Augmented Reality (Field Equipment)

Pacific Gas and Electric Company

Documentation

StakeHolders: Matt Rettagliata, Pete Baker, John Nichols
Group Members: Edward Han, Enoch Hang, WoongHee Lee, Sally Wei

Table of Contents

Preface

This document is presented by undergraduate students of the University of Southern California Viterbi School of Engineering enrolled in *CSCI 401: Capstone and Design and Construction of Large Software Systems* with Pacific, Gas, and Electric Company (PG&E) as stakeholders

Project Proposal

In the physical world, we are surrounded by tangible technology as well as their benefits and safety risks. For example, in the field of electrical engineering, a technician may operate on faulty devices or equipment full of electrical wires, which requires care and precision. It would be convenient to have an on-hand interface automatically pull up accurate and important statistical, geo-spatial data regarding the surrounding terrain or details of the device being fixed, that streamlines the technical operation. This is preferred over the options of phoning an outside source for an information look-up or using more cumbersome methods such as paper documents and laptops.

As a result, we have partnered with Pacific, Gas, and Electric Company to develop a mobile application utilizing augmented reality for our capstone project that would provide field workers better **situational awareness**. The goal is to produce a prototype or proof-of-concept that demonstrates a native, inexperienced approach to augmented reality, stressing the concept of a **minimum viable product** (MVP) that implements AR in gas and electric field work.

Project Details

Members/Participants

Stakeholders (PG&E)

1. Matt Rettagliata
2. Pete Baker
3. John Nichols

Developer Team (USC)

1.	Edward Han	hane@usc.edu
2.	Enoch Hang	enochhan@usc.edu
3.	Woonghee (Brian) Lee	woonghel@usc.edu
4.	Sally Wei	sallywei@usc.edu

Meeting Times

Jan 17 - Apr 13, 2018 on Fridays from 2:00 PM - 3:00 PM

Proposed Deliverables

We include our intended roadmap for the project below.

Deliverable #2 (February 5, 2018)

- Finalization of required document which may include:
  - User Stories
    - Possible augmented reality data set(s)
    - Technologies needed for the project
      - Different approaches to AR, with alternatives
    - Determination of the Minimum Viable Product (MVP)
  - Complete AWS Practitioner Essentials

Deliverable #3 (February 19, 2018)

- Sprint 0
  - Discover platform to use

Deliverable #4 (March 5, 2018)

- Sprint 1
  - Document any failures/findings/limitations
  - Why we want to experiment in AR
  - Focus on data format and development of platform.

Deliverable #5 (March 19, 2018)

- Sprint 2
  - Iterate and improve on previous sprints.
  - Backlog grooming
    - Prepare for adjustments moving forward

Deliverable #6 (April 2, 2018)

- Sprint 3
  - Iterate and improve on previous sprints.
  - Backlog grooming
    - Prepare for adjustments moving forward

Deliverable #7 (April 16, 2018)

- Delivery of MVP
- Documentation of project learnings (as important, if not more than MVP itself)

User Stories

1st Draft (Feb 5, 2018)

1. Field Equipment Stories

As a / an ...	I want to...	so that...
Field Worker	retrieve information about a device by scanning a serial code on it	I can fix the device that I am working with
Field Worker	scan a device that I am working with live	also retrieve on-demand information about it and its status.
Field Worker	actively scan the surrounding area of a device that I am working with	I can see additional information about power lines nearby
Field Worker	add information to the device augmentation	I can update equipment information

2. Pipeline Stories

As a / an ...	I want to...	so that...
Field Worker	visualize the pipelines on the device beforehand	The digging process around the pipelines is more structured and systematic
Field Worker	After raising your phone, be able to see an overlaid visual of the pipe or wire or device color-coded by age, material, or some other data value	to aid in repairs or maintenance checks

3. Electricity/Gas Stories

As a / an ...	I want to...	so that...
Field Worker	Have a projected, digitized layout of the electricity lines	I can safely assess electricity lines with minimal physical interaction and make necessary adjustments or repairs without

		disrupting lines that do not need to be disrupted
Field Worker	auto-display data such as the customer (owner), name, ID	I can measure electric/gas meters in an efficient manner with all necessary data laid out at once
Field Worker	Be able to see where meters in a large building are in a projection	I can save time reading a map and trying to find where the meter is.

4. Other

As a / an ...	I want to...	so that...
User of the App	be able to log in with an authentication system [Auth0]	I can access information.

2nd Draft (Feb 19, 2018)

Additional User Stories/User Stories Revisited

To supplement our devised user stories in Deliverable #2, additional, more specific user-stories and details are provided below on behalf of our stakeholders, who have engaged in direct contact with field-workers on-site to produce these “solutions”. These user-stories helped us shape our MVP proposal.

User Story #1

User would look at an asset in the field, and the system would overlay heat signature and other sensor based readings.

Questions:

- What is the reason for overlaying heat signatures? Would it help to see the flow of things like water, gas?

Notes/Answers:

- Related to User Story #3
- The term “user,” to be more specific, refers to titles such as trouble-man, general construction repair crew, field workers, and so on.
- Assets can refer to transformers, distribution boxes, circuit breakers, energized high voltage, high powered devices, etc. It **could be useful to point to the asset and recognize the device in question**
- **Engineering experts can recognize the humming of a device and its associated frequency in say, Hz, and be able to conclude a malfunction**
- Environmental factors, who worked on it last, model number, etc.
- The goal is to aid system maintenance, **enhance situational awareness**, super-charge their instincts (augment their expertise)
- Cut down troubleshooting time, lead them to a direction that they would otherwise not explore
    - Differences between an electrical substation case, and electrical lines case
        - 2 different lines: transmission line, distribution line
        - In a major disaster, it would be useful to pull up statistics that tell whether the lines are charged or not charged
- Field workers that call the dispatch/control center would receive a response that a line is not charged which assumes the line’s only source of power is from the responding control station. However, there is always the potential for something to go wrong. It would be helpful to have an additional level of certainty, real-time overlay of sensory information

- A maintenance worker for a power plant would like to be able to identify a device's model by pointing their phone in the direction of the device, so that they can speed up their work and have a way to double check their intuition.
- A field worker checking power lines after a disaster would like to be able to tell if a fallen power line is live or dead by pointing their phone at it from a distance, so that they can have a quicker feel for the situation and make clearer judgements.

User Story #2

User would look at an asset in the field, and the system would provide an asset tag and be able to stream data pertinent to the work they are doing.

Questions:

- Is the asset tag the identifier? What are some examples of what would be useful/helpful "pertinent data" to stream real-time?
- How consistent would this asset tag be across all devices?

Notes/Answers:

- Depends on type of device, and distance of scanning device from devices
    - Meter, capacitor bank (used for voltage control, reactor power control)
    - Might not be able to take out badge number from something like capacitor bank
    - Could be tricky when multiple poles are side by side

User Story #3

User would look at an asset in the field, and the system would provide highest safety risks and work procedures.

Questions:

- Any specifics on what the highest safety risks and work procedures exactly entail? For example, how would the app itself provide safety, and if there are accidents, what safety measures are prioritized?

Notes/Answers:

- The provided risks and procedures are generated from a database, which in itself draws from research and public information
- There are company risks and company assets (around 25 – 30ish for each device)
- So when you scan a device, the procedures for the specific device and the TOP 5 biggest risks would be pulled up on the screen
    - Recommendation: mock up the risks, pulling the data from PG&E could be time-conducive
    - Be general: gravity, falling objects, etc. (for the MVP)

User Story #4

The user would look at the ground and see underground assets and their states (charged or pressurized).

Questions:

- What do the terms: “charged, pressurized”, mean for assets?

Notes/Answers:

- DMS system is where the information would be pulled from

3rd Draft (Mar 5, 2018)

General Application User Stories

* See [2] under references for how these user stories were revised and enhanced

Assets:

- transformers, electrical substations, distribution boxes, circuit breakers, energized high voltage, high powered devices

User Story #1

User would look at an asset in the field, and the system would overlay heat signature and other sensor based readings.

(a) As a technician I want to overlay infrared signatures on transformers by measuring their apparent temperature difference or contrast radiant intensity (CRI), so that I can determine abnormalities in transformers by looking at their temperatures (as an extra indicator). [1]

- Identify the transformer in the field by its location
- Retrieve infrared signature from the transformer
    - Measure a transformer's apparent temperature difference
    - Measure a transformer's contrast radiant intensity
    - Correlate the received information with the identified transformer
- Overlay signatures on top of the transformer

(b) As a technician I want to overlay information about a target asset's current energy states from a mobile device's infrared or asset voltage sensor based readings so that I can properly assess the safety of an operation without hurting myself.

- Determine a target asset
- Retrieve the target asset's current energy state
    - Retrieve the target's infrared readings
    - Retrieve the target's voltage sensor readings

- Overlay the information on the target asset

- **S**pecific because it specifies system information about an asset's current infrared and voltage levels from sensors.
- **M**easurable because a technician can clearly see the system information or lack thereof being displayed on the screen, without additional extraneous user-induced steps. The technician can verify and compare the sensor readings with actual known data about a specific asset for any discrepancies.
- **A**chievable because we can simply query information from PG&E database and overlay the information we want onto the mobile device, alongside sensor-read information.

- **R**ealistic because if there is already a system that draws system information from assets, the information only needs to be displayed together. This system would utilize built-in sensors on a mobile device or peripherals in conjunction with the PG&E database.
- **T**ime-related because this user story can be completed within a manageable time frame and depending on its priority can be assigned a certain level of urgency. As long as the team can understand how to work with the sensors and extract relevant data, we should be able to overlay the required information.

(c) As an electrician, I want to project information from sensor based readings within an electrical substation so that in a major disaster, I can pull up useful statistics that would tell me whether transmission and/or distribution lines are charged or not.

- Detect if in an electrical substation?
- Detect nearby transmission lines
- Detect nearby distribution lines
- Read whether a detected line is charged or not
- Project the reading onto the screen, in a coherent manner (overlay the nearby line?)

User Story #2

User would look at an asset in the field , and the system would provide an asset tag and be able to stream pertinent data to the work they are doing.

(a) As an electrician, I want to be able to read a tag on an asset to request additional information about the asset onto my mobile device, so that I can better understand the situation that I am attending to and enhance my situational awareness.

- **S**pecific because it specifies the action requested (scanning an ID tag) and response (additional information to enhance context of the device).
- **M**easurable because an electrician scans a tag, with the mobile device's camera capturing the entirety of the tag held in place for at least 1 sec, and can then verify that the device will display the intended additional information onto the screen or the lack of thereof.
- **A**chievable because we can use text or image recognition to capture the tag's information, parse it if necessary, lookup and retrieve the corresponding information in the PG&E database, and display it onto the device in a user-friendly manner.
- **R**ealistic because we have all the resources necessary to achieve this. We assume the tags would be designed to be parsable in a consistent manner across all PG&E assets and that the existing PG&E database holds accurate information. Current augmented reality technology for text recognition on mobile devices exists as well.
- **T**ime-related because this user story can be completed within a manageable time frame and depending on its priority can be assigned a certain level of urgency. As long as the tags are parsable and recognizable in a consistent manner the team should be able to complete this user story in a timely manner.

(b) As a repairman, I want to be able to read an ID tag on an asset to be able to request model information about the device so that I can make less mistakes and identify devices I'm less familiar with.

User Story #3

User would look at an asset in the field, and the system would provide highest safety risks and work procedures.

(a) As an electrician, I want to see the most prioritized safety risks and work procedures while assessing a device so that I may proceed carefully and with more awareness.

(b) As an electrician, I want to be able to visualize the highest prioritized safety risks about pipes around me as information on my mobile device, so that I can more safely and cost effectively work around these situations.

