SENIOR DESIGN REPORT

Mobile Navigation Control for Planetary Web Portals - Trek Controller -

5/19/2017

Team members:
John Calilung
Miguel Martinez
Frank Navarrete
Kevin Parton
Max Ru
Catherine Suh

Faculty Advisor:
Dr. Elaine Kang

JPL Liaisons:
Emily Law
George Chang
Shan Malhotra

Table of Contents

I. Executive Summary

This is the final report for the Senior Design group assigned to JPL’s Mobile Navigation Control for Planetary Web Portals. The purpose of the year-long Senior Design class at Cal State LA is to facilitate an environment similar to the work field - to have students work together and with professional engineers on a large-scale project. The project is to develop a mobile controller for JPL’s existing Planetary Web Portals, also known as the Trek Portals. The following sections explain the project further, specify the software requirements needed for and describe each component of the project, and describe the application flow. Also included is information on performance, the technologies used, and extra features added to the project.

II. Overview

The purpose of this project is to develop a mobile remote controller for JPL's Mars Trek and Vesta Trek. As stated on the official Mars Trek website, 'Mars Trek is a NASA web-based portal for exploration of Mars.' It provides an exciting way for students, citizen scientists, and the public to explore some of Mars' most spectacular landforms as well as NASA areas of operation on Mars' surface. Due to privacy constraints, the mobile controller, titled Trek Controller, will only allow exploration of a standalone site which mimics Mars Trek.

Trek Controller is implemented as both an iOS native app and a Web app so that users of all devices can explore Mars. There are three controller modes which use joystick controls, touch gestures, or motions that utilize the mobile device's gyroscope. Trek Controller also offers guest queueing for situations such as conferences or museum events where multiple users wish to use the controller. A user must enter a 4-digit code or scan a QR-code to enter the queue or, if there is no queue, to enter the default controller mode which uses touch gestures with the phone oriented upright (portrait).

Once in a controller mode, the user can easily switch between the three modes using the options on the menu bar at the bottom of the screen. To use the joysticks or gyroscope, the mobile device must be turned sideways (landscape). In all controller modes, the enabled movements are up, down, left, right, the diagonals between, zoom in and zoom out. There is also a toggle button to toggle the view of Mars between 2D and 3D as well as a reset button which will reset the view of Mars to its default center on the page.

