Radiological Simulator Mobile Application

Sponsor: Lawrence Livermore National Laboratory
Mentor: Greg White Instructor: Sepideh Mazrouee
Michael McBeen, Casey O’Leary, Sidney Dimmel, Daniel Skrei

Background

Lawrence Livermore National Laboratory is a federal research facility primarily funded by the U.S. Department of Energy and has research programs in radioactive materials. This project aims to provide enhanced training techniques by taking advantage of mobile platforms and devices to simulate radioactive source scenarios. Lawrence Livermore National Laboratory has developed their own Windows application for this project, and is now defined for Computer Science to develop applications for Android and Mac mobile devices.

Project Definition

Implement the application on the iOS platform for use with iPads
Core Application Functionalities

- Real-time GPS location display
- Multi-Peer Networking via local network (wireless)
- Dose and Spectra physics calculations
- Replace Windows UI
- XML parsing for scenario data
- Real-time graphs containing radiation data

Hardware & Environment

System: Macbook Pro
IDE: Xcode 7
Programming Language: Swift & Objective C
iOS Device: iPad 2 and iPad Air
External Libraries: Core Plots, CFNetwork, Cocoa Pods
All application testing was performed on campus. Debugging the iOS application was done in conjunction with the Windows version.

Design Summary

1. Graphs, GPS, map image, local scenarios, and networking thread are initialized
2. XML file containing scenario data is loaded into the application and parsed
3. Real-time calculations and Networking
   i. Radiation channel and Dose rate begin updating in real-time relative to the source and your current location, which is then represented via graphs
   ii. In a separate thread, networking is constantly searching for peers, establishing peer connections, and relaying all GPS/radiation data to connected peers

Implementation

- Networking: Implemented using CFNetwork library, bridged from Objective C, to handle underlying socket programming and data transfer
- GPS: Using the Core Location library provide in the Swift package, a GPS module was implemented providing location data for calculations and map display
- Map: MKMapKit provided the framework to manipulate maps in the user interface
- Graphing: Both graphs were implemented using the Core Plots library, which was bridged from Objective C to Swift in the Xcode project

Graphical User Interface

Technical Challenges

- Compatibility of Xcode and Objective C bridged libraries in the context of calling Objective C functions from Swift
- Challenges regarding actual data values during Debugging
- Networking had to be reversed engineered due to lack of documentation and debug ability of Lidgren
- Calculations and data handling needed to be adapted as data types were handled different in Swift, especially when converting
- Compatibility of iOS and Xcode when installing the application from Xcode to a USB connected iOS device
- Physics calculations resulting with in-accurate data
- Restructuring application architecture without events, which were used extensively in the Windows version

Future Work

- Complete iOS networking: general framework established
  - Obtain more Packet structure implementation from Lidgren
  - Use same encryption standard from Lidgren, currently unknown
  - Extend message handler to include more Lidgren type messages
  - Possibly consider moving towards multi-platform networking library for all versions of application
- Implement Command/Sensor Node functionality
  - Command mode allows the node the see the location of the other nodes on the map
  - Sensor mode allows the node only to see the source ellipse data on the map, along with the node’s position

Acknowledgements

- Lawrence Livermore National Laboratory for the opportunity to work on this project
- Mentor Greg White for his time, technical direction, and support
- Instructor Sepideh Mazrouee for guiding and accompanying the team throughout the duration of the project

Team Radiological Simulator