Senior Design Final Report

Augmented Reality for Hydrology Version 2

Version 1.0 - 05/04/2019

Team Members:
Mher Oganesyan
Refugio Arroyo-Martinez
Leonardo Obay
Anthony Soto
Gilberto Placidon

Faculty Advisor:
Dr. Elaine Kang

Liaisons:
George Chang
Natalie Gallegos
Emily Law
Shan Malhotra
Michael Rueckert

Table of Contents

1. Introduction 2
    1.1. Background 2
    1.2. Design Principles 2
    1.3. Design Benefits 3
    1.4. Challenges 3
    1.5. Achievements 4
    1.6. Testing Results 4
2. Related Work and Technologies 5
    2.1. Existing Solutions 5
    2.2. Reused Products 5
3. System Architecture 6
    3.1. Overview 6
    3.2. Data Flow 7
    3.3. Implementation 8
4. Results and Conclusions 11
    4.1. Results 11
    4.2. Future Work 11
5. References 12

1. Introduction:

1.1. Background:

Jet Propulsion Laboratory (JPL) is well known for their satellites and the mars rover. They develop numerous technologies for space exploration, but that is not all that they do. JPL also collects and stores hydrology data into the Watertrek database from the Western United States, using a combination of sensors from in environment to in space and by mathematical models that they have developed. One example of this is using sensors placed in the mountains to weigh snow pillows and give an approximation of the snow depth. Provided the Augmented Reality (AR) framework for Android from last year's Senior Design team, version 2 of the Augmented Reality for Hydrology application is aimed at improving the representation of the hydrology data by implementing a more robust and visual user interface.

Augmented Reality is the use of computer-generated graphics to alter the user's perception of the real world. This is done by taking a device that has a camera and a display that can render computer graphics. As the user looks at the live camera feed, computer-generated objects can be place on the display; giving the user the feeling as if those objects were in the real world.

JPL currently has a web application that displays Watertrek data on an interactive map of the world. The limitations to this are that the user can only view the graphics that are drawn onto the map and not the data that is correlated with each point-of-interest.

Motivation to make the Android application are that it will allow for an interactive way for the user to learn about hydrology in their local area with the use of augmented reality from version one's framework. As the application will be on Android devices, it will also allow the user to have access to the data on the go. Since the data is large and in a raw format, we would be providing the historical data in a graph format for the user to be able to see trends. In addition, the AR technology will help visualize data sets that are not visible to the user since they are located underground.

1.2. Design Principles:

The main deliverable is to create an Android application the represents hydrology data from Watertrek by using the augmented reality components from version one's framework. Our goal is to make the user interface robust and intuitive

to provide a smooth user experience. As the user experience is at the top of the priority list, along with the representation of the scientific data, we focused on that. For the user experience, we wanted fast loading times so the user did not have to wait for data population and smooth device performance. Fast loading times are achieved by using techniques that split up the data set, along with sorting algorithms that require small computation time. Some data may be large; therefore, they can be preprocessed on application start up. In most cases, the data can be cleaned up to reduce complexity, which will increase performance. In addition, the user interface must be simple enough for the user to easily navigate it without having any knowledge about the application or the hydrology data. Lastly, the application must have modular components for future developers to expand on the application, while implementing a single responsibility principle for the applications activities.

1.3. Design Benefits:

By creating an application where the user experience is the top priority for representing the hydrological data, it helps ensure for the smoothest performance of the application. Load times to data is near instantaneous and device performance is smooth even when rendering terrain data. Creating modular components allowed us to easily update the application several times without spending too much time reprogramming. This will also be useful for future developers for the application as they can easily improve and expand on all the different components provided.

In addition to focusing on the user experience, the user interface is much more intuitive and appealing than before. The user can easily navigate throughout the application without having the need to figure out where they left off or how to get to the specific view that they require.