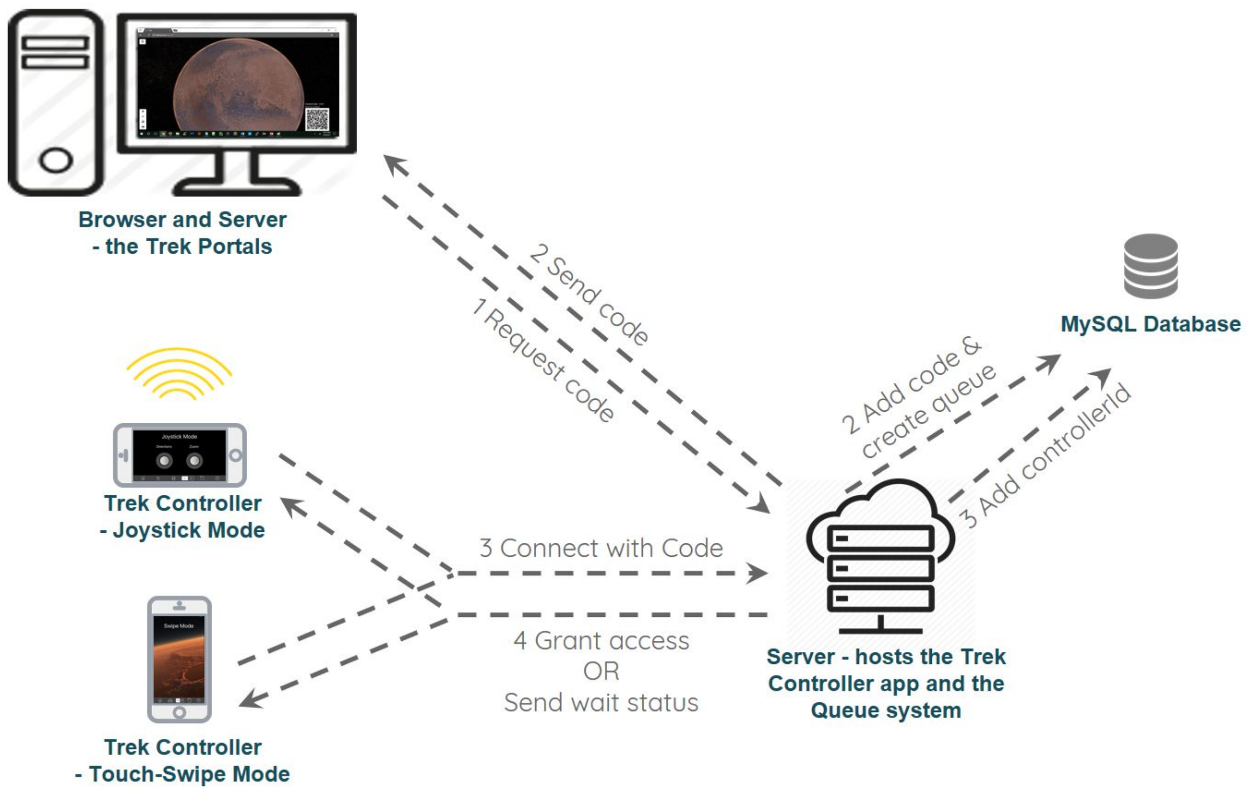
Trek Controller also includes other features like Mars Fast Facts, Weight on Mars, and social media links. Mars Fast Facts includes information such as the temperature on Mars and number of days to a Mars year. Weight on Mars will prompt a user for a weight and convert it to what the weight would be on Mars. The social media links will take a user to JPL accounts of those social media avenues. Additional features we had hoped to implement are Presenter Mode which would disable the guest queue and allow handing over control to particular users, Bookmarks of particularly interesting coordinates on Mars, and Screenshots saved to the mobile device of whatever view is currently on the screen.

III. Environment Setup

1) iOS Application
    a) Xcode 8.1 and later
        i) Xcode 8.3 and later Recommended
    b) iDevice ≥ 5 and iOS ≥ 8.0
2) Web Application
    a) Java (SE 6 or later)
    b) Eclipse IDE for Java EE Developers
    c) Tomcat 7
        i) External libraries needed:
            1) JSTL 1.2
            2) MySQL Connector JDBC Driver 5.1
3) Queueing Server
    a) MySQL Version 5.7
        i) Root
            1) User: root
            2) Pass: Cholula2020$picy
        ii) Databases
            1) User: tc_user
            2) Pass: Tapatio2020$picy

