Senior Design Final Report

BOE Sidewalk Assessment System

Version 1.0 - 05/09/2022

Team Members:

Aquil Alam,
Alejandro Chanocua,
Omar Eclicerio,
Ernesto Garcia,
Francisco Gastelum,
Henry Gonzales,
Gui He,
Perla Ramirez,
Rishi Shah,
Daniel Zeng

Faculty Advisor:

Jungsoo (Soo) Lim

Liaisons:

Ted Allen,
Alisa Blake,
Irvin Nguyen,
Christopher Tsangaris,
Jonathan DeLeon,
Miguel Grajeda,
Raul Virgen

Table of Contents

1. Introduction:

1.1. Background:

NASA's Jet Propulsion Laboratory (JPL) is world famous for their work in space exploration. JPL also does a lot of work in Earth Science. One example JPL's Watertrek database. Watertrek contains a vast amount of data on hydrological features such as wells, reservoirs, rivers, etc, but currently this data is readable only by scientists. By visualizing this data in a more meaningful such as with Augmented Reality AR techniques, this vast amount of information may be more approachable to the general public. Therefore, JPL has teamed up with California State University, Los Angeles to develop an AR Application to showcase Watertrek data. In addition, a Development Framework will be provided for other developers to use. The framework is designed for general purpose AR, meaning that the same techniques and libraries can be applied to other scientific data - not limited to just hydrology.

What is Augmented Reality? In short, Augmented Reality allows projecting information on to the real world. One way to do this is with a mobile device, such as a smartphone. When somebody points their smartphone at the world, the camera can display the real world onto the devices screen. Then images, 3D objects and text can be drawn on top of the camera Image. This makes it appear as if the images, 3D objects and text are actually in the real world, augmenting what the user sees.

The Augmented Reality App and Framework are both initially developed for Android devices, with the plan to expand to other platforms such as iOS or DJI drones at a later time.

1.2. Design Principles:

The Framework is the main deliverable, and the App uses the Framework in order to showcase the functionalities of the framework by visualizing hydrology data from Watertrek. The goal is to make the Framework general-purpose, meaning it should not be dependent of Watertrek or any specific data workflow. In addition, since the Framework may be used by software developers who may not be familiar with computer graphics technique, it needs to provide some interfaces that abstract computer graphics and still provide what the developer needs. The App to showcase the Framework needs to be simple, intuitive, and it needs to not make excessive calls to the Watertrek database, so that the app does not hog the database's traffic. Also, since Watertrek is still currently under development, the App needs to be simple in design so that future maintenance and expansions will not be too complicated.

1.3. Design Benefits:

By having an architecture that allows the Framework to expand with the demands of the developer, we essentially ensure its usefulness for future augmented reality visualization projects. The Framework aims to provide facilities for developers of all levels of expertise, so it can reach a broader audience, and their demands will drive its success in the market. The framework provides a simple interface that can be used easily by developers without OpenGL knowledge, while allowing additional features and components to be added by developers that do have that knowledge.

The App, on the other hand, is designed with simplicity in mind. Since the Framework does the bulk of the work necessary to set up augmented reality visualization, the App just has to concentrate on retrieving data from the Watertrek database, providing local storage, and providing a nice user interface. In this way, we keep the App simple and scalable, and this allows it to improve along with the Watertrek database, which is still under active development.

1.4. Achievements:

Over the course of the academic year, our team has been able to develop a preliminary Framework that provides augmented reality visualization of simple primitives such as billboards and 3D geometric objects. The Framework is currently built and hosted via a Maven repository, so any changes/revisions to the Framework does not require any changes/revisions to the implementations at all and the implementations only need to be rebuilt to reflect these changes.

An App that utilizes the Framework has also been successfully developed, and it can currently retrieve and visualize wells, reservoirs, and soil moisture data from Watertrek. The architecture for the App has been kept as simple as possible, and we have successfully decoupled most augmented reality visualization components from the App, so the Framework is solely responsible for how augmented reality visualization appears, and the App simply has to retrieve data. The Framework takes care of user permissions, sensors, camera, and 3D computer graphics for the App. There are still facilities in the Framework that allow for more in depth customization of the augmented reality visualization provided, and so the Framework may now scale from super simple projects all the way to very complicated projects that are heavily coupled with the Framework.

2. Related Technologies:

2.1. Existing Solutions:

We have looked into some existing frameworks for creating augmented reality visualization applications. These include Apple's ARKit, Google's ARCore, and Vuforia. While these solutions do allow us to reach our goal with this project, they come with certain restrictions and limitations.