1.4. Challenges:

The main challenges that we ran into during development of the project were:

- Data processing and device performance
- 3D terrain mesh scaling accuracies
- Discrepancies from sensors of different devices

1.5. Achievements:

We made an Android application that visualizes hydrology data using Augmented Reality components built from version one's framework. Improvements that came with version two of the application are as follows:

- Improving the user interface and user experience
- Refining the interactive user interface that visualizes the dataset
- Provide line-of-sight information
- Enhancing the performance of data processing for REST calls
- Providing historical data in a graph format for clearer data analysis
- Ability to filter the historical data by specifying a range in time
- Superimposing a 3D terrain mesh onto the actual terrain through the camera feed
- Use the implementation of maps to help guide the user to a specified point-of-interest

1.6. Testing Results:

The application was tested in various areas throughout Los Angeles and Mount Wilson of the Angeles National Forest. Achievements that were provided above were from testing results. We saw a major performance boost in retrieving large amounts of data from Watertrek's database; a thirty second wait time for data to load has gone down to a near instantaneous loading time. Testing in the mountains allowed us to view the accurate registration of the terrain mesh onto the actual terrain through the camera view.

2. Related Work and Technologies:

2.1. Existing Solutions:

As mentioned previously, the only existing system that visualizes the hydrology data is the Watertrek web application. Even then the historical data that is stored in the Watertrek database that correlates to the different points-of-interest is not displayed to the users of the application. The web application provides visuals on the map that can show river networks, locations of wells, soil moisture, change in the terrain over time, etc. Historical data can only be accessed by selected a specific year on the timeline located at the bottom of the web page, thus changing the visualization on the map.

2.2. Reused Products:

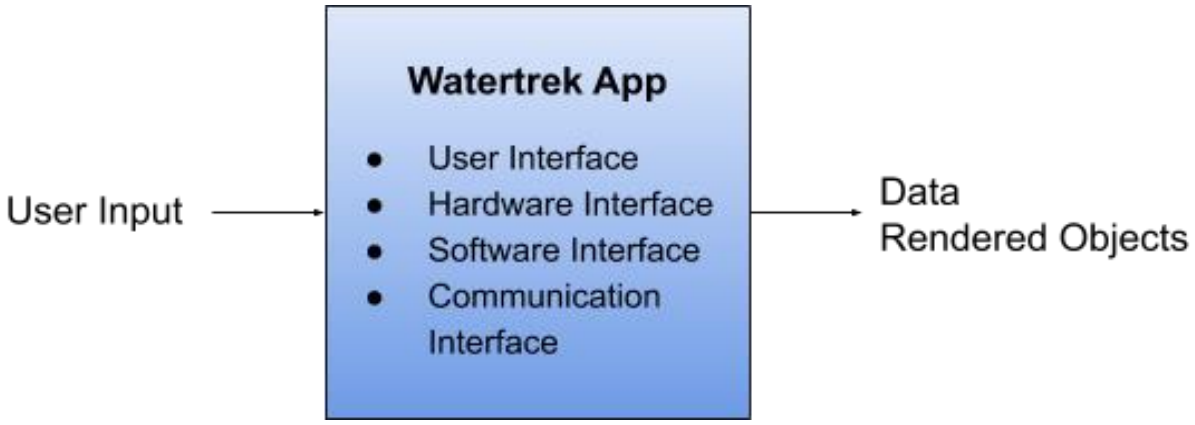
The framework from version one was used to develop the application that was developed on Android Studio by using Java. OpenGL ES 2.0 was used to alter and create new augmented reality components for the framework. Open Street Maps was used to provide an eagle-eye view of the point-of-interest. Graph View was used to create graphs for the different data types to display trends. Arc Menu was used to make an easy-to-access menu for selecting different data types. Swagger was used for managing and testing the REST calls correlated to Watertrek. ArcGIS was used for to retrieve data for the terrain mesh.