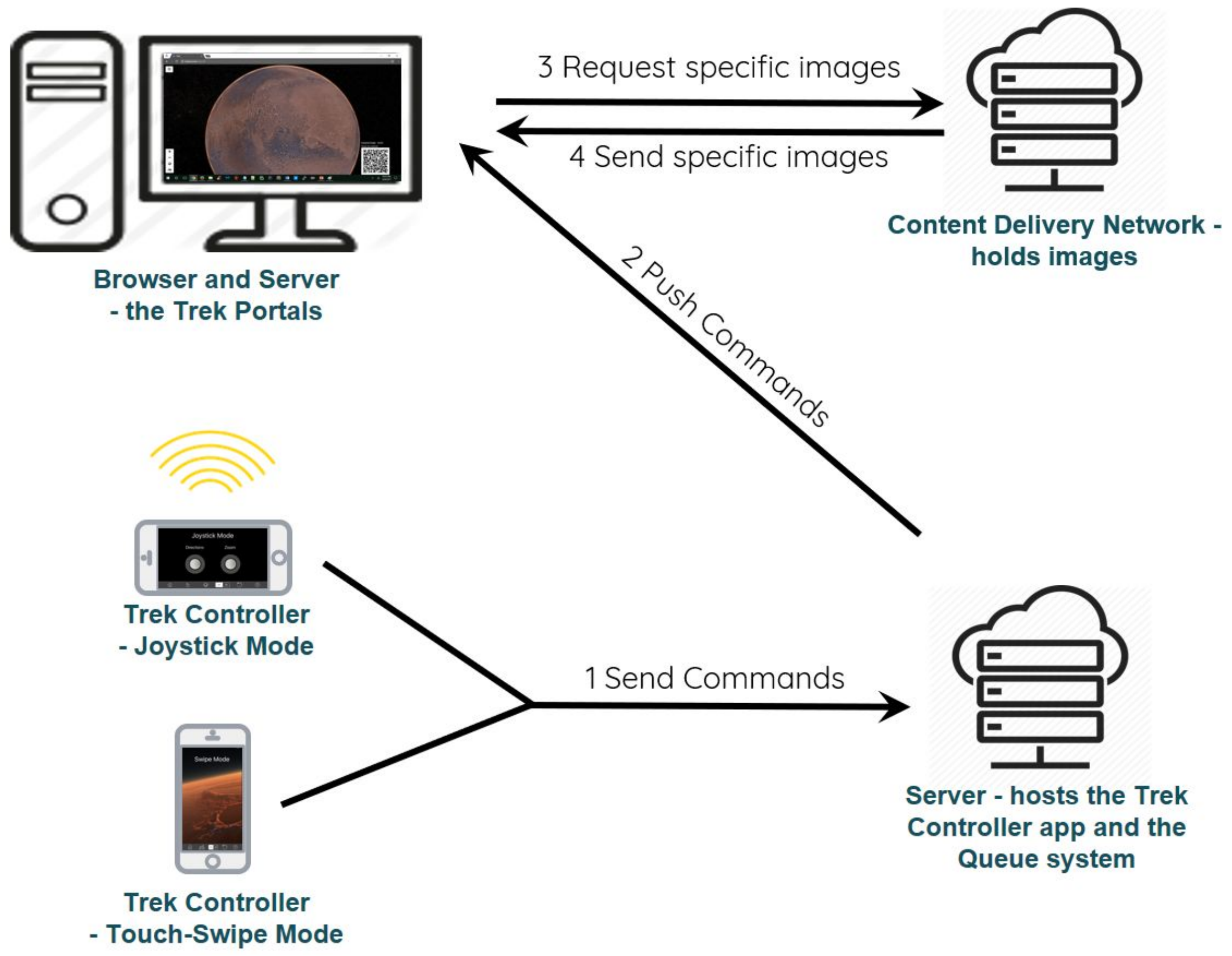
IV. Application Flow

Initialization


Browser and Server - the Trek Portals
2 Send code
1 Request code
MySQL Database
2 Add code & create queue
3 Add controllerId
Trek Controller - Joystick Mode
3 Connect with Code
4 Grant access OR Send wait status
Server - hosts the Trek Controller app and the Queue system
Trek Controller - Touch-Swipe Mode

1. Trek Portal goes up and requests unique ID from Queueing Server
2. Queueing server generates unique ID by checking available codes in database then returns code. Trek Portal displays unique QR code.
3. Trek Controller scans QR code, connects with server, server creates a new ID and returns to controller.
4. Server updates Trek Controller with position in queue or sends command to start controlling trek.

In Control


3 Request specific images
4 Send specific images
Content Delivery Network - holds images
Browser and Server - the Trek Portals
2 Push Commands
Joystick Mode
Trek Controller - Joystick Mode
1 Send Commands
Swipe Mode
Server - hosts the Trek Controller app and the Queue system
Trek Controller - Touch-Swipe Mode

1. Trek Controller sends commands to server.
2. Server looks up which controller is to receive the command and pushes the command accordingly.
3. Trek Portal receives command and goes to CDN for additional planetary images.
4. CDN returns images and Trek Controller reacts to command from Trek Controller.

V. Implementation

1. Agile development principles were used to allow for frequent testing periods, which helped the developers refine the design, features, and functionality.

2. Multi-Platform support was taken care of my making an iOS app for Apple devices and a Web App for all mobile devices.

3. A Queuing system was made to handle multiple users with position feedback.

4. Asynchronous Javascript and XML (AJAX) is a client side script to send the user's move commands from the Web App Trek Controller to the Command Server.