Apple's ARKit is specific to Apple's iOS platform, and it is only available on version 11 of the operating system, with further restrictions on the different devices that can access this framework. So ARKit does not work for us if our intention is to reach a bigger audience.

Google's ARCore is in beta form, and it is very constrained in terms of performance. A very powerful Android device is necessary in order to run ARCore at reasonable performance. Furthermore, ARCore is only available on Android version 8.0 and upward, and again, much like Apple's ARKit, it is limited to only a few devices.

Vuforia is a reliable and proven multi platform AR framework. However, we decided against using it initially because of their approach to AR (as discussed in the next paragraph) and because there is a licensing cost.

The main approach of the AR frameworks mentioned above is to use Computer Vision techniques to recognize features in an image (such as the image from a smartphone's camera). When the device moves, the movement of the features on the screen can be detected. This allows the AR framework to render objects in the correct place on the screen, even as the device is moving.

This approach works well at detecting motion, but suffers when the registered features are not within sight (e.g. far away or obstructed by walls, trees or other objects). Also, while Computer Vision has come a long way at recognizing that a picture contains a building or mountain, these techniques have a hard time quickly identifying which building or mountain.

Since our AR framework is meant for representing distant specific objects, we need to use a different approach. Instead we are using GPS location and rotation of the device along with the coordinates of the object to be tracked to determine the proper place to draw objects on the screen. We did not find existing frameworks provided this functionality and thus were required to write our own.

2.2. Reused Products:

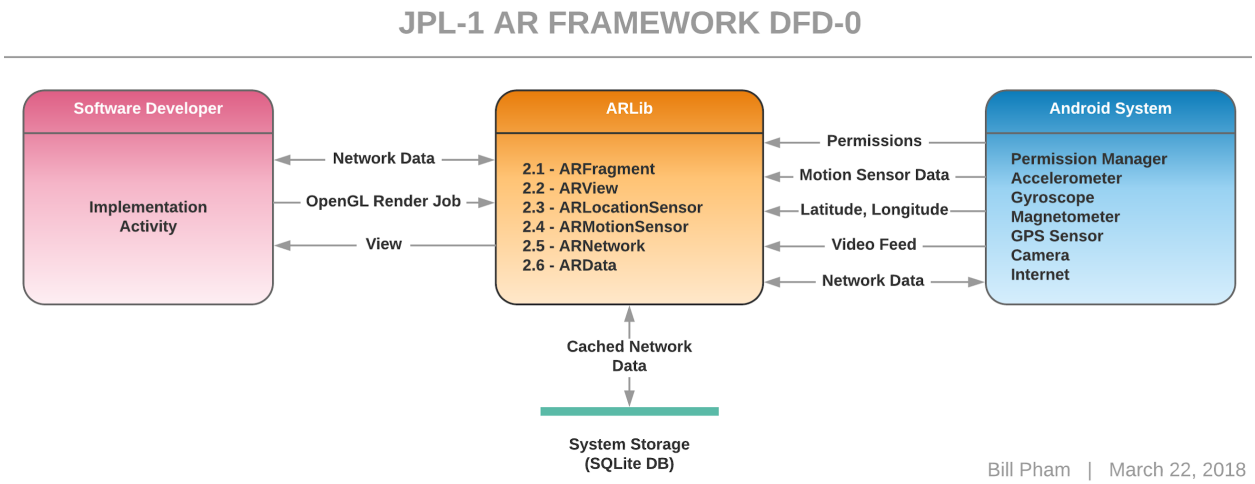
Both the Application and AR Framework are developed in Java, using the Android SDK. The Android SDK provides facilities such as Sensor Access, UI Elements, Camera Access, Network Access, and OpenGL es Drawing Routines. SQLite Database Management System is used for local storage for various operations on data, such as filtering search results.

3. System architecture:

3.1. Overview:

The architecture for both the Framework and the App is the same and can be broken down into three main factors: the User (software developer), the Framework (ARLib), and the Android system.