- **S**pecific because we want to focus on the most important safety risks about pipes (ideally the top 5 safety risks) in order to be safe and cost effective during operations. Safety would entail prevention of any physical injuries, while cost-effectiveness would entail using less resources and operating under less time when possible.
- **M**easurable because the mobile device should display the top safety risks, and any helpful widgets or alerts about the surrounding pipes and the operations to be performed on them.
- **A**chievable because as long as the data for safety risks and cost-effectiveness exists, it can be formatted concisely and then simply be displayed on the mobile device.
- **R**ealistic because the data for theses risks as well as data about pipe architecture exists in the form of GIS data or as PG&E data. Estimates for cost-effectiveness can be made on past statistics, research, experience, or data, in conjunction with the data for safety risks.
- **T**ime-related because this user story can be completed within a manageable time frame and depending on its priority can be assigned a certain level of urgency. As long as the data can be accessed and be presented in a clear manner, the team should be able to complete this user story in a timely manner.

User Story #4

User would look at the ground and see below ground assets and state (charged or pressurized).

(a) As an electrician, I want to be able to visualize the pipes and assets of companies in the area that I am working in, through a virtualized object representation, so that I can be cost effective and safe in completing my task.

- **S**pecific because we are trying to visualize the specific pipes and assets for PG&E with a virtual overlaid version of the objects prior to technical operations, to prepare for safety, to be cost-effective (see User Story 3b), and to have an understanding of the

architecture of the pipes and assets that may otherwise not be easily accessible or available on-site.

- **M**easurable because an electrician will be able to see a color-coded, virtual representation of a pipe or asset and/or its transparent internal architecture on the mobile device, to provide preemptive examination of the target object.
- **A**chievable because we would use locational data from the PG&E database to recognize a specific pipe or assets in the field, and incorporate plane detection and object placement to achieve virtualized object representation.
- **R**ealistic because current augmented reality technology allows plane detection and object placement, which would make this user story possible.
- **T**ime-related because this user story can be completed within a manageable time frame and depending on its priority can be assigned a certain level of urgency. If the team can successfully emulate object placement on the basis of some data (e.g. asset at a location), the team should be able to successfully complete this task.

(b) As an electrician, I want to see whether an asset (ie a pipe in my area) is either charged or pressurized so that I can work safely around or with them when necessary.

4th Draft (Mar 19, 2018)

1. As a technician…
    - ...I want to overlay **temperature readings** of the target asset on a mobile device camera, so that I can clearly see the readings and where they correlate to on the asset.
    - ...I want to overlay **current voltage levels** of the target asset on a mobile device camera, so that I can clearly see the levels and where they correlate to on the asset.
    - ...I want to be able to **remotely** interface with the target asset through temperature readings from a safe distance so that I can reduce any risks of burn as a result of interacting physically or directly with the target asset

2. As an electrician,
    - ...I want to **scan an ID tag** on an asset so I can retrieve information even when the program fails to automatically identify an asset.
    - ...I want to pull up the **name of the person** who last checked the asset so that I can speak with them later if necessary.
    - ...I want to pull up the **maintenance history** of the asset so that I can gain context about the condition of the asset.

3. As an electrician, I want to be able to visualize the highest prioritized safety risks about pipes around me as information on my mobile device, so that I can more safely and cost effectively work around these situations.
    - ...I want to overlay the **top 5 safety risks** when working with pipes onto a mobile device camera so that I can educate myself on how to more safely perform an operation on the target pipes site.
    - ...I want to identify the nearby asset **with geospatial data** so that I can identify the device safely from a distance.
    - ...I want to **retrieve safety risks** for the relevant asset **from the database**, or identify them if they cannot be retrieved so that I can more quickly assess a situation.
    - ...I want **another page** to appear post clicking one **of the safety risks** so that I can be more informed about the risks that must be addressed.
    - ...I want to see the **top safety risks in order** by its risk level so that I can better prioritize safety risks. (Prioritize safety risks by danger level, situational relevance)

4. As an electrician,
    - ...I want to visualize pipes through the camera display so that I can see how to avoid them.

- ...I want to interface with the visualized pipes and electric lines so that I can be sure to not work with live lines or active pipes.
- ...I want to visualize pipes with depth around me so that I can better plan to avoid them.
- ...I want to be able to pull up the **model** (virtual object representation) of the asset so that I can gain context about the condition of the asset.

Minimum Viable Product (MVP)

1st Draft (Jan 22, 2018)

- At bare minimum we want to display PG&E or just general data from a database on a target device in question
    - To make this useful with Augmented Reality(AR), it would involve scanning/identifying some device through some identifier to display the data
- Recognizing different device types and being able to overlay relevant data accordingly

2nd Draft (Feb 5, 2018)

MVP Proposal

The minimum viable product will be one that demonstrates the fundamental capabilities of augmented reality to assist in the situational awareness of field workers. For example, the ability to gather distributed information about a field asset and present it to an on-site worker after they have pointed the device at an asset and identified it somehow (either manually or by the application automatically via the asset's unique location, an ID, or some other method). This distributed information can include information such as device type, make and model, last time of maintenance, last maintainer, location, important safety measures, and specific measurements associated with the device. Information can be displayed either directly on screen as in a pop-up, or it can be overlaid on the visual itself (ie a visual temperature reading of a device or the location of an underground pipe).

- Identify an object by pointing in its direction (visual recognition).
- Identify an object by its location.
- After identifying an object, draw information from more than one source and display it on the screen.
- After identifying an object, draw real-time information from a source and display it overlaid on the screen (next to the object or overlaid on top of the object).

3rd Draft (Feb 19, 2018)

A Web AR application that can display a pipe in AR and its relevant information according to the pipe's geographical coordinates or to an ID tag for the pipe.

Refer to User Stories:

1(b) [read an ID tag to present a 3D model]
2(a) [read a visual tag for additional information]
3(b) [visualize safety risks about nearby pipes]
4(a) [visualize nearby pipes]

4th Draft (Mar 5, 2018)

A Web AR application that can display an asset whether pipe or device in AR and its relevant information according to the its geographical coordinates or ID tag. This application will be used to help bring greater safety and efficiency to PG&E and their fieldworkers that directly work with its assets.

Sprint Planning

The drafts below show the team's progression of sprint planning, starting from an initial plan, to later revised sprints.

1st Draft (Mar 5, 2018)

Sprint 1

- Web AR (Google three.ar.js) on WebARonARKit on an iPhone 7+ (or WebARonARCore on an Android device). Develop a basic application that demonstrates the following in combination with AR:
    - Plane detection
        - *An extension of three.ar.js supports this. (WebVR API extension for smartphone AR)*
    - Text recognition
        - *Tesseract, the most popular OCR (optical character recognition) software, has been ported to JS. It currently supports reading from all sorts of objects (canvas, blobs, pngs) but does not currently (explicitly) support VRDisplay (an abstraction of the AR device used by three.ar.js)*
            - *Does VRDisplay.stageParameters count as an ImageData instance (contains width, height, data)?*
            - *Alternately, capture the camera image (or video?) separately from VRDisplay and feed it into Tesseract*
            - *There may be performance considerations about doing this in quick succession (e.g. for every frame) since Tesseract seems to run pretty slowly (ref. Tesseract's official demo)*
            - *Downloading and unzipping the language data takes the most time for the English and Russian demos, whereas for the Chinese one, recognizing the text takes up the majority of the time*
            - *Stop reading when Tesseract has read something successfully? How fast is Tesseract to respond if the image given doesn't have text in it?*
        - *Wikitude SDK Plugin API integrates AR Core and AR Kit into it in something they call "SMART" - seems like Wikitude is a separate development environment that builds on top of AR Core and AR Kit in order to provide a JavaScript environment.*
            - *They call it ARchitect View*
        - *TextRecognizer: API from Google Vision that allows the phone camera to detect and recognize the text in an image. It's not too complicated to implement but because it's fairly new, it has less functionalities than others.*

Cannot limit the words it recognizes in real time. Also, this may be compatible with Android but not with web AR.

- Object placement
    - *Object placement is supported by three.ar.js to the level of placing items at a "hit." Placing objects based on some other event (not OnClick()) still needs to be looked into and tested.*

Sprint 2

- Object recognition and tracking.
    - Detect an object by its coordinates
        - *Geolocation to get current location [of the user]. Possibly combine with other features to manually calculate an object's distance and extrapolate its coordinates. Then use those coordinates to query a database by location for the related object.*
    - Track and recognize objects
        - *Tracking.js provides both image and object recognition as well as a library for training on a custom set of objects/images*
    - Connect with GIS (read and use data from shapefiles)
        - *ArcGIS, ESRI's JavaScript API. Uses WebGL Scenes to display its data. Combining this with AR may be more complex than originally assumed, due to the different displays and underlying libraries (WebGL vs three.js) used.*

Sprint 3

- Display information based on identification of an object on screen
    - Display information about specific objects when pointing the camera at it.
        - *Recognize the object, pull its information, format the information, display the information on screen.*
        - Temperature
            - *Temperature-related JS libraries on npm - on a mobile device would this require a 3rd-party (physical?) sensor to be able to overlay heat levels on a camera image? Or is this already provided in a database?*
    - Display safety risks about an object when pointing the camera at it.
        - *Recognize the object, identify its category of risks, fetch those safety risks from a database or somewhere, then display those risks on screen (in "reticles" - three.ar.js provides this function).*

Sprint 4

- Visualizing pipes based on coordinate location from GIS.
    - Object visualization (eg drawing pipes)
        - *Three.ar.js provides a "graffiti" example that draws where the user touches. This would need to be modified to place the drawn object at locations dependent on fixed coordinates or some other measurement.*

- *Ref. "Phase 2: Detect an object by its coordinates" for a similar problem to positioning pipes to be drawn relative to the user's location*

References

[1] https://en.wikipedia.org/wiki/Infrared_signature#Apparent_temperature_difference
[2] https://jessica80304.wordpress.com/2008/08/04/smart-requirements/

2nd Draft (Mar 19, 2018)

Sprint 1 (Mar 5, 2018)

1. Set up an environment for AR application
    - (Android Studio, Camera permissions)
    - Run Android Studio project using a phone

2. As an electrician, I want to **scan an ID tag** on an asset so I can retrieve information even when the program fails to automatically identify an asset.
    - Identify the object's ID tag with text recognition
        - Implement text recognition function using Text Recognizer API
        - Set up letters/numbers for testing
        - Output the ID Tag for the user to check if it's correctly recognized
    - Create a submit button in order for the user to get information of the ID tag

3. Plane detection
    - Test ArCore's examples of usage of plane detection.

Sprint 2 (Mar 19, 2018)

1. Identify the asset with an ID tag
    - Identify the object by accepting the user's manually-inputted ID tag

2. Retrieving & displaying data (voltage, temperature readings, 3D device model)
    - Create dummy data tables for information about different types of assets
        - Fake voltage readings
        - Fake temperature readings
        - Placeholder 3D models (use OpenGL for compatibility with both Web AR Core and Android AR Core)
        - Safety warnings, priorities, icons, "further reading" details
    - Set up a local server to connect with android studio (Preferably SQL)
    - Learn how to interact between SQL server and android studio

3. As an electrician, I want to overlay the **top 5 safety risks** when working with pipes onto a mobile device camera so that I can educate myself on how to more safely perform an operation on the target pipes site.