3. System architecture:

3.1. Overview:

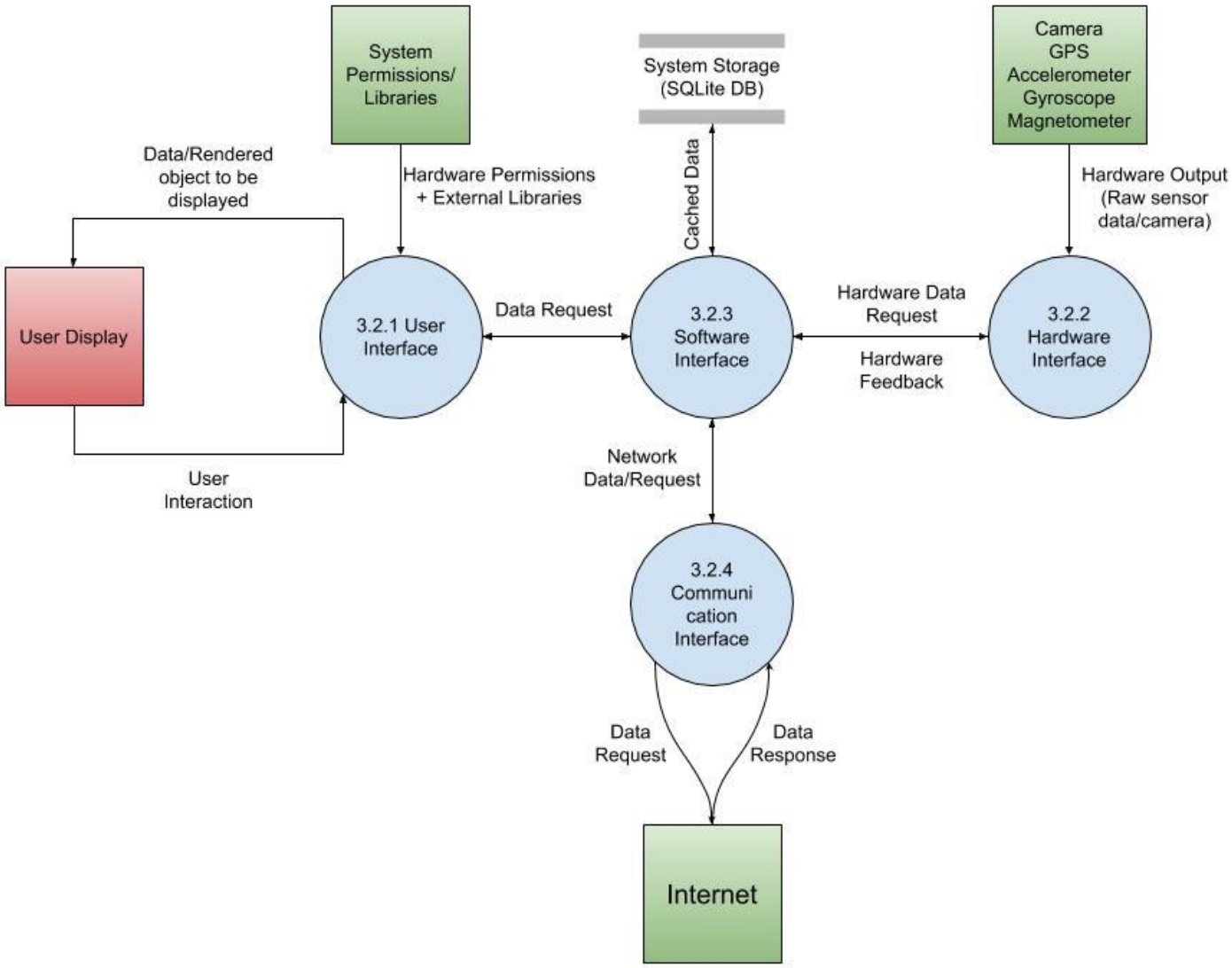
The architecture for the application is self-contained and is split into four major modules. Each module handles the various components needed for the operation of the program. The overall application is based on user input and taking that input, a specified output will be displayed to the device with relevant Watertrek data.

The diagram below (DFD level 0) is a high-level representation of how the application operates:



3.2. Data Flow:

Below is a diagram (DFD level 1) that portrays how the different components interact and communicate with each other to have a single functioning application. This is a brief overview of how they work together:



3.2.1 User Interface (UI): Allows the user of the application to interact with and navigate the application. It takes in user inputs and processes them by passing the necessary hardware readings to the software interface to make network calls and data processing to output the relevant data to the user.