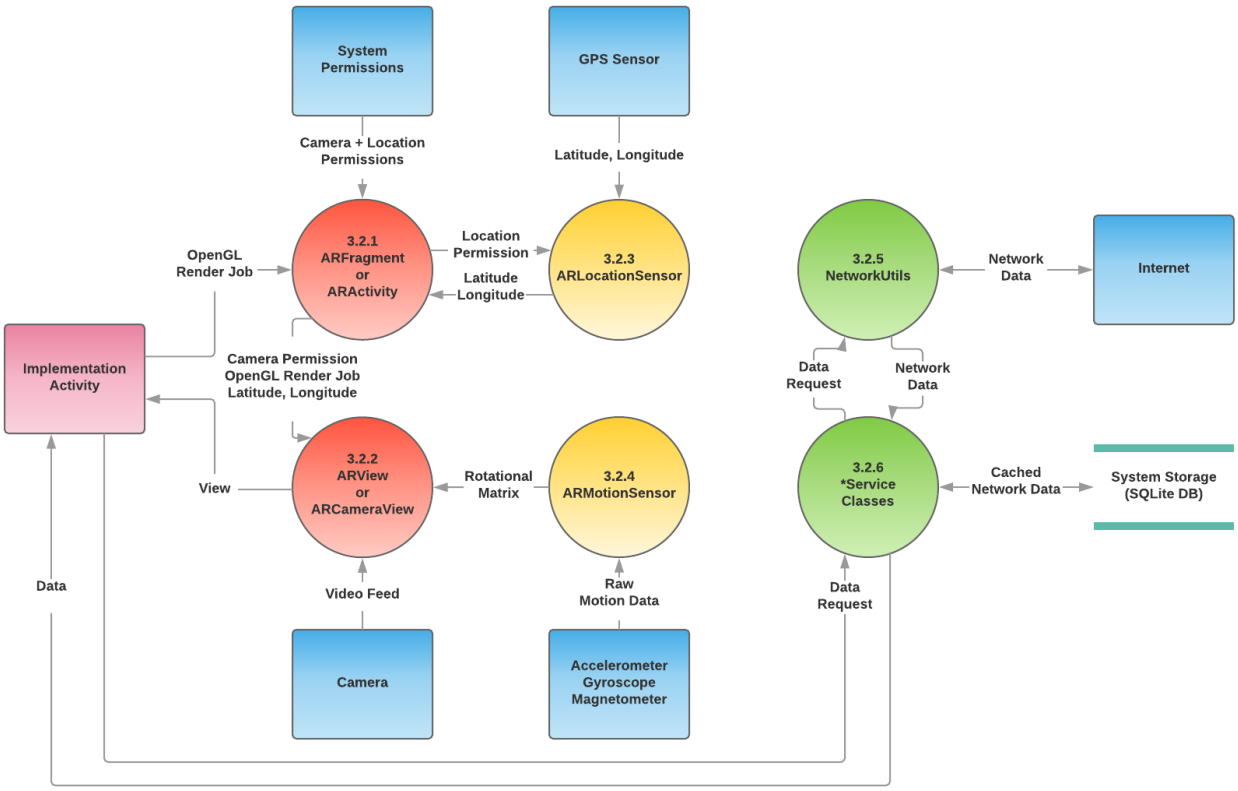
Here is a diagram (DFD level 0) that shows how this architecture works at a high level:



- **The User (Developer):** this is a factor in the architecture because considerations need to be made on how the Framework will be used. To that end, one of the team members, Christopher Hung Nguyen, acts as the user for the project, and his work on the App embodies this factor. Whatever the user requires in order to implement his App is exactly the base functionalities that we have to provide in the Framework.
- **The Framework (ARLib):** this is the main goal of the project. The Framework provides functionalities that the User can utilize to implement their App. Also, the Framework acts as an intermediary between the User and the Android System to abstract and take care of many of the setup procedures that the User would otherwise have to do by themselves. For instance, the Framework automatically takes care of setting up permissions, motion sensor data, camera, and OpenGL ES rendering, so the user does not have to go through the trouble of doing that.
- **The Android System:** through the use of Android SDK, many interfaces are available that allows for more convenient access to hardware features that are available on the device. For instance, Android provides an Orientation sensor, a mixture of the accelerometer, magnetometer, and gyroscope depending on how the system is setup (this is potentially different depending on the hardware of the device), so this sensor can be used directly without us having to implement our own code to do about the same thing.

3.2. Data Flow:

Here is an overview of the Framework as a system, and how it connects to the App (Implementation Activity), incidentally, this is also our DFD level 1:



There are six major modules in this system. They are described in more details in section 6. Here is a brief overview of them:

3.2.1. ARFragment / ARActivity: these are separate classes but they act as one module, as the functionalities they provide are almost the same, and it comes down to whichever style the developer prefers. These components are specific to the Android interface or subsystem, and as such, they are provided only as a convenience for the user. Both of them automatically take care of setting up permissions, sensors, camera, and OpenGL rendering so the user just has to implement what they would like.

3.2.2. ARView / ARCameraView: both of these are, again, the same and they act as one module. These components are implemented by ARFragment or ARActivity, respectively, to provide a camera view and an interface for the motion sensors. The user may choose to directly implement either of these views to create their own activity or fragment with a custom behavior that is more tailored toward their needs.