- ...I want to **retrieve safety risks** for the relevant asset **from the database**, or identify them if they cannot be retrieved so that I can more quickly assess a situation.
    - Create dummy data tables for a list of safety risks about different pipes
    - Identify asset with automatic ID-tag scan or manual input of ID
        - Retrieve corresponding safety risks that are mapped to the id of asset
- ...I want to see the **top safety risks in order** by its risk level so that I can better prioritize safety risks. (Prioritize safety risks by danger level, situational relevance)
    - Safety risks data (in database) can be ranked by a number indicating a danger level

Sprint 3 (Apr 2, 2018)

1. Identify the asset with an ID tag
    - Using a submit button, send the identifier to the database using SQL server
    - Look up the identifier in the database
    - Return an error if the identifier is not found
        - (prompt for resubmission of ID)

2. Retrieve the necessary information from the server
        - Safety risks: retrieve from database, include in multiple pages, rank by importance
        - temperature readings, voltage levels
        - object representation/model of the asset
    - Output any retrieved information plainly on the phone
    - As an electrician, I want to see the **top safety risks in order** by its risk level so that I can better prioritize safety risks. (Prioritize safety risks by danger level, situational relevance)
        - After pulling list of safety risks from the database, rank safety risks for target asset and return the top 5 ones by danger level

3. Visualize pipes and relevant information
    - As an electrician, I want to visualize pipes through the camera display so that I can see how to avoid them.
    - ...I want to visualize pipes with depth around me so that I can better plan to avoid them.
    - ...I want to identify the nearby asset **with geospatial data** so that I can identify the device safely from a distance.
    - ...I want to be able to pull up the **model** (virtual object representation) of the asset so that I can gain context about the condition of the asset.

- ...I want to interface with the visualized pipes and electric lines so that I can be sure to not work with live lines or active pipes.

Sprint 4 (Apr 16, 2018)

1. Identify the asset with an ID tag
    - Allow the user to choose what type of data to display with buttons & an options menu

2. Display the necessary information from the server
    - Retrieve the information for the user's selection
    - Display the retrieved data
        - For temperature & voltage readings, display in a graph & plot data retrieved over time
        - For safety warnings:
            - Organize by priority
            - Display icons only for the major warnings
            - Hide lesser warnings under a "There are X more minor warnings, click here to read more" message
            - Click the warning icon or read more message to open a popup with further details
        - For model, display the 3D model.

    - ...I want **another page** to appear post clicking one **of the safety risks** so that I can be more informed about the risks that must be addressed.
        - Create the safety risks text clickable or surround it with a button with said safety risk text that is clickable
            - Once safety risk is clicked on, display a subview that displays information about the safety risk (another attribute of the same data row)
    - ...I want to visualize pipes with depth around me so that I can better plan to avoid them.
    - ...I want to be able to pull up the **model** (virtual object representation) of the asset so that I can gain context about the condition of the asset.
3. Display necessary information from sensors
    - As a technician, I want to overlay **temperature readings** of the target asset on a mobile device camera, so that I can clearly see the readings and where they correlate to on the asset.
    - ...I want to overlay **current voltage levels** of the target asset on a mobile device camera, so that I can clearly see the levels and where they correlate to on the asset.

- ...I want to **remotely** interface with the target asset through temperature readings from a safe distance so that I can reduce any risks of burn as a result of interacting physically or directly with the target asset
- ...I want to pull up the **name of the person** who last checked the asset so that I can speak with them later if necessary.
- ...I want to pull up the **maintenance history** of the asset so that I can gain context about the condition of the asset.

3rd Draft (Apr 2, 2018)

The following is a decomposition of the general user stories that we were using to produce our MVP. We break them down to the most basic of actions, functionality, development action but not in the form of user stories.

Our original stories:

1. Pull up information about an item (ie 3D models, related documents)
2. Overlay instrument readings from a target device
3. Safety risks
4. Display pipes and other nearby assets (e.g. wires)

Broken down:

1. Pull up information about an item (ie 3D models, related documents)
    a. Identify a device via geolocation
        i. Identify distance of the object from the user
        ii. Translate distance into global coordinates for the object using the user's coordinates as an anchor
    b. Obtain coordinates and data of objects
        i. Attach longitude/latitudes to Anchors of objects
    c. Text recognition for scanning letters or QR codes or other ID tags.
    d. Read in shapefiles of objects
        i. Extract data from the shapefiles (or GeoJSON)
        ii. (stretch goal) Having a server over a network to demonstrate/test caching of data.
2. Overlay sensor readings
    a. Retrieve sensor readings from an instrument attached to the AR device (e.g. magnetometer)
        i. Interface with instrument
3. Safety risk icon
    a. Visualize risk icon next to a device as an object onto a plane.
        i. First, identify a device.
        ii. Then, locate the device in relation to the camera/user.
        iii. Retrieve the risk icon.
        iv. Then, display the risk next to the device.

4. Draw Pipes
    a. Draw objects based on provided dimensions (with OpenGL)
        i. Be able to draw them with altitude
    b. Visualize an object's location based on provided coordinates with OpenGL
        i. Produce planes
        ii. Read locations from custom-defined 3D objects (through GeoJSON)
        iii. Be able to manipulate the drawn objects' dimensions
        iv. Draw them in AR according to coordinates
    c. Calibrate their locations.

Sprint 3 (Apr 2, 2018)

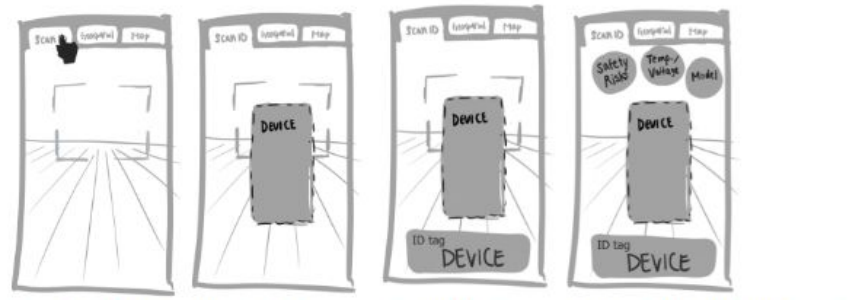
1. Display two different objects on a screen, like mixed assets or icons.
    a. How do you place an object on top of another object? (ie for overlays)
    b. Can you do this by placing multiple planes on top of each other?
        i. This needs to be tested, but a theory is that applying one object's Pose transformation to another object's Pose will move the first object to the second's location.
    c. Can you stamp a specific location to a longitude/latitude on a plane?
2. Determine the distance between objects and the user.
3. ArcGIS library learning

Sprint 4 (Apr 16, 2018)

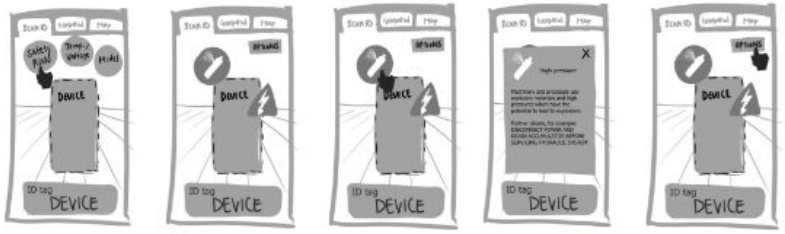
1. Determine Anchors for coordinates objects or planes (instead of using hits (taps on the screen) to determine location)
    a. Can it be done by placing objects at a certain latitude, longitude, and/or altitude?
    b. Calibrating the AR simulated space to real compass directions (North, East, South, West).
2. Calibration of the AR distances and projections with the real world.
3. Incorporating a 3rd and 4th dimension (time and altitude) to the object's 2D latitude/longitude data
    a. 3rd dimension represents altitude/elevation, since we don't want to just remain at sea level
    b. 4th dimension represents time interval
4. Mock-up data using the web app geojson.io

Wireframes

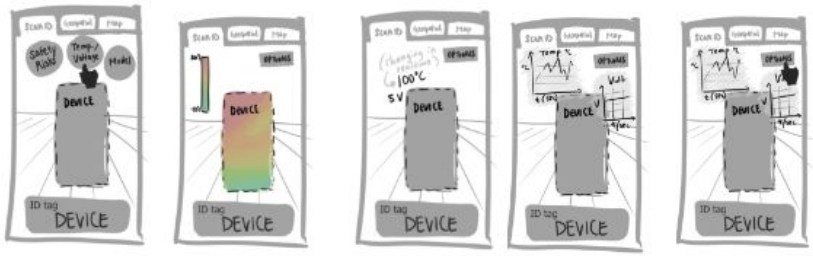
Mar 19, 2018



Select **Scan ID** tab | Point at device | Read ID tag | See options for what data to display

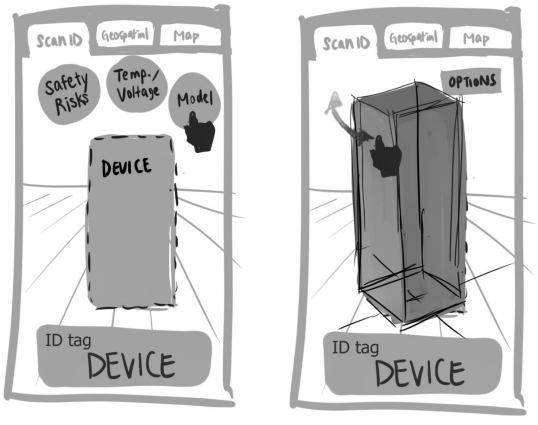


Choose an option | Display data | Select a risk | Read details | Return to options



Choose an option | Display (Fancy) | |<- Possible (Simple) Displays ->| | Return to options

---



Choose an option

Drag to rotate model

Request for Materials

Mar 5, 2018

Augmented Reality (AR) uses hardware built into the mobile device for positioning, specifically camera calibration techniques for spatial alignment in order to recognize 3D spaces and objects. This means that in order to test the AR functionality during development, it is required to have a physical device that is new enough to include hardware that can support AR.

Furthermore, since Web AR code itself is web-based, it requires a special prototype browser for testing that can support AR functionality. The two currently available browsers, both being developmental prototypes, currently only support a small subset of AR-compatible devices. Our team currently lacks any mobile devices that are compatible with either of the two available prototype browsers.

For development on Android, to test the AR functionality, the prototype browser WebARonARCore is required. This browser is only compatible with the following Android devices (the setup instructions for the browser specifies that they cannot be Android emulators): Pixel, Pixel XL, Pixel 2, Pixel 2 XL, or Samsung Galaxy S8.

Devices that are compatible with the prototype browser WebARonARKit for iPhone (iOS 11) include: iPad (2017), iPad Pro (9.7, 10.5 or 12.9 inches), iPhone 7 and 7 Plus. WebARonARKit also requires a Mac OS that runs the IDE Xcode 9 to deploy tests from.

Development of AR for a web browser, i.e. Web AR, has the benefit of being independent of the platform (Android vs iOS, WebARonARKit vs WebARonARCore) it is being deployed on. Web AR code can be written once and deployed on any device that supports an AR browser. Currently, because such browsers are only in the developmental phase, the number of supported devices is also small, but in the future it is likely that every device will be able to support an AR browser. The number of supported devices will also increase when AR browsers are officially released.

Although we looked into resources on our campus, we failed to find any compatible or accessible devices that could be freely used to test, deploy, and further develop on. We are looking for the provision of a compatible mobile device, preferably on the iOS side of AR to better match with our stakeholders' request for development on an iOS platform.

Organizations on Campus

Organizations and clubs on campus that can be funded for future research into AR.

GamePipe Laboratory
This lab seems focused on AI, machine learning, and emotion, rather than AR or VR.

VrSC
VrSC is a student club founded in 2016 to explore Virtual Reality. (Cinematic Arts Article)

Mixed Reality Lab
This lab is a part of USC ITP, working in collaboration with the School of Cinematic Arts' Interactive Media program, MedVR Lab, and Graphics Lab. Their focus is currently on VR, but can be expanded to AR.

