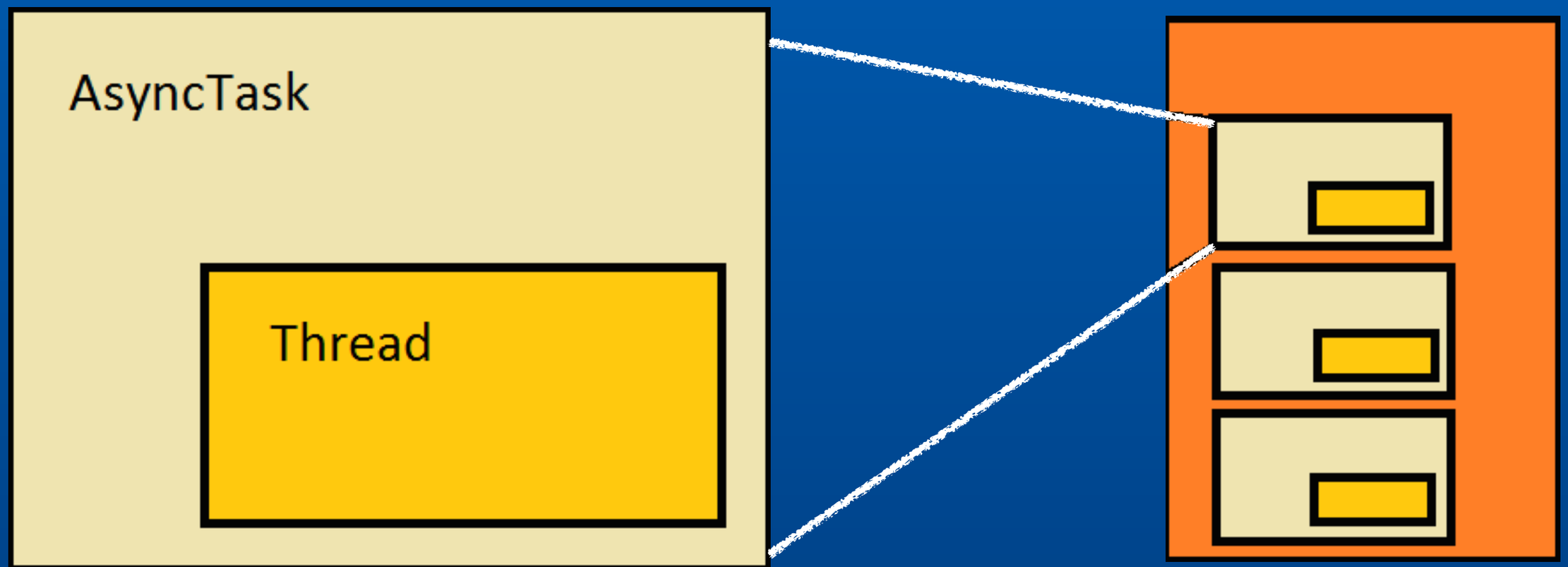
- AsyncTask will run in the background. It will constantly execute the thread's run() function.
  - There are 3 asynchronous tasks in our code.
  - One for the Cat to make a connection for any connecting mouse.
  - Two for the mice: one to search for new devices and one to search through already paired devices



AsyncTask(cont.)

Bluetooth running in the background

Main thread



Ran’s Contribution

- wrote the code that lets the app access the parse server.
- organized the server inputs and defined the entries for the table
- worked on the code for part of the game logic, different activity displays, layout of the app, and integrating the server code into the rest of the app.
- For the map:
    - upload and download of location data that is currently allowed to be viewed
- For the game logic:
    - upload and check conditions on the server to see if we can:
        - make a game
        - join a game
        - won/lost a game

Nero's Contribution

- Used the Google Map API to create a map in the app and support the following functionalities
  - Acquire the device's current location through Wi-Fi or GPS
  - Update current location through a location listener
  - Draw a marker on the map with a location as input
- Integrated Parse API for upload and download (from Ran) to support the following functionalities
  - Put the device's current location on Parse as current location updates
  - Download the locations of other devices from Parse and draw markers for those locations upon player's request