5. HTTP requests is a native HTTP communication protocol used to send commands from the iOS devices to the Command Server.

6. iOS Mobile Controller - Controller Modes
    a. **Touch-Swipe Mode:** There are two primary functions used from the UIKit - touchesBegan and touchesEnded. TouchesBegan is called when the user first touches the screen. Once the user removes his or her finger from the screen, touchesEnded is called. Within our touchesEnded function, the angle of the swipe is calculated then sent to the state class. The state class then returns if the user's swipe was up, up-right, right, down-right, down, down-left, left, or up-left.
    b. **Joystick Mode:** The two joysticks from joystick mode uses an open source library called "AnalogJoystick.swift" to create the joysticks. We made a scene for the two joysticks then added the scene to a skView. A link to the library can be found in *Section VIII: Technologies*.
    c. **Motion Mode:** The CMDeviceMotion Class from the Core Motion framework in iOS encapsulated measurements of attitude, pitch, and yaw by using the device's gyroscope. Using the measurements, appropriate movement commands could be sent.

7. Web Mobile Controller — Controller Modes:
    a. **Touch-Swipe Mode**: Used an external open-source library called “hammer.js”. For touch-swipe mode, we created a ‘div’ element that fit the whole screen minus the menu bar. We then attached “touch recognizers” to the element so that it can recognize touch gestures. A link to the library can be found in *Section VIII: Technologies*.
    b. **Joystick Mode**: Used an external open-source library called “nipple.js”. We created two ‘div’ elements and placed them in the middle of the screen. We then created two joysticks and attached each of them to their corresponding ‘div’ element. A link to the library can be found in *Section VIII: Technologies*.
    c. **Motion Mode**: This has not yet been implemented for the Web Application. Although, we will most likely use the DeviceMotionEvent API for implementation.

8. Standalone Site Integration (Mars Trek)
    a. Created a namespace (trek-ctrl-namespace.js) for our variables throughout the source code that we were given.
    b. Created a communication point (trek-controller.js) from our server to the standalone site.
    c. Modified the following files from the source code:
        i. **3DGlobeView.js (3D Control)**: There are keypress events in this file that listens for user movement on the Trek site. We placed our variables alongside these events to simulate a movement command. We then would turn on or off these variables according to user commands.
        ii. **SearchMap.js (2D Control)**: We found a Maps object in this file and we looked at the ArcGIS documentation to manipulate the extent or viewport of the current scene. We used built-in methods to move the map with any of the 8-way direction movements and we could also zoom in or out.
        iii. **ControlBar.js (QR Code)**: Used this file to overlay the QR code that is generated from qrcode.min.js. We also used this to put up the 4 digit code that is generated from our server.

9. Queueing Server API
   Trek Controller uses the following API to connect devices to the Queueing server and send commands.

   Specifications
   Protocol: HTTP
   Request Type: GET
   Request URL:
   http://csproject.calstatela.edu:4050/WebMobileController/TakeCommands

   Sample Request URL:
   http://csproject.calstatela.edu:4050/Hyperwall?x=1&y=0&z=0&w=1

Variable Table:

Variable	Req/Opt	Type	Description	Example
x	optional	int	Left, stop, or right {-1,0,1}, respectively	1 0
y	optional	int	down, stop, or up {-1,0,1} respectively	1 0
z	optional	int	Zoom out, stop, or in{-1,0,1} respectively	1 0

Example 1:
Scenario: swipe from center to the top left of the screen:
x=1
y=1

Full URL to send:
http://csproject.calstatela.edu:4050/Hyperwall?x=1&y=-1

Example 2:
Scenario: swipe from center to the the right
x=-1
y=0

Full URL to send:
http://csproject.calstatela.edu:4050/Hyperwall?x=-1&y=0

Example 3:
Scenario: Pinch out (spreading) to zoom in

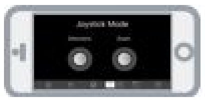
z=1

Full URL to send:
http://csproject.calstatela.edu:4050/Hyperwall?z=1

10. QR Code
    a. **Library**: QRCode.js generates QR Code image from the browser after receiving ID from Queueing Server
    b. **Trek Portal**: Using jQuery, existing on-page elements are modified to fit a QR Code image so that a Trek Controller can scan it.
    c. **Trek Controller**: using any phone QR Code reader or the iOS Trek Controller App: scan the QR code on the Trek Portal screen to be placed in the queue to control trek.