World Building Media Lab (WbML)
Part of the School of Cinematic Arts' Media Arts + Practice Division. They focus on narrative worlds as vehicles for expression, but have engaged in mixed-reality projects like the Hay Grant Project (which utilized VR) in collaboration with other groups.

USC Information Technology Program (ITP)

USC Interactive Media program

Project Summary (Sprint 1)

* See Learning #6

Project Summary (Sprint 2)

AR Plane development

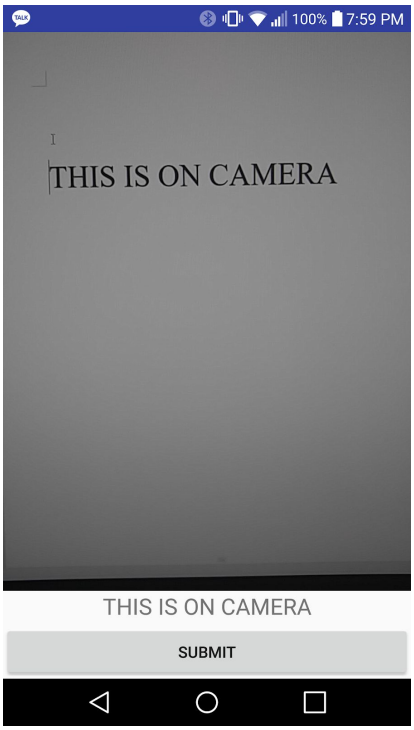
The discussion and presentation of utilizing the ArCore's Library for Plane recognition and development along with environment setup for future developers to also seamlessly produce and utilize.

Android Environment Setup

Developing augmented reality capabilities with Android Studio can be done with either of two options:

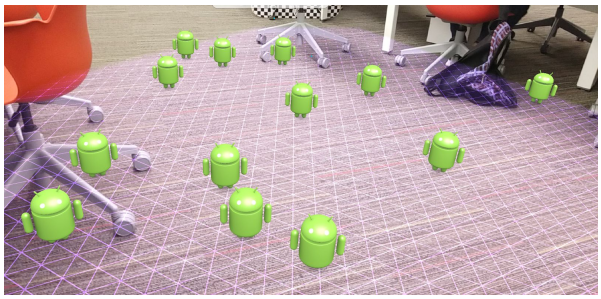
1. Android Studio 3.0.1+
    - This platform is preferred for development especially if you have devices to test with that are compatible with augmented reality.
2. Android Studio 3.2+ Canary 7+
    - This is preferred if you do not have a device that is compatible with ArCore to test with. Provided in this version are emulators of devices that are compatible and can be used to test your program.
    - Further instructions on setting up this environment, install the SDK for the correct emulator
        a. Go to the Tools tab to find the SDK manager to download the following found through checking the 'Show Package Details' box:
            i. Android 8.1 (Oreo)
            ii. Android SDK Platform 27
            iii. Sources for Android 27
            iv. Google APIs Intel x86 Atom System Image
            v. Google Play Intel x86 Atom System Image

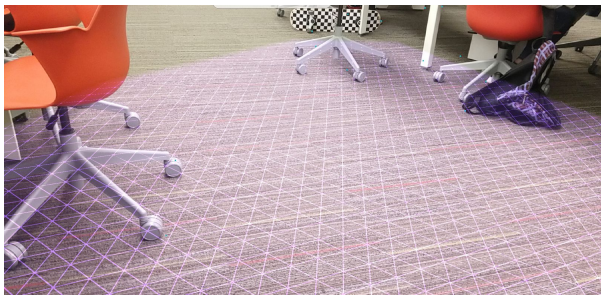
Implementation



Text Recognition

Using TextRecognizer API from Google Vision, the phone camera can be used to recognize letters and numbers. It is constantly looking for detectable letters/numbers and storing these words in an array of TextBlock. This is a simple demo of recognizing the phrase “THIS IS ON CAMERA” and showing the output below it. Later, we may be able to use this output as an identifier of an set to get any information about the asset.

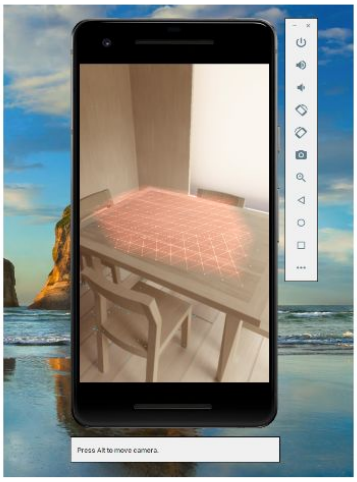




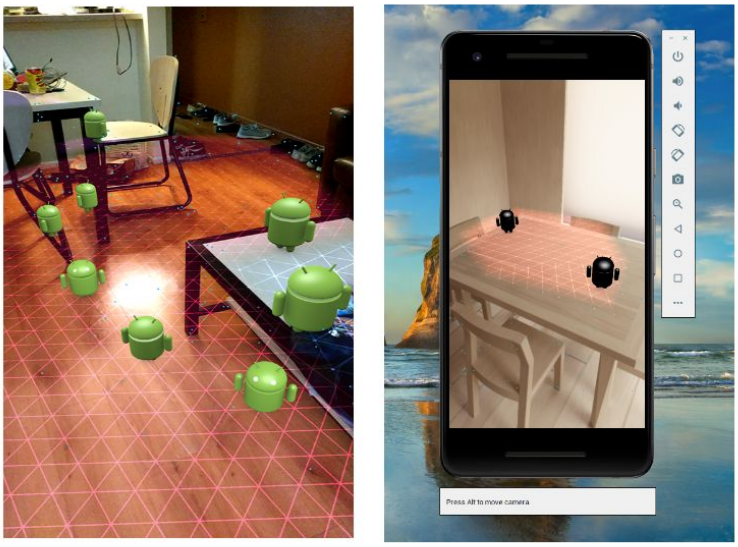
Plane Detection

Using Google’s ArCore Library, the essentials to creating the planes are the classes ‘Session’, ‘Plane’, ‘PointCloud’, ‘HitResult’, along with libraries ‘PlaneRenderer’ and ‘PointCloudRenderer.’ After using an instance of the session to update the frame and get the camera’s tracking state, each point that is being scanned is checked to see if it forms a polygon or rectangle similar to the plane’s. Once found, each polygon is made and formed together to create a connected plane. After all possible planes are found, they are drawn onto a matrix that would be drawn all within android onDrawFrame.

Sample pictures using ARCore on Android Emulator



Detecting planes



Placing object on the plane

This is a simple demo of plane detection using the same implementation we mentioned above. The difference here is that the android emulator is using a predefined environment whereas a smartphone would interact with the real world. Since an emulator does not support important features that phones have (such as camera and GPS location) and only few smartphones are currently compatible with AR, the ability to use AR on emulators allows more users to interact with current technology. While we are not planning to develop any further using emulators, this may be a great opportunity for others who may not have compatible devices.

Project Summary (Sprint 3)

Summary

We focused on expanding the flexibility of our current program. In this Sprint, we have implemented determining the straight distance between the user and a AR-placed object, calculating the coordinates of an AR object, and placing different models of AR objects.

Testing the Beta Emulator

Attempting to test code without a physical device means relying on the extremely new emulator (Android Studio 3.1.0 beta's AR emulator, released Feb 23, 2018). We in the team have had varying levels of success in using the emulator. Examples of issues encountered include an inability to install the actual apk for testing within the emulator, and an inability to start the app after it's been installed due to a failure in the emulator's AR camera.

Even when the emulator is working properly, it sometimes takes a relatively long time (on the order of tens of seconds) to recognize a surface.

All in all, the issues range from minor inconveniences to complete unreliability. For stability of testing, we recommend using a compatible physical device instead, although the emulator could work as a replacement.

Text Recognition

Text recognition is easily and readily available through a library. It has no complex dependencies or incompatibilities to our knowledge.

However, the library we are working with will read any and all text that appears on the screen by default, and must be changed manually to constrain where it will read text from. We have decided to set aside this issue for later, in favor of prioritizing AR-specific features.

Plane Drifting

Once a surface is recognized, the mapping of the surface will sometimes slide across the floor of its own volition. This is called plane drifting, and it is caused by phone sensors not being calibrated enough to keep it in place. This happens because the calibration between the phone and the camera can fail if the phone is moved too abruptly and the AR cannot keep up with it. The better the phone, the less drift there is.

What's Working Well

Placing Multiple Objects with OpenGL

At the beginning of this Sprint, object creation and placement was a black box. Our next step would be to determine how to place different kinds of objects, as opposed to placing multiple instances of one object.

Each object in the AR "hello AR" demo that we are building off of is represented by two ObjectRenderer objects: the object itself and its shadow. To place multiple objects, the new object must have its own ObjectRenderers defined for itself and its shadow declared in the onCreate().

Placing Objects Based on Latitude/Longitude

Is possible, but not as a direct integration with Location Services. We learned that we will have to manually take coordinates and transform them into local coordinates for use with AR Core's Anchor's Pose class. This is because the location services only returns the coordinates of the mobile device, not the location desired inside the camera. This problem has been solved before for OpenGL. However, since none of us are very familiar with OpenGL, modeling 3D space in computer graphics, or the math involved in such calculations, understanding these topics took time.

Something we discovered is that testing location services as a tool to be integrated into an AR plane requires good network service reception, so the device cannot retrieve its current location when, say, inside a building.

This issue has been raised before at least twice on ARCore's github, once where it suggests the use of magnetometer readings and rotation to align the device's space to geographical coordinates, and once suggesting emulating the HitTest function to calculate distance of an object to the device.

AR Core explains how its engine sees the world in this article on its developer site.

We have decided to calculate the object's coordinates based on the coordinates of the phone and the object's distance to the phone. The basic math changes the object's distance into the object's offset in Cartesian coordinates, then adds those to the device's location to get the object's location. AR Core's unit for its translations are in meters[1].

[1] "Translation units are meters." https://developers.google.com/ar/reference/java/com/google/ar/core/Pose

However, an extra step must be taken to transform the object's AR-world coordinates into the offset of how far in meters the object is from the device in terms of true north and true east. This requires use of the device's magnetometer.

The world space is the model of the world as sensed by the AR device, centered around the AR device. Each AR anchor has a Pose which provides the functions qw(), qx(), qy(), qz() that return the object's rotation, and tx(), ty(), tz() that return the object's translation in world space. In a quaternion (x,y,z,w), x,y,z are the 3D coordinates (y is the vertical/up axis, x and z are the horizontal axes), and w is the scale factor. [2]

Given tx() and tz(), we have the offset of the object from the AR device. These offsets are aligned along the AR engine's x- and z-axis, so we must transform them into offsets aligned to true north and true east.

Blockers

Lack of Documentation

There is a distinct lack of documentation related to combining AR Core and Location Services (ie GPS). This issue has been brought up before on the AR Core github and on Stack Overflow, to which the answer has been that AR is simply too new to have this documentation ready (even though this was asked six years ago, in 2012). The current workaround is to use manual solutions.

Limitations of Object Files

We lacked example GIS object files to reference, since they existed in multiple formats and we were missing an understanding of what data they contained.

Objects are constructed with an object file (.obj) that contains a list of 3-dimensional vectors to construct a 3-dimensional model from a 2-dimensional image, and a image file (.png) that provides a texture as well as the basis for the object file to construct with. The data from the object file is being mapped to the image.

We had two initial ideas for constructing pipes. It could be that each pipe is defined in its entirety in one object file, or that each pipe is broken down into a series of reusable objects that need to be constructed individually and reassembled before they are placed.