3.2.3. ARLocationSensor: this is an interface that abstracts the GPS sensor's functionalities. It is further abstracted by either the ARFragment or ARActivity to provide data to the user without making them go through the trouble of setting up permissions for this.

3.2.4. ARMotionSensor: this is yet another interface that abstracts the motion sensors' functionalities. It is further abstracted by either the ARView or ARCameraView to provide data to the user without making them go through the trouble of having to set it up.

3.2.5. NetworkUtils: This is an interface that is specific to the app, and is not part of the Framework. This interface allows the user to make a connection to the internet without having to go through the procedures of having to set it up. It can be directly referenced by the user, but in this system, it is utilized by the *Service classes to provide data for the user.

3.2.6. *Service classes: these are described in more details in section 6.2.0. Essentially, these are the various service classes that provide an interface to connect to the Watertrek database and retrieve Hydrology data.

3.3. Implementation:

The project was split into four sections to allow for efficient development: Watertreak and hydrology data, local data management, user interface and experience, and the AR framework. Each section plays a key role in presenting the progression of the project.

3.3.1. Watertrek and Hydrology Data

Hydrology data is gathered using JPL's REST API and the project's framework's back end. Overlaid digital elevation map (DEM) data is loaded locally to indicate altitude. Each pixel in the raster layer of the DEM corresponds to a range of coordinates and the visual representation of the pixel represents the elevation and altitude value.

3.3.2. Local Data Management

Using an internet connection, raw data from JPL's server is stored in a long string and is parsed before being pushed into the local database. The data can then be pulled from the local database and be displayed. Data may be updated and overwritten when an internet connection is available.

3.3.3. User Interface/User Experience

A user interface was designed specifically for the project to visually present the framework. The UI allows for interaction with the functions of the framework without accessing its components or requiring any knowledge of how to use it.

3.3.4. AR Framework

The framework contains three major components: user interface, sensors, and rendering system. The user interface provides a surface for displaying camera output with 3D graphics drawn on top. The sensors provides GPS coordinates, altitude, accelerometer, gyroscope, magnetic field, gravity, and orientation of the

device. The rendering system makes use of drawables, entities, the camera, and a scene.

4. Conclusions:

4.1. Results:

We have created a framework that allows rendering 3D objects, 2D objects and text on top of a smartphone's camera output. This gives the appearance of the graphics/information existing in the real world. By taking the smartphone's location and orientation sensor data and forwarding it to the rendering system, the graphics adjust as the device is moved, thus maintaining them in the correct position.

We have created classes and methods that allow easy access to many of JPL's Watertrek data. This is not only needed for the application we are producing, but can be used in apps from other developers.

The database module of our app stores the downloaded data locally, and allows efficient processing of the data as needed by the application.

Finally, these parts have been brought together with a aesthetically pleasing and well designed user interface to showcase the developed technology.

4.2. Future:

As discussed earlier in section 2.1, AR is not new. The techniques of using the GPS and sensors of the device along with the coordinates of real world objects to properly render the AR Objects have been used before as well. However, our project makes an important and significant contribution to AR by making these techniques available in a framework that can be quickly and easily adapted to many applications. There are many potential uses in for this software related to learning, research, guided work and entertainment. There is much still to do and we are happy to know that this project will be carried forward by following design groups at California State University, Los Angeles. Following are suggested improvements for the next year's team to work on:

- Improvement to tracking of the device's location. Right now location is accomplished strictly with the device's GPS sensors. This is a very coarse and sometimes unreliable reading. Improvements could be accomplished with filtering and combining GPS location with accelerometer/gyroscopic sensor data or by using Computer Vision techniques.
- Adding features to the rendering system. This could be accomplished by adding new rendering primitives, or connecting the AR components of this framework with an existing rendering library.
- Add the ability to retrieve terrain elevation data from the network and generate a mesh from this data. This mesh could be rendered on top of the existing terrain to

allow highlighting geological/hydrological features in the world. Even if it is not rendered (or is transparent) this mesh could have other uses, such as allowing terrain features to be clickable, or to prevent drawing objects that are not within line of sight.

- Creating general purpose components for use in the Network and Database subsystems.
- Expanding to other platforms.

5. References:

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

Vuforia Documentation: https://library.vuforia.com/

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

OpenGL shading language 1.0 standard:
https://www.khronos.org/registry/OpenGL/specs/es/2.0/GLSL_ES_Specification_1.00.pdf

SQLite documentation: https://www.sqlite.org/docs.html