VI. Major Components

Trek Controller



Trek Controller is implemented as both an iOS native app and a Web app so that users of all devices can explore Mars. There are three controller modes which use joystick controls, touch gestures, or motions that utilize the mobile device's gyroscope.

Trek Portal

A NASA web-based portal for exploration of Mars. Trek Controller integrates with Trek via client-side JavaScript functions. Trek Portal uses its own server for the hosting of HTML files.

Trek Portal CDN

Trek Portal gets its images of Mars via a Content Delivery Network (CDN) on Amazon Web Services. The fetching of images in handled automatically with the ArcGIS JS library.

Trek Controller Web App Server and Queueing Server

Trek Controller Web App Server and Queueing Server hold all the files necessary for the web app to function as well as the queueing system to handle all Trek Controllers and Trek Portals.

Queueing Server Database

Stores states of Trek Controllers and Trek Portals. Is used to generate unique IDs when a new Trek Controller or Trek Portal goes live.

VII. Extras

1. Mars Fast Facts: the Mars Fast Facts section of the application is meant the give the user a few fast, fun facts about Mars. For instance, the largest volcano in the solar system is on Mars and is larger than the state of Arizona, and the length of a day on Mars is 24 hours and 37 Earth minutes.

2. Weight on Mars: input an Earth weight in pounds and see its equivalent weight in pounds on Mars.

3. About Trek Controller: a credits page to open source libraries used to create Trek Controller.

4. Social Media Links: social media links to the pages of JPL on Facebook, Twitter, Youtube, and Instagram.

VIII. Performance

Latency: average time to send a user’s move command from Trek Controller to the server is on average less than 20 milliseconds over WiFi.

Disconnects: Trek instance disconnects have automatic reconnect attempts handled with SSE’s. Lost commands are not resent by the server; SSE settings allow for the most recent lost data to be sent again, but we decided against utilizing this as it would cause camera movements not directly tied in real-time to user commands. Furthermore, controller disconnects lead to lost commands since connections are over HTTP Get Requests using AJAX.

IX. Technologies

1. JavaScript was used in the development of the Web app; the Standalone site that was provided was developed in Javascript as well.

2. Third Party Software was used for client side purposes. Qr.js is a library used to generate the QR code on the standalone site. Hammer.js was a library used for the touch-swipe controller mode, and Nipple.js was a library used to generate the joysticks on the Web app. AnalogJoystick.swift was used to generate the joysticks on the iOS app. The icons on the menus were all sourced from Icons8. Here are the respective links:
   http://hammerjs.github.io/
   https://github.com/yoannmoinet/nipplejs
   https://github.com/MitrophD/Swift-SpriteKit-Analog-Stick
   https://icons8.com/

3. Server Sent Events (SSE) was used in order to have a uni-directional communication protocol; it uses HTTP to push the user's move commands from the Command Server to Trek Portal in real time.

4. Servlets are Java programs that extend the capabilities of a server and were used in the implementation of Server Sent Events (SSE) communication protocol.

5. Swift 3 is Apple's programming language used to develop iOS apps. It is the current standard and Apple requires apps to be follow it's guidelines if it is to be published to the App Store.

6. MySQL is a relational database management system and was used in the queue implementation and stored the Controller ID and Trek ID.

X. Conclusion

We’ve designed the entire communication architecture of Trek Controller and integrated it into the prototype of JPL’s Trek Portal. For the back-end, we created our own control variables to read and issue move commands by piggybacking on the existing camera movement API to control what is seen on Trek Portal. The front-end work resulted in a user-friendly mobile controller that supports 8-way movement and zoom. The Queueing system was designed using a MySQL database to keep track of users and Trek Portal instances. Additional features we would have liked to implement are the ability to bookmark locations and save screenshots to the mobile device of what is seen on Trek Portal.