In the case that each pipe is only one object, we won't need to deal with testing the placement of many different types of objects (each pipe would be). If each subsection of a pipe is even just

---

[2] https://gis.stackexchange.com/questions/2951/algorithm-for-offsetting-a-latitude-longitude-by-some-amount-of-meters

slightly different from each other, and there are far too many pipes and subsections of pipes, then the accurate connection of individual pieces of a pipe could also prove to be difficult.

In the end, we decided it would be better to use GeoJSON to represent pipes instead of separate object files.

There is also the current limitation of AR being unable to provide stable support for a large number of objects (above 20 in our demo), whether they are copies of the same model or entirely different models.

ArcGIS' Incompatibility with AR

Seeing that ArcGIS only performs the information-based part of what we want to do. We want to provide an overlay of the ArcGIS map on top of the physical world, but it isn't possible since ArcGIS's 3D presentation is through a "Scene" or "Map" that must be created first, and which takes up the whole phone's view space. To make the camera and "Scene" work together, the two will need to be manually integrated. Furthermore, the provided "SceneView" is closer to using a map than it is to AR. Figuring out a way to integrate the two libraries has been difficult in the aspect of bringing global coordinate presentation of the ArcGIS into the ARCore structure.

Examples of Our Learning

1) Placing two different objects

Video Link: https://www.youtube.com/watch?v=ciax9_fnzPw
https://www.youtube.com/watch?v=okMnMluLWwo

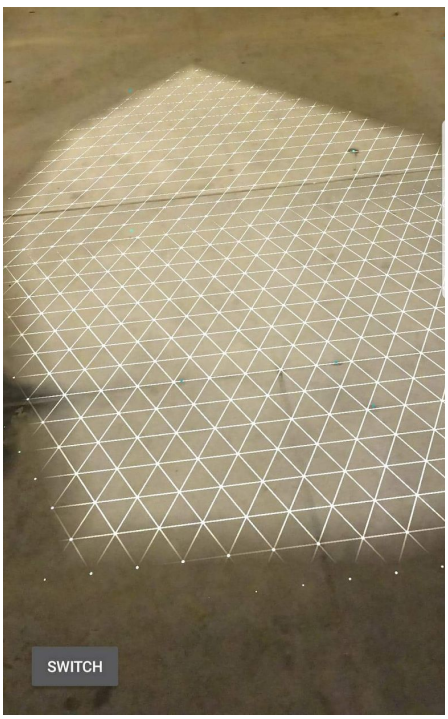
For this demo, we created a button that toggles between 2 objects (android icon and the pipe). When clicked, it changes the 3D image that is placed in the plane upon tapping on the screen. Using this method, we are not limited to only one object, can easily expand it to more than two objects.

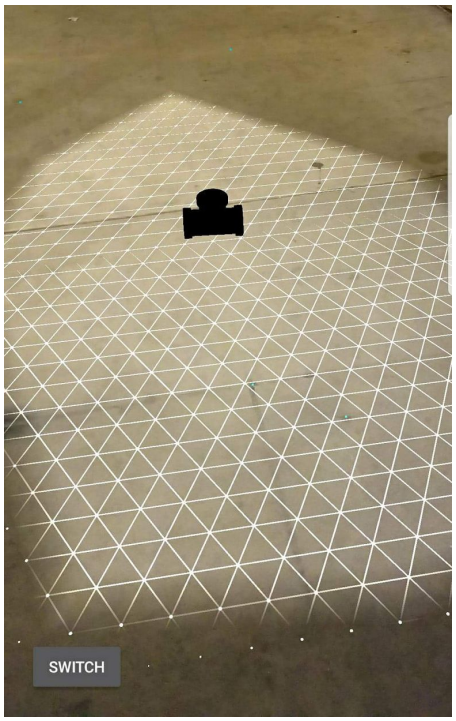
Previously, we were only able to place the default Android object. Our next step was to allow the placement of multiple objects. We first found an object file for the pipe, but the task was not as simple as changing the object file's name when creating an object. We had to instantiate each type of object separately, and kept track of each type in a list indexed by "mode," or a string to describe the object. We referenced the object by its "mode" whenever we had to update its position.

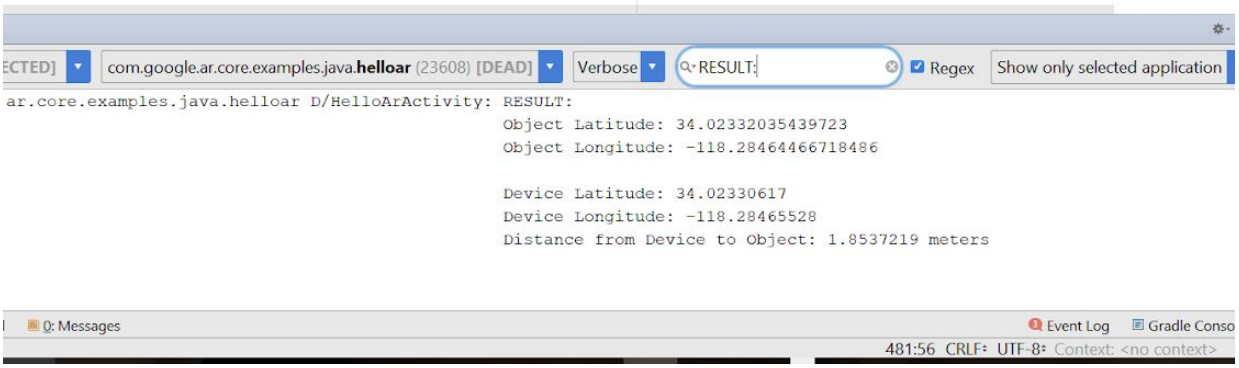
2) Showing a) longitude and latitude of the device
b) longitude and latitude of the object placed
c) distance between the device and the object

This demo shows the example of calculating the longitude and the latitude of an object placed inside the detected plane. This could be useful when we try to connect our planes with the real world. We first found the longitude and latitude of the device itself.

As mentioned before, we used the translation values of our object to calculate the longitude and latitude with the equation we researched online (cite this). To check the accuracy, we found the longitude and the latitude of the location that we placed our object and compared it with our results. Also, we used our current location and the object location to determine the distance between them. To expand on this, we may need to figure out the altitude of an object and its status over time.







Project Summary (Sprint 4)

Sprint 4 (Apr 16)

1. Determine Anchors for coordinates objects or planes (instead of using hits (taps on the screen) to determine location)
2. Calibration of the AR distances and projections with the real world.
    a. Calibrating the AR simulated space to real compass directions (North, East, South, West).
3. Incorporating a 3rd and 4th dimension (time and altitude) to the object's 2D latitude/longitude data
    a. 3rd dimension represents altitude/elevation, since we don't want to just remain at sea level
    b. 4th dimension represents time interval
4. Mock-up data using the web app geojson.io

Summary

Currently, the team is able to place objects given its latitude and longitude relative to the phone's orientation and coordinates. This part of the project relates to displaying objects based on information read from GeoJson or ShapeFiles. The goal is to provide a consistent display of features and objects placed from files.

During the development of this functionality, we discovered inconsistencies in the locations of objects that were being placed. These objects would appear at different places, leading us to realize that the calculations we were doing only placed these objects relative to the phone's camera coordinate system by ARCore, rather than relative to compass north and east. As a result, because the phone's orientation is not aligned by default with the direction of (true or magnetic) north, objects will not be placed accurately at their respective longitudes and latitudes.

Notes

We don't know much about this problem, but the device we're testing with seems to restart sometimes/stop working when we run our ARCore application. It may be due to a system failure/crash that involves a cache buildup or resource exhaust. Although we can resume testing for a while after the device restarts, it is worth mentioning to highlight issues with efficiency and integrity, and could be further looked into if it becomes a hindrance to testing. At one point, the phone did stop responding so the application may be affecting the system's firmware, which could be a problem.

What's Working Well

Placing Objects Without Tapping

AR Core for Android's AR system consists of a session, which contains the camera and any Trackable objects. Trackable objects are "anchored" to a physical location by Anchors, and these Anchors' locations are described by their Poses. Trackable objects are always in one of three TrackingStates: TRACKING (on screen and with its position up-to-date), STOPPED (for a deleted Anchor), or PAUSED (the position is not being updated currently, but could be updated at a later time).

During our initial trials, we tried initializing the object in two ways: once during each frame and once in the initialization function. The latter doesn't work since the session must exist before the object can be initialized, and the session isn't guaranteed to exist while AR Core itself is being initialized.

Placing Objects From File

Reading from a GeoJSON file in the application, and using the GeoJSON parser to parse the file, we can place objects predefined in the file. Due to the limitations of our own AR app, we are currently limited to placing points.

Calculating Altitude

Altitude is estimated by distance from a WGS84 reference ellipsoid, which doesn't line up with the distance above sea level.[3] The difference must be looked up or calculated by hand to produce the accurate distance above sea level.

GPS hardware will report its information in ASCII strings according to the NMEA format. This format specifies a "Global Positioning System Fix Data" to fix the discrepancy between measured and actual altitude, called a GPGGA sentence. This GPGGA sentence will contain time, latitude, longitude, altitude, and geoid separation among other values measured by the GPS. Adding the altitude to the geoid separation part of the GPGGA sentence will return the accurate altitude relative to sea level.

There are many factors to calculate altitude, which can make it difficult to retrieve the most accurate value:[4]

[3] "GPS returns the altitude as an offset from the WGS84 reference ellipsoid" https://stackoverflow.com/a/12811478

[4] Factors used in calculating accurate altitude https://gis.stackexchange.com/questions/234430/inaccurate-altitude-gps-data-with-my-android

1. The altitude value returned from ARCore's getAltitude() function is a calculation of the altitude above the WG284 reference ellipsoid, and may not represent the true *topographic height* (this may be the measurement that the stakeholders are looking for), which is the altitude of the device/object above the current surface, relative to the whole topography of the earth.
2. Assuming the satellite(s) used to calculate location and altitude maintains the same distance from the device/object, when the satellite moves to a different altitude, the distance between the two will not change as much the altitude of the satellite, which will cause for greater errors and inaccuracies in calculating the altitude for the device. This process is something known as GDOP (Geometric Dilution of Precision)

Supposedly, the true altitude (topographic height) can be calculated using location (latitude/longitude), the elevation or ellipsoid height, and the geoid height.

- The elevation or ellipsoid height is assumed to be the measurement that the mobile device returns based on the GPS receiver
- The geoid height can be calculated with an online calculator that uses latitude and longitude values
- The equation to calculate the topographic height is as follows:
  Topographic height = ellipsoidal height - geoid height

Blockers

Reorienting Placed Objects

Currently, objects are placed according to the phone's orientation during the frame the object is to be placed. To keep placement consistent with the global coordinate system, we can rotate each object as it is placed so that it will be aligned to the global coordinate system, rather than to the phone's internal coordinate system, which is dependent on the phone's orientation.

Objects can be rotated to face a different direction by manipulating its matrix. Objects' positions can be rotated (ie calibrated to true north) by manipulating the rotation of their Anchors' Poses.

A rotation is represented by a quaternion in the form

(sin(theta/2)*x, sin(theta/2)*y, sin(theta/2)*z, cos(theta/2))

If we want to rotate about the y-axis (the "up" vector in OpenGL), then we transform the pose by the quaternion **(0, sin(theta/2), 0, cos(theta/2))**.

Calibrating the device to true north can be done by calculating the azimuth from magnetic north and then converting that into the bearing from true north. However, the azimuth displays "-Infinity" since the sensorEventListener was never called.

We also had issues with the object moving on its own after it had been placed due to GPS inaccuracies, and had problems with using sensors to determine the angle of the rotation to rotate the object by since the measurement of true north was also inaccurate. So we settled on an alternative design where the user would manually calibrate the device to true north, so that any coordinates would line up within the AR simulation.