Jonathan’s Contributions

- Bluetooth
  - Hours of research through online documentation and experimentation
  - Creating the **BluethoothRunner** and **AccetThread** classes wich allowed the Cat to run its Bluetooth without blocking the main thread.
  - Creating the **ClientRunner** and **ConnectThread** classes, which allowed the Mouse to run its Bluetooth tasks without blocking the main thread.
  - Creating the **BackRunner** and **BackThread**, which allowed any device to check its “already paired devices” for a reconnection.
  - Using the classes, programmed the socket communication between AcceptThread and ConnectThread to exchange a mouse’s ID number to the cat for further processing.
  - Debugging our Bluetooth implementation

Jonathan’s Contributions

- Parse
  - Deciding on using Parse after considering and researching many other choices
  - Supporting Ran on learning how to use Parse to update server
  - Using Parse queries and Parse puts for game logic and Bluetooth functionality, and game logic
- Google Map
  - Supporting Nero on Google Map API functions and with debugging
- Layout
  - Create the GameStart activity. Created and mapped each button to functions.
  - Creating the HunttheTrojan activity. Configured each object so any events won't alter or distort the UI. Created and mapped each TextView and Button to the logic.

Jonathan’s Contributions

- Game Logic
    - Creating “Join Game” function in the “GameStart” activity. The function makes a ParseQuery and checks the cat’s “isTaken” and “isAlive” fields. Programmed Toasts to prompt the user on the state of the game.
    - Creating “Make a game” function in the “GameStart” activity. The function makes a ParseQuery to check the cat’s “isTaken” field. Programmed Toasts to prompt the user depending on the response from Parse. Programmed on success to display new buttons “Drop the Cat” and “Start the Game.”
    - Creating “Drop the cat” function in the “GameStart” activity. The function makes a Parse submission to free the cat and resets the activity to default state. This button and the “Start the game” button are hidden.