3.2.2 Hardware Interface: Continuously outputs sensor data. That information is grabbed by the user/software interface whenever a user interacts with a UI element that needs sensor data to process network calls.

3.2.3 Software Interface: Handles network calls by either grabbing cached data or requesting a network call from the Communication Interface and caching the data for later use when requested by the user. Also, does data processing from retrieved data then passes data to user interface.

3.2.4 Communication Interface: Takes in network requests from the Software Interface, executes them, and retrieves the output from the internet to pass into the Software Interface for caching of data or instant use by the User Interface.

3.3. Implementation:

Our project was focused on six areas: the database, local data processing/management, mesh processing, textures for the mesh, user interface and user experience, and improvements to the AR framework.

3.3.1. Watertrek Hydrology Data

Hydrology data is gathered using JPL's REST API, provided valid user credentials. Before calls can be made to JPL's servers, we establish a connection with the credentials and use that instance throughout the application. Since there are different data types that we provide, we have base URLs that we append parameters (start date, end date, or even bounding box) before executing the call. The REST calls are also split into multiple URL requests, separating by date.

Digital elevation map (DEM) data is retrieved by making a network request from a base URL, specifying the latitude, longitude, bounding box, resolution, return type (e.g. TIFF), and pixel data type (e.g. signed 16 bit). Retrieval of the DEM is executed on application startup, in the loading screen, due to long loading time and processing.

Textures for the terrain mesh are retrieved by making a network call to the JPL server and ArcGIS server (one for river network and the other for terrain texture respectively). A bitmap is retrieved by the network call based off of the latitude/longitude, bounding box, and resolution specified. Calls for the textures are also done on application startup.

3.3.2. Local Data Processing/Management

Hydrology data from JPL's database is pulled via internet as a JSON format. Due to its key value pair and fast response time of O(1) . It is then parsed using GSON, which serializes and deserializes Java objects to JSON. The data then can be used locally in an arraylist of plain old Java object (POJO) in order to access their unique attributes such as unique id , storage units and time and date. In order to improve the processing of the data we decided to call upon the JPL database in various small increments instead of one large network call.

Data processing and management for the DEM and terrain texture is specified below.

3.3.3 Mesh Processing

Network call is made to JPL server to retrieve a DEM of the requested subset area based off of a bounding box surrounding the user's location. The bounding box has a width and height of 0.2 degrees along the latitude and longitude (about 22236 meters for latitude and 11118 meters along longitude when near the equator).

TIFFLib is used to create a TIFFImage object of the DEM along with its Raster and Directory methods for reading the pixel information of the DEM for constructing an arrayList of vectors consisting of xyz components; pixel location(x and y component) and pixel value(z component).

Vectors are then downscaled by 100 as its original range along the xyz values stretch out far. Afterwards xy components are then normalized to be in the range of negative to positive where the south/west is negative and north/east is positive, as they are originally all in the range of 0+. This is done so that the center of the mesh will be at the location 0,0,0 where the user's initial location is.

Indices are processed in quadrants (two triangles) in a counterclockwise direction (Left Top, Left Bottom, Right Bottom, Right Bottom, Right Top, Left Top).

An arrayList of vertices is made based off of the list of Vectors in the order of the indices, which is then passed on to the Vertex Shader for further handling.

3.3.4. Textures

Texture Coordinates are calculated in rows of quadrants(two triangles) in a counterclockwise direction. A higher resolution is used for the textures(800x800 or higher), as a low resolution (such as 400x400) will look blurry when seen up close. The range for the texture coordinate values along the width and height is 0 to 1 for any texture used. ArrayList of texture coordinates is then passed to the Vertex shader for further handling.

3.3.5. User Interface/User Experience

The user interface and user experience were a large portion of the project since we were using it help represent the hydrological data. We provided a compass on the main camera screen to help guide/orientate the user. The orientation views are comprised of a view for azimuth and pitch. The azimuth view aids the user with a horizontal angle, while the pitch view aids the user in determining the x-axis orientation of the phone. Both views were implemented using recycler views in which each view populated its own separate value from the gyroscope. The recycler review was made to be circular, so the data comes back around in a loop like a compass would. As we retrieved updates from the gyroscope, the views were updated automatically so it looked like the recycler view was scrolling on its own. An arc menu was implemented for one handed selection of different data types. The arc menu supports enabling multiple data types at once. The menu was borrowed by a third-party developer by the name of Moshen Hatami by which is credited in the source code.