Examples of Our Learnings

* See Screenshots in Learning #8

References

Android

- Android Developer. Save Files on Device Storage. https://developer.android.com/training/data-storage/files.html

Manipulating Objects

- FreeCAD. Working with Transformation Matrices and Placement Objects. https://forum.freecadweb.org/viewtopic.php?t=2324
- Github. Google AR, arcore-android-sdk. Issue #103 How to draw object using camera position in ar-core? https://github.com/google-ar/arcore-android-sdk/issues/103
- OpenGL. Transformations. https://open.gl/transformations
- Stack Overflow. Detecting if an tap event with ARCore hits an already added 3d object. https://stackoverflow.com/a/46808473
- Stack Overflow. How to check ray intersection with object in ARCore https://stackoverflow.com/questions/46017301/how-to-check-ray-intersection-with-object-in-arcore/46028084#46028084

Calculating Coordinates and Altitude

- Android Developers. “Converting GPS Data” https://developer.android.com/things/sdk/drivers/gps.html#convert-gps
- GitHub. Google AR, arcore-android-sdk. Issue #119: Feature Request: Provide a geo-oriented world coordinate space.
- https://github.com/google-ar/arcore-android-sdk/issues/119
- GPS. NMEA sentence information - GPGGA Message http://aprs.gids.nl/nmea/#gga
- Stack Overflow. Android: How to get Accurate Altitude? https://stackoverflow.com/questions/9361870/android-how-to-get-accurate-altitude
- Stack Overflow. ARCore+Unity3D: How to make a scene oriented to north? https://stackoverflow.com/a/47167558

- Unavco. “Tutorial: The Geoid and Receiver Measurements” http://www.unavco.org/education/resources/tutorials-and-handouts/tutorials/geoid-gps-receivers.html
- Unavco. “Geoid Height Calculator” http://www.unavco.org/software/geodetic-utilities/geoid-height-calculator/geoid-height-calculator.html

Rotating Objects

- Android Developers. Location.bearingTo(Location dest) https://developer.android.com/reference/android/location/Location.html#bearingTo(android.location.Location)
- Android Developers. Position Sensor https://developer.android.com/guide/topics/sensors/sensors_position.html
- Android Developers. Sensor Event Listener https://developer.android.com/reference/android/hardware/SensorEventListener.html
- Android Developers. Sensors Overview https://developer.android.com/guide/topics/sensors/sensors_overview.html
- Google Developers, AR Core. Pose.makeRotation() https://developers.google.com/ar/reference/java/com/google/ar/core/Pose.html#makeRotation(float,%20float,%20float,%20float)
- Stack Overflow. Use orientation sensor to point towards a specific location https://stackoverflow.com/questions/5479753/using-orientation-sensor-to-point-towards-a-specific-location/5518318#5518318
- Stack Overflow. Android: get device orientation from Azimuth, roll & pitch https://stackoverflow.com/questions/25972519/android-get-device-orientation-from-azimuth-roll-pitch
- Wikipedia. Quaternions and spatial rotation https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Learning #1
Augmented Reality Platforms

February 5, 2018

Possible Data Sets

1. **Pipelines**: Coordinates and location data, material of the pipe, what it carries, its name/ID, managed by which company/department/person (if that'd be relevant or useful), past issues & where did they happen, last time inspected?, size of pipe (circumference, diameter, thickness)?
2. **Field Instruments**: Data that would be useful when overlaid on the instrument itself - instrument readings, metadata, owner, model type, blueprints of the instrument, date last inspected, etc.

iOS (ARKit)

Notes/Findings

- Developed in Xcode
- **ARHitTestResult (Class)**
    - Information about a real-world surface found by examining a point in the device camera view of an AR session.
    - https://developer.apple.com/documentation/arkit/arhittestresult

Thoughts/Opinions

- So, I was thinking about the whole displaying relevant data depending on the current location. ARKit has a class called ARHitTestResult that provides info about the real-world surface/topology through the camera, which has several useful data members:
    - var distance: CGFloat
        - The distance, in meters, from the camera to the detected surface.
    - var worldTransform: matrix_float4x4
        - The position and orientation of the hit test result relative to the world coordinate system.
    - var localTransform: matrix_float4x4
        - The position and orientation of the hit test result relative to the nearest anchor or feature point.
- Especially the worldTransform one, which assumes that the current location is based on the world coordinate system, probably GPS. One idea I had from this is that when the PG&E data is finally involved, we could have an attribute along with the data in the same table or something, that holds some sort of a range of coordinate values. This could provide some ideas on how to automatically pull-up data, if in that exact coordinate, the device in question is there as expected. *Needs to be explored more.*
- I feel that actually displaying the data is not that difficult, rather the method as to which the data needs to be pulled up automatically is the tricky part. There needs to be some algorithm/basis/criteria for this.

ARKit is meant for catering towards gaming and user interfaces for retail services

Google (AR Core)

- https://developers.google.com/ar/
- Java, Unity, Unreal, C, Web
- They provide APIs for all except Web
- Web
    - Web has a portion where they refer you to three.ar.js which extends the three.js 3D library
        - https://github.com/google-ar/three.ar.js?files=1 three.ar.js github page
        - https://developers.google.com/ar/develop/web/getting-started getting started for AR Core for Web
        - https://github.com/google-ar/three.ar.js/blob/master/API.md three.ar.js API
        - https://developer.mozilla.org/en-US/docs/Web/API/VRDisplay three.ar.js builds on top of VRDisplay
    - Web version can be tested with special browsers provided by Google (install them on your phone):
        - WebARonARCore (Android)
        - WebARonARKit (iOS)
    - Seems very light, doesn't provide for object recognition and whatnot.
- Java
    - Needs Android Studio and an Android device
    - Uses OpenGL to display graphics
    - Can only acquire up to 8 objects/point clouds.
    - Lighting estimation
- C
    - Requires Android Studio
- Examples
    - https://experiments.withgoogle.com/ar

Google's ARCore is more tailored towards informational and accessibility assisting softwares.

Limitations of AR

- Exact location of a user/where the user is pointing
    - Current method (GPS, compass, internet tapping?) is just an **estimate**, unreliable for structures (AR will also become an estimate -> bad for engineering which needs exact measurements)

    - Could only assume user is standing still & user can only turn around. Then it becomes more like a 2D image. (you can't move to look around the image you're pointing at)
    - There's a **solution**. $30 - $200.
- Spatial problem: pipe doesn't look like it's under ground
    - Superimpose an autocad over the object?

Amazon (AR View)

- Within the Amazon iOS app?
- Not a development platform, just an example of what AR looks like??
- Amazon tries to create a virtual reality environment and does not focus on working towards using live development and creation of an augmented view.

Amazon caters toward corporate and global companies and their attempts to eliminate logistical overhead.

PWA (progressive web apps) and AR

- Not a platform for AR apps but a method of improving user experience (specifically, reducing load times by caching)
- https://developers.google.com/web/showcase/2017/twitter
- https://medium.com/arjs/why-web-apps-are-the-future-of-augmented-reality-c503e796a0c5

Interesting Relevant Links / Studies

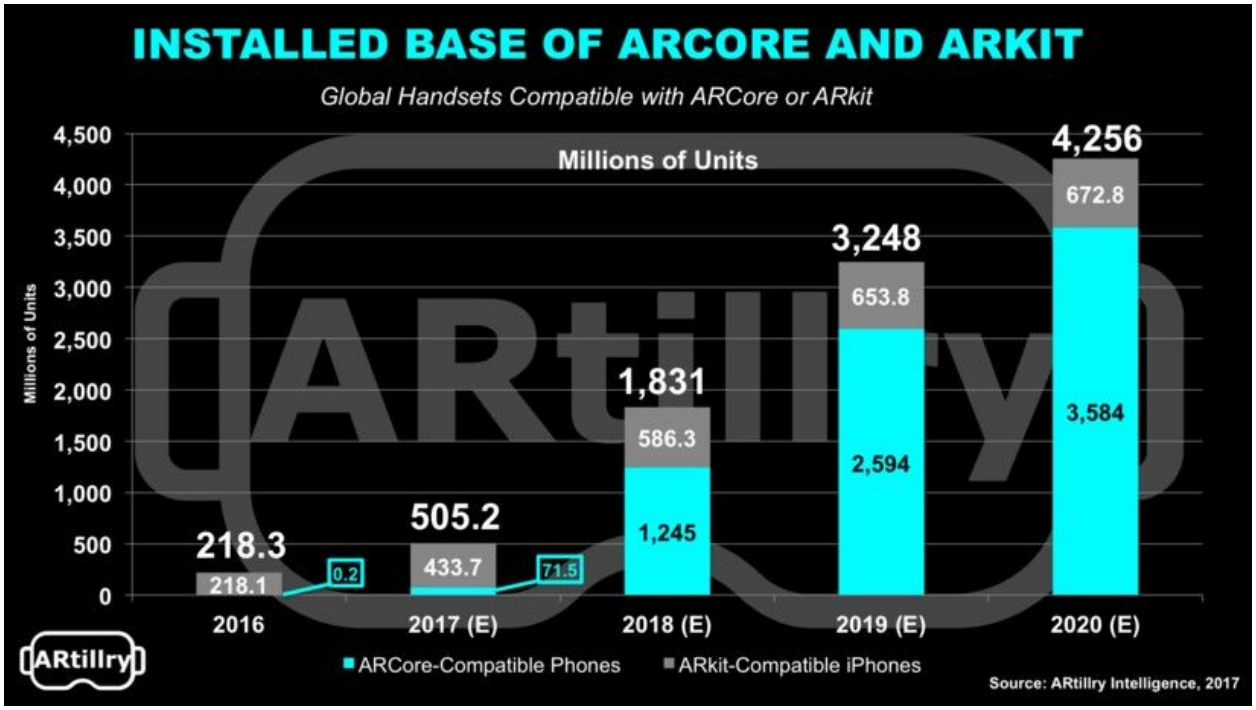
Annotating Real-World Objects

- So this has been done before but it requires a lot of preparation and devices that work simultaneously to provide accurate, detailed information
- Also not completely relevant, since we're doing mobile, but it's still cool to know that it has been done before in other platforms. Hopefully the possibility exists for us!!!
- Uses the concept of superimposing a virtual world onto the environment, which may be too much for our project/not what we need

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.7105&rep=rep1&type=pdf

Comparison Between ARKit and ARCore

https://jasoren.com/capabilities-and-limitations-of-apple-arkit-and-google-arcore/



- A very aggressive projection of user base and accessibility of AR.

- **ARKit**
  - **Advantages**
    - For lighting, they use color temperature and intensity
    - Apple has an obvious benefit when it comes to the operating system fragmentation since this company has a unified product line. Regardless the total number of iOS devices lower than Google-based ones, many Apple devices run recent iOS versions
  - **Disadvantages**
    - Less tracking capabilities than ARCore, probably not as large maps
    - Only compatible with IOS 11 or higher
- **ARCore**
  - **Advantages**
    - For lighting, they use a shader through Unity API or pixel intensity through Android Studio
    - SLAM (Simultaneous localization and mapping)
    - Tracker checks if a map is available
    - Maintains larger maps that makes it more reliable. Therefore, if user's device loses tracking, ARCore will better recover a map.
  - **Disadvantages**
    - the Android device market is significantly fragmented into numerous devices which rarely use the latest operating system which can lead to some **compatibility** issues.
    - Runs only on Android Nougat 7.0 or higher

Wikitude

- Android , IOS, Smart Glasses
- Very good tracking system

Maxst

- Android IOS Windows, MacOS
- Smooth Rendering.

Alternative Ideas

Overlaying blueprints

- For wires or pipes, overlay their locations on a map (similar to the way highways are displayed on a map)

GPS on the blueprint

- Being able to set that well.