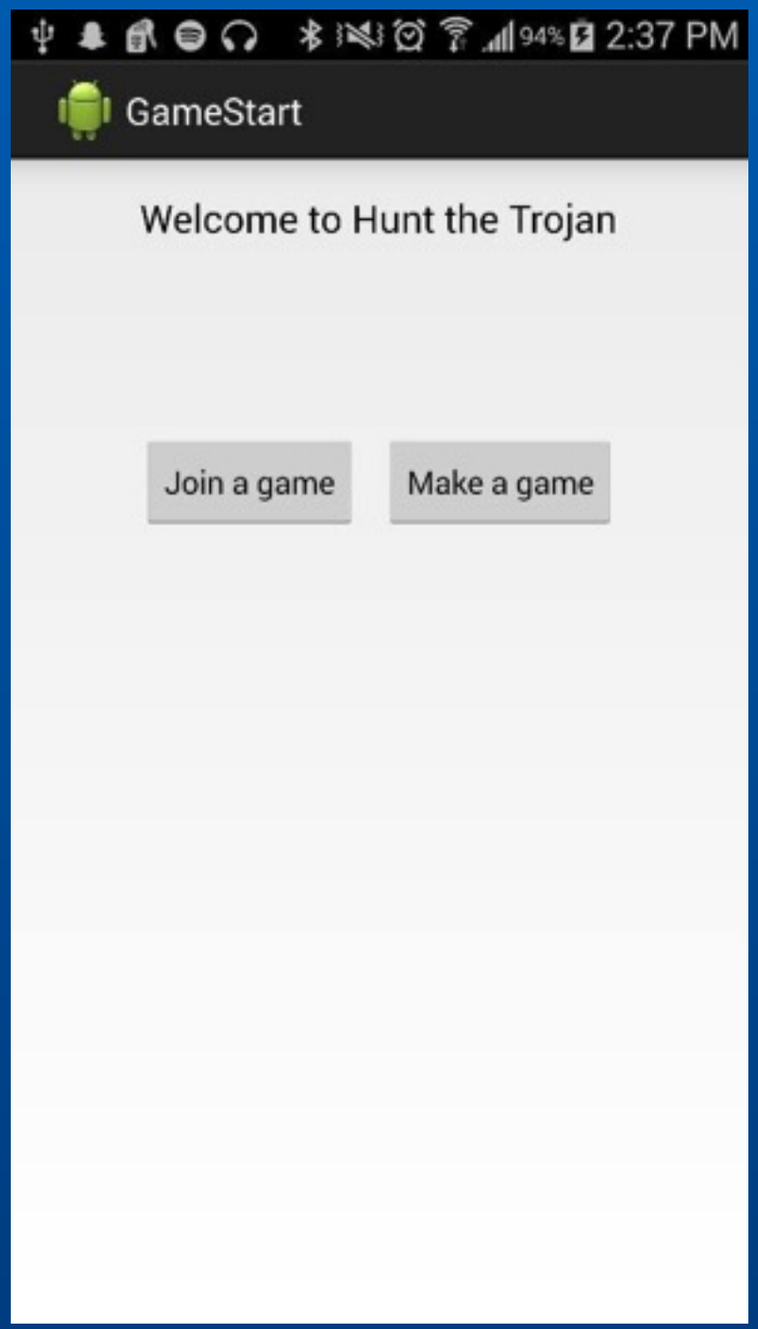
Jonathan’s Contributions

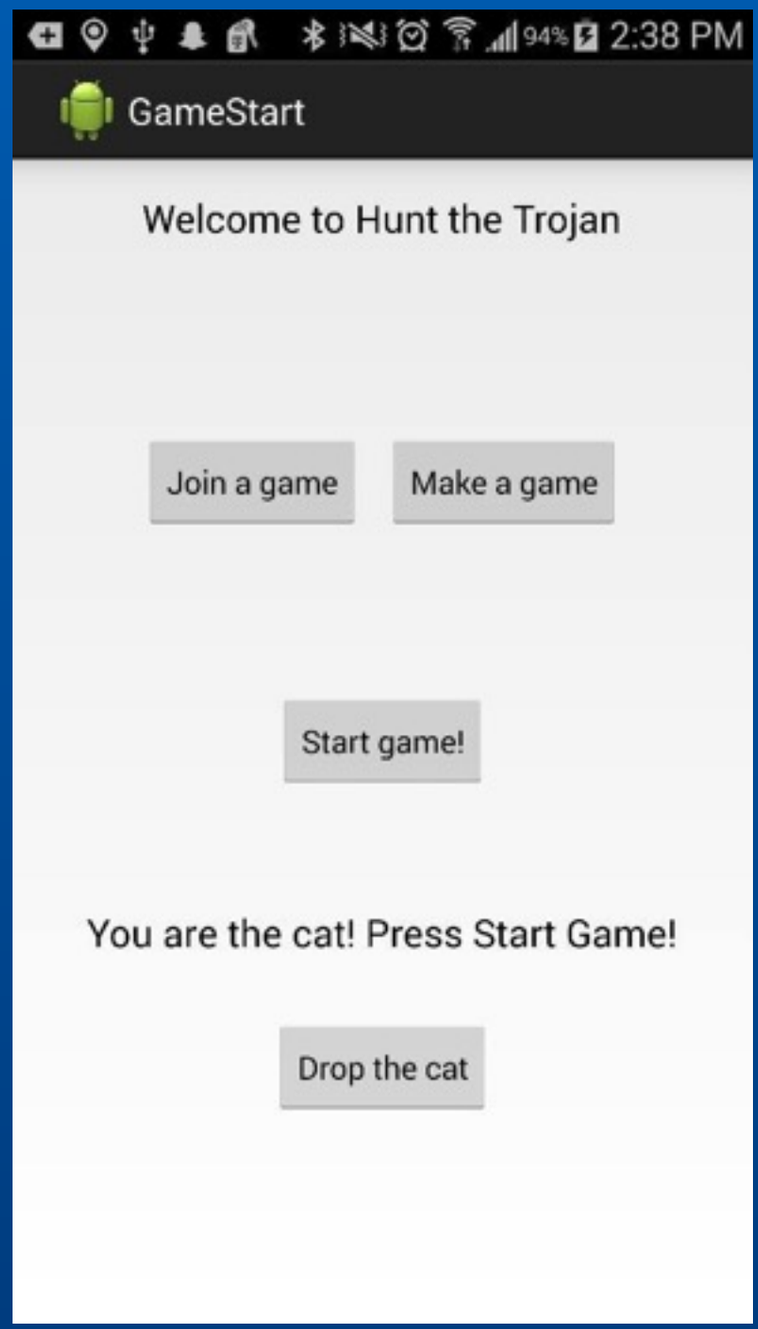
- Game Logic
    - Creating “Start the Game” function in the “GameStart” activity. The function makes a Parse submission and sets the cat’s “isAlive” to true so that mice cannot join the game once the game has started. Success will start the “HunttheTrojan” activity.
    - Creating the functionality for MouseConnect in the “HunttheTrojan” activity. After a connection to the cat is made, a text displays that the Mouse has lost and prompts for any action on the screen. Any action will leaves the “HunttheTrojan” activity and returns to the “GameStart” activity.
    - Creating the function CatConnect in the “HunttheTrojan” activity. Programmed so that after a mouse connects to the cat, the cat sets the ”isAlive” for the corresponding mouse to “false.”
    - Creating the Timer clock in the “HunttheTrojan” activity. Programmed this clock to update a TextView and triggers a gameEnd when the Timer has timed out.