A bottom navigation was used so the user can navigate between different views for the selected point-of-interest. The bottom navigation was an activity of its own that called to fragments, a user interface component, and loaded them onto the display. Those fragment components were the graph, list and map view. Open street maps was used for the implementation of the maps, taking the latitude and longitude of the point of interests and displaying it on the map. List view was a recycler view of the raw data that was retrieved from JPL's servers. Arraylists of the data are used in populating recycler view. The view also had a date picker component to select the start and end date of the data. Graph view was implemented by using the graph view library for android, displaying the data as line graph. Dynamic zooming of the graph was enabled with a simple parameter. For the x and y axis to be dynamic, we saved the x and y labels as strings, then set the number of labels that we wanted to display per axis (10 per axis), and upon resetting the values of the graph, the labels are dynamically updated.

3.3.6. AR Framework/Components

The framework provided by last year's senior design team had three main components: user interface, sensors, and rendering. We used the framework to render AR components to the screen to visualize the hydrology datasets. Changes made to the framework are as follows: perspective added to components for distance, positioning of components in relevance to real world positioning, and adjustments for scaling of terrain mesh.

Billboards size have been scaled according to their distance from the user. Points of Interest don't all have elevation values, but with the location of the POI we can grab the elevation of the corresponding location on the DEM.

The terrain mesh was the only new component added to the framework. It was loaded in during application startup and down sampled to compensate for device performance. An option for terrain textures are available for the user. Purpose of the terrain texture is to add detail to the terrain despite the down sampling.

4. Results and Conclusions:

4.1. Results:

We developed an application that represents the hydrological data using augmented reality components from version one's framework with a more robust and visual user interface. This allows for an interactive way for the user to learn about hydrology in their local area.

Historical data for the different points-of-interest now loads faster, making it more user friendly and less strenuous on the network call. Graph representation of the historical data allows for better data analysis as the user can see trends, not just raw data. Data can be filtered by the user by specifying a range in time to view trends during specified periods.

Additional components have also been provided such as line of sight and maps. Open Street Maps was used to implement maps as it is open source and it provides a nice way to help guide the user to the selected point-of-interest. Line of sight is an algorithm developed by JPL that returns the elevation of the terrain of where the device is pointing to.

The terrain mesh that is superimposed onto the screen is to scale in relevance to the real world and updates as the user travels. It requires little computation as it is down sampled, and textures are used to mask any rough edges from down sampling and to make the mesh more appealing.

All the various components that we wanted in the application were implemented successfully, but there was difficulty registering the terrain mesh properly since the framework solely relies on the gyroscope for positioning objects. In areas that interfered with the compass, the AR objects would not stay in their proper position.

4.2. Future Work:

As we were able to develop an Android application the visualize hydrology data by using augmented reality components, we were still only to accomplish so much during the school year. The following are improvements that we would have wanted to implement ourselves and would suggest for future work:

- Offline mode for areas with poor or no internet connection
- Better registration of the terrain mesh using computer vision techniques
- Improve the tracking of the framework by using the acetometer to track movement when the compass is being affected by external forces
- Cross platform deployment of the application such as iOS
- Large scale deployment of the application for public use by utilizing Amazon's Web Services
- Major design improvements to user interface
- Provide additional data from Watertrek

5. References:

Android Developer Website: https://developer.android.com/reference/

OpenGL es 2.0 standard:
https://www.khronos.org/registry/OpenGL/specs/es/2.0/es_full_spec_2.0.pdf

Graphview: https://github.com/jjoe64/GraphView or http://www.android-graphview.org/

OpenStreetMaps: http://osmdroid.github.io/osmdroid

ArcGIS REST Service: https://services.arcgisonline.com/arcgis/rest/services