Required Technologies (Approaches to AR, with alternatives)

AR Development Environment (currently leaning towards Google AR Core, but Apple AR Kit is also one option)

- Auth0 for login functionality
- Possibly look into GIS technologies: ESRI, ArcGIS
    - ESRI *Revealing Hidden Information With Augmented Reality and GIS* http://www.esri.com/esri-news/arcuser/fall-2015/revealing-hidden-information-with-augmented-reality-and-gis
    - ArcGIS *Get Started With ArcGIS Pipeline Referencing* http://pro.arcgis.com/en/pro-app/help/production/location-referencing-pipelines/get-started-with-arcgis-pipeline-referencing.htm

Other Research

- https://www.upwork.com/hiring/for-clients/building-augmented-reality-mobile-apps/

Project: Bentley

- https://communities.bentley.com/other/old_site_member_blogs/bentley_employees/b/stephanecotes_blog/posts/augmented-reality-for-infrastructure-a-first-step
- https://communities.bentley.com/other/old_site_member_blogs/bentley_employees/b/stephanecotes_blog/posts/augmentation-of-subsurface-utilities-the-problem-of-spatial-perception

Talks about GIS and AR

- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.3513&rep=rep1&type=pdf

Current tech for mapping underground utilities

- https://www.spar3d.com/blogs/guest-blog/underground-utility-detection-tech-vision-future/

Chicago (Alternative to AR)

- https://www.youtube.com/watch?v=HgI_JbEIAbE
- https://www.cnet.com/news/what-it-takes-to-become-a-smart-city-chicago-louisville-cincinnati-san-diego-rahm-emanuel/

Learning #2
Geographic Information Systems

February 5, 2018

What is GIS?

Geographic information system

- Captures, stores, organizes **geospatial** information
- Allows one to create interactive **queries**
- Can related unrelated information using **location** as a shared key
- Can show many types of information on a simple map (in **layers**)

Used to:

- Optimize services
- Inform decisions
- Improve communication
- Improve record-keeping
- Create **3D** images/maps
- Aid scientific investigations, resource management, development planning

Specific use cases:

- LA county dept of regional planning
    - "Viewing and investigating zoning, land use policy, subdivision activity, aerial imagery, and many other features **pertaining to land use** within the unincorporated communities of Los Angeles County. These applications also allow the user to **manipulate the maps** in various ways and **save** map extents to Portable Data Format (**PDF**), which can then be printed."
- Time-based GIS technology (National Geographic)
    - "One important use of time-based GIS technology involves creating **time-lapse** photography that shows **processes** occurring over large areas and long periods of time. For example, data showing the movement of fluid in ocean or air currents help scientists better understand how moisture and heat energy move around the globe."
- **National Pipeline Mapping System**
    - A fully-functional Geographic Information System (GIS)
    - Contains location + selected attributes of:
        - hazardous liquid and gas transmission pipelines
        - liquefied natural gas (LNG) plants
        - breakout tank farms
        - including those pipelines that are offshore
    - Public-access pipeline operator contact information

Sources

- https://en.wikipedia.org/wiki/Geographic_information_system
- http://www.esri.com/what-is-gis
- http://planning.lacounty.gov/gis
- https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/
- *National Pipeline Mapping System: Standards for Pipeline, Liquefied Natural Gas and Breakout Tank Farm Operator Submissions (October 2017)* https://www.npms.phmsa.dot.gov/Documents/Operator_Standards.pdf
    - NPMS guidelines, data submission guidelines, data format + verification + updates, attribute data rules, geospatial data rules, metadata
- *California State Fire Marshal Pipeline Mapping System Operator Submission Standards* http://osfm.fire.ca.gov/pipeline/pdf/mapping/spms_standards.pdf
    - Acronyms, naming conventions, pipeline feature class data, start & end point feature data, Event Attribute Tables for Age, Category, Commodity and Diameter, Examples Scenarios & Attribute Information.
- https://en.wikipedia.org/wiki/Shapefile **SHAPEFILES**
    - https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf Environmental Systems Research Institute (ESRI) (big company providing GIS tech?) **ESRI Shapefile Technical Description**
    - http://shapelib.maptools.org/ (C shapefile library)
    - https://stackoverflow.com/questions/2139335/which-c-library-for-esri-shapefiles-to-choose (C++ shapefile library)
    - https://pypi.python.org/pypi/pyshp (python shapefile library)

Learning #3
Text Recognition Through A Phone

February 19, 2018

Description

The Text Recognition AR app is a simple android app that utilizes the TextRecognizer class and the device's camera to recognize and capture any letters and numbers that are visible, and stores them in an array. Our approach was based on using the letters and numbers as an identifier, for target assets/devices.

Technological Requirements

- Android Studio (IDE)
- Any android device to test mobile app exported from Android Studio
- Text Recognizer from Google Vision API

Possible Use Cases

Field workers may use this app to point to a model number / identifier of an asset and be able to pull up relevant sensory data. By using a precise location of the device, we can also limit the numbers of devices we have to search for in the database, in order to retrieve the correct info.

Pros

- Fairly easy to implement in theory (and through our actual development of this prototype).
- Can work on any android device (version/brand independent)
- Does not require workers to manually insert the identifier, in situations that may require use of both hands

Cons

1) Captures every single letter that the camera recognizes and currently, we as developers have no ability to limit that real-time functionality.

   Possible Workarounds:

   Option 1: Have the user point to the identifier for some amount of seconds and if the data is consistent for those few seconds, submit it to the database for querying.

   Problem: Use has to make sure the phone stays still.

Option 2: Display what user captures and have user press a button when the user can confirm the screen displays the correct identifier.

Problem: Requires extra steps/work for the user that defeats the purpose of trying to simplify the information retrieval and display process.

2) Might not be useful if the asset does not have a unique identifier or if we cannot get close enough to the asset, for safety reasons.

Learnings

We realized that not many assets in the field have identifiers or that it's difficult to get close to the asset. It might be better for us to use the GIS data and location service to recognize the asset itself rather than pointing to the device. However, we could speed up our queries by recognizing some of the texts that help us to identify the asset.

Relevant Link(s)

Referenced Documentation for the TextRecognizer Class under Google API's for Android
https://developers.google.com/android/reference/com/google/android/gms/vision/text/TextRecognizer

Learning #4
Google AR Core For Web

February 19, 2018

Requirements

**See [1] for reference used for Requirements*

- **Three.ar.js** [1] (Javascript library)
    - Requires **Three.js** (since it extends three.js)
- Prototype browsers (one or the other)
    - **AR Core on Android** [2]
        - **Android device** (non-emulator) (ONLY works on Pixel, Pixel XL, Pixel 2, Pixel 2 XL, Samsung Galaxy XL)
        - **ARCore APK**
        - **WebARonARCore APK**
            - Can be built from source on **Chromium** [3] (see below for Chromium's requirements)
        - **WebViews** for debugging (opening **chrome://inspect** in your browser while the Android device is connected via USB)
    - **ARKit on iOS** [5]
        - **Xcode 9**
        - **iOS 11** (Recommended devices: iPad (2017), iPad Pro (9.7, 10.5 or 12.9 inches), iPhone 7 and 7 Plus)
        - **Apple Developer account** (sign up at http://developer.apple.com/)
- Chromium: [3]
    - Requires **git** and **python**
    - A 64-bit Intel machine running **Linux** with at least 8GB of RAM. More than 16GB is highly recommended.
    - At least 100GB of free disk space.
    - Uses Ninja [6] to build itself

It is possible that the lesser amount of development in these web-based libraries makes this option undesirable to continue on. However, because these libraries are web-based, they provide a framework that will work in the future on any browser that can support AR.

Notes

Github project link: https://github.com/whl827/ARTeam/blob/master/ArCoreWeb

- ARCore's package provides examples and boilerplate code in the included /examples directory
    - Of note is this example showing how to read from markers (i.e. QR codes)
    - And this one describing a reticle.
    - This may be interesting as well, describes recognizing surfaces and placing cubes on those detected surfaces

Known issues [2]

- The current implementation of WebAR is built on top of the Chromium WebView flavor. This has some implementation advantages but some performance and use disadvantages. We are working on making the implementation on a full version of Chromium.
- Pausing/resuming/switching away from the app causes screen to turn black. This is a consequence of having built the implementation on top of the WebView flavor of Chromium. A proper implementation on full Chromium or a rebase to a more recent Chromium WebView version (>57.0.2987.5) might solve this problem.
- The Web Speech API is a standard web API for text to speech and speech to text conversion that is available in Chromium. As WebARonARCore is built on top of the WebView version of Chromium, does not provide this functionality by default. There is a solution though, using a polyfill we provide, but in order to use it, you need to either a) include the three.ar.js library before making any use of the Web Speech API or b) include the ARSpeechRecognition.js file also before making any reference to the Web Speech API. Only speech recognition is suspported, not speech synthesis for now.

- How to integrate this with a database? (ie using REST?)
- How to integrate this with GIS?
    - ESRI Labs provides this blog post about their product AuGeo. On the project's github page, it claims to have had plans to release its source code in July 2017, but it's already Feb 2018 and there's no source code...
    - ArcGIS API with Javascript (example 3.x to 4.6 migration guide)
- Example markup (ARML) [6]

```
<gml:Point gml:id="ferrisWheelViennaPoint">
 <gml:pos>
  48.216622 16.395901
 </gml:pos>
</gml:Point>
```

Links

[1] https://github.com/google-ar/three.ar.js#threearjs

- **Three.ar.js github** page
- Includes examples in the /examples directory

[2] https://github.com/google-ar/WebARonARCore

- **WebARonARCore github**
- Contains instructions on how to set up on an **Android device** (can't be an emulator: must be Pixel, Pixel XL, Pixel 2, Pixel 2 XL, or Samsung Galaxy XL)
- Also has instructions [3] on how to set up on a **Linux virtual machine** with **Chromium** ("from source")
- Lists afterwards an alternate set of instructions (shorter, it seems, but accomplishes the same things)

[3] https://chromium.googlesource.com/chromium/src/+/master/docs/android_build_instructions.md

- Is linked to from [2], which also details an alternate set of build instructions
- **Instructions for setting up Chromium** on the Linux VM from the Chromium repo

[4] https://developers.google.com/ar/develop/web/getting-started

- **Getting Started with AR on the Web | AR Core | Google Developers**
- Includes links to examples and other builds for AR Core (i.e. Unity)

[5] https://github.com/google-ar/WebARonARKit

- **WebARonARKit** github
- How to set up on an **Apple device** (requires Xcode 9 and iOS 11, as well as an Apple Developer account)

[6] https://ninja-build.org/

- Website of Ninja, the tool that Chromium [3] uses to generate builds

[7] https://en.wikiversity.org/wiki/Augmented_Reality/Resources

- Augmented Reality/Resources (a collection of links)
- Lists ARML (augmented reality markup language) as an issue to be resolved… [XML + EcmaScript]

[*] https://link.springer.com/article/10.1007/s13369-015-2006-1

- "Building Construction Information System Using GIS" in the *Arabian Journal for Science and Engineering* (Oct 2016), Vol 41, Issue 10, pp 3827 - 3840

[*] https://www.wikitude.com/arml-20-standard-approved/

- ARML 2.0 is approved
- “...main AR Browser vendors, Layar, Metaio and Wikitude, teamed up to make the browsers' interoperable utilizing **ARML** 2.0 as the common interchange format between them “
    - not sure what this means for us but I’ll put it here anyway

[*] https://books.google.com/books?id=gRpH49rlVRsC&pg=SA14-PA10&lpg=SA14-PA10&dq=ARML+support&source=bl&ots=YboGQOHybx&sig=zbCLX27sVXEbTOYAYhjqxNVpcPU&hl=en&sa=X&ved=0ahUKEwj_h_nvmKfZAhVX3mMKHQ4gCIIQ6AEIUDAF#v=onepage&q=ARML%20support&f=false

- Professional Augmented Reality Browsers for Smartphones: Programming for junaio, Layar and Wikitude (May 2011)
- The (linked above) appendix has a list of ARML parameters for wikitude, junaio; might have been truncated due to limited preview

Learning #5
ARCore Development with Android

February 19, 2018

ARCoreGIS development

Beginnings of Mobile Application Development

Within ARCore of Google, there are three different engines that can be used to develop the application on an Android phone, Android Studio, Unity, and Unreal. There are some pros to using each specific engine which we will provide a short excerpt about below.