Jonathan's Contributions

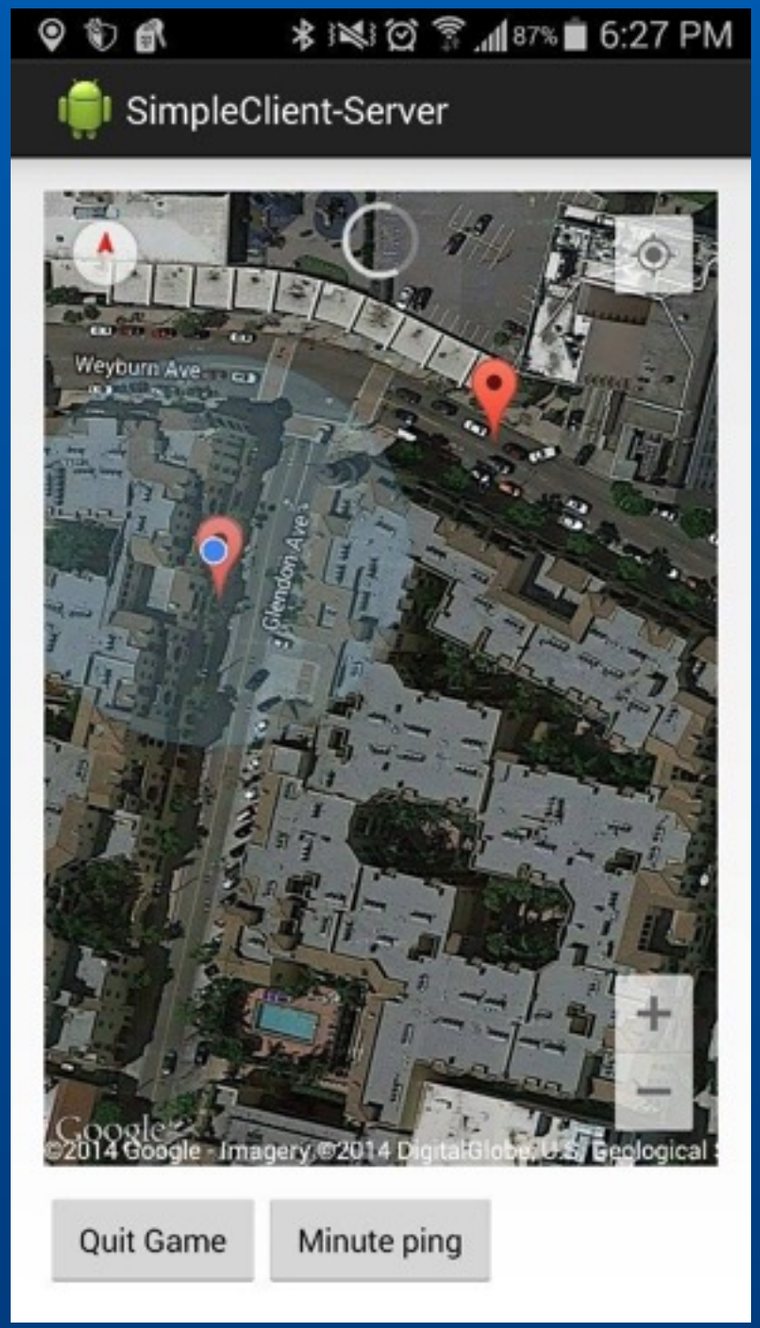
- Game Logic
  - Creating the MinutePing function in the "HunttheTrojan" activity. Programmed a Timer to act as a cool down. The button cannot be clicked until the timer has finished.
  - Creating SuperPing function in the "HunttheTrojan" activity. Disabled for mice; has same "cooldown" functionality as MinutePing, but runs at a much shorter duration.

Screenshots - Start





Screenshots - Map



Future Plans

- Lower Bluetooth range or measure distance with Bluetooth so the capture distance is not so long (> 30 m).
- When data signal is lost, instead of crashing, just give a warning.
- Make the app run more smoothly.
  - Give the mouse a display while waiting for game to start after they join.
  - Add some effects to hide lag caused by accessing server.
- Improve the overall UI

Q&A

Thank you!