1. Android Studio
    a. Has extensive libraries and capabilities to help with the ease of its use in developing mobile applications for Android devices.
    b. Compatibility with outside APIs and with APIs that work directly with Android are both well developed and used in the community of Android development.
2. Unity
    a. Was created to develop games and allows for the use of multiple game development libraries (Unreal included) that help with 3D space development.
    b. Because there is no specific mobile platform that this platform caters to, it is easy to develop 3D mobile applications on one mobile platform and migrate/ develop for another.
3. Unreal
    a. This is an open source development platform which allows for deep programming in C++. The source files are available for modification locally, giving freedom towards development in all areas.
    b. Multiple threads are used to run the game engine, providing for greater functionality and fluidity for applications and games that are developed and run on. (Android Studio included, Unreal is on a single thread)

Developing in Android Studio

Starting with the first option, Android Studio. For this first part of development, a basic app that we created helped provide insight towards the how the ARCore development was being done, along with reading documentation. After further research into the library and platform that ARCore provides, it has proved to be a durable and flexible environment to help create the programs that we are pursuing.

ARCore provides an environment in which points are declared and placed in a PointCloud where they are all stored. These points are attached to the world that is discovered through the optical lens that interfaces with and the other functionalities that are required to produce quality points. For the purposes of the product, what can be enhanced here could include having the ability for points to also store information about real world longitude, latitude, with height and distances based of an anchor point that is created. Using these points and further discoveries

with ESRI's API, this platform is proving to be a solid environment to use to develop the product and its functionality of visualizing pipelines and electricity lines.

Further Discoveries

As we researched the abilities and limitations to development in AR, we discovered that AR for mobile phones are only limited to specific Apple and Android phones. For Apple, phones from Iphone 6 and further are available with their ARkit versus Google's ARcore with only Samsung Galaxy 7s and Google Pixel phones along with their following developed products. The reason being for this is because of the other sensor and devices that they needed to include to help make AR possible for these mobile devices. Knowing this, we learned that if any phone that is not supported or is not being developed with these sensors, a thorough process of calibration and addition of software and hardware would be needed to provide quality AR development and usage.

Learnings

Through this, what we are discovering here is that though the accessibility of AR to phones are limited now, the potential for it to grow is arriving with great speed. Development in AR is something that can and should be pursued. Despite the limitations of how many users will be reached, the information of the need for software calibration of cameras and other sensors on a device may be crucial for the rest of the development of visualizing assets.

Link to Project: https://github.com/whl827/ARTeam/tree/master/ArCoreGIS

Learning #6
AR Core 1.0 with Android Studio

March 5, 2018

Details of ARCore 1.0's Release

The ARCore 1.0 SDK has been released. AR apps can now be downloaded and installed from the Google Play store. Android Studio Beta now includes an AR-compatible emulator.

Google is also looking into working with multiple phone manufacturers to further increase the amount of devices that can be compatible with AR. [1]

The current release focuses on ARCore's builds for Android [Studio], Unity, and Unreal. In September 2017, Google teased upgrades to their AR-enabled prototype browsers (WebARonARKit and WebARonARCore) [2] which were provided January 22, 2018. [3]

Setup Instructions

(Referenced from [5])

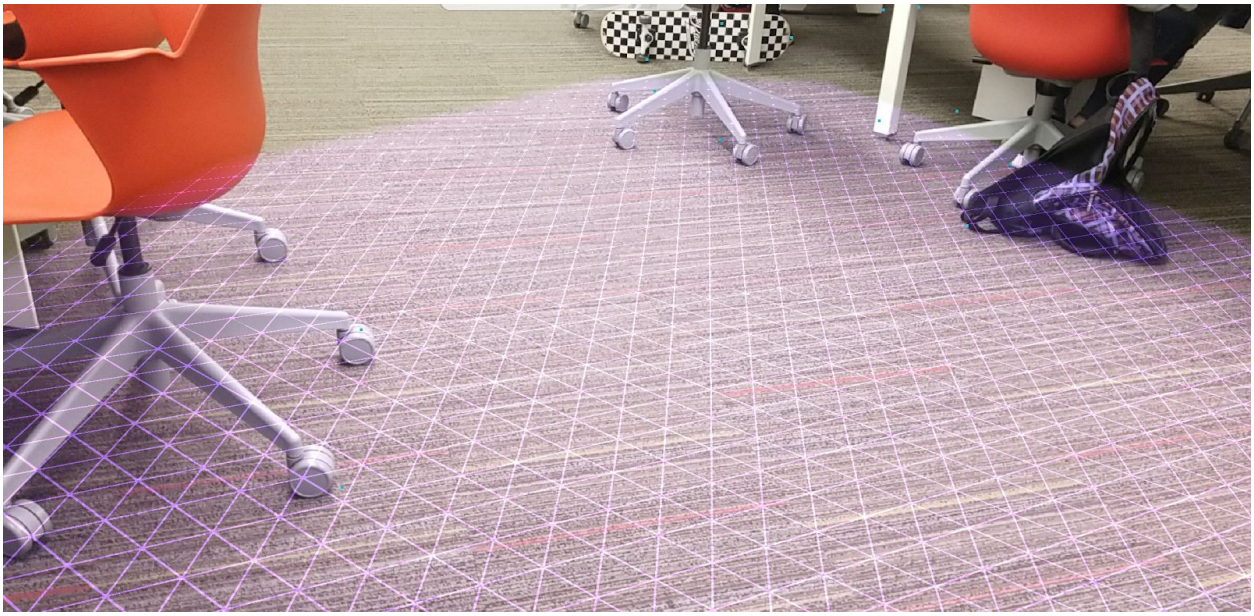
1. Install Android Studio 31.0 (aka Android Studio Beta, Android Studio Preview). It can be downloaded, installed, and used alongside a non-beta Android Studio (details provided here).
2. Download the ARCore 1.0 SDK.
3. Set up an AR emulator in Android Studio 3.1.0.
4. Install ARCore 1.0 onto the emulator.

Source Link

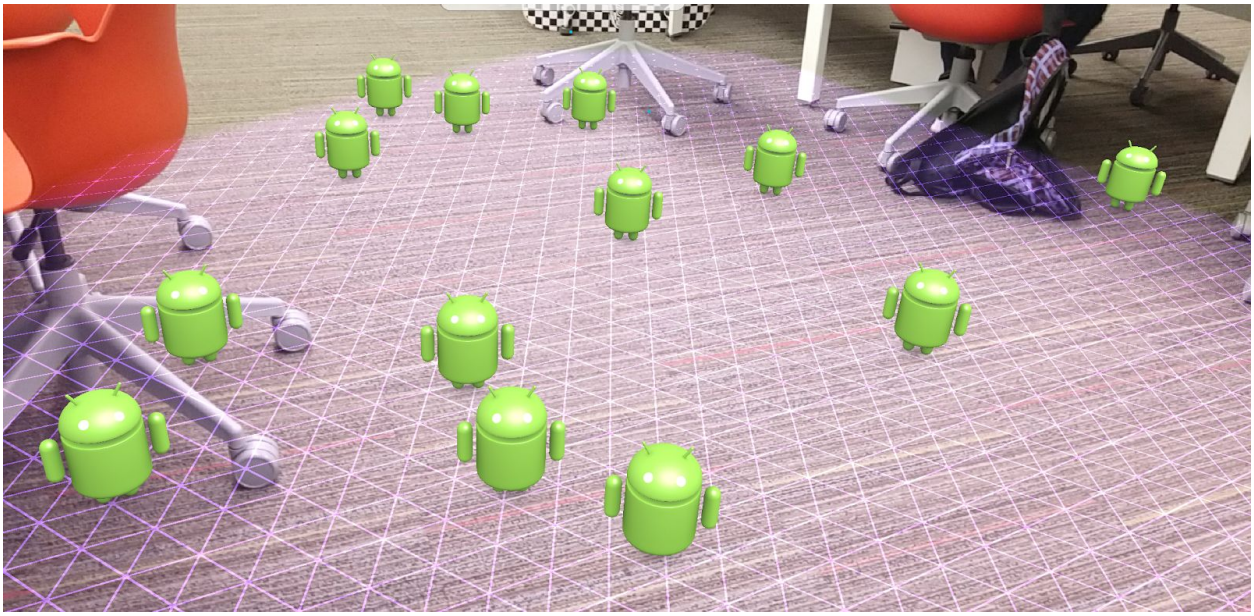
https://github.com/whl827/ARTeam/tree/master

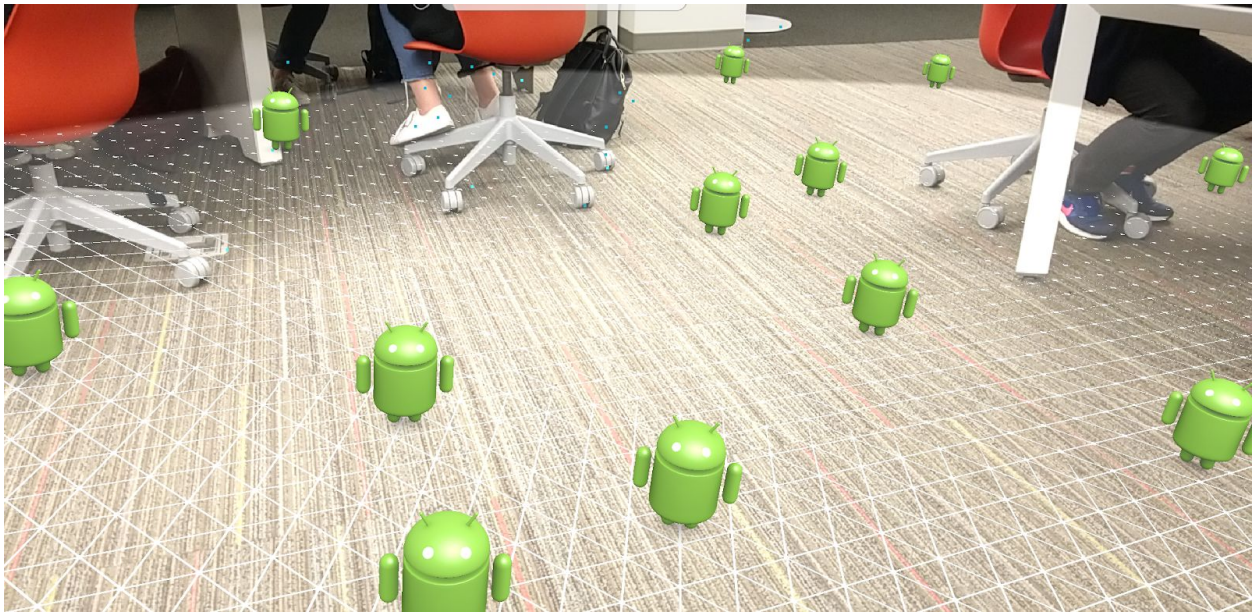
Screenshots

Plane Detection/Mapping of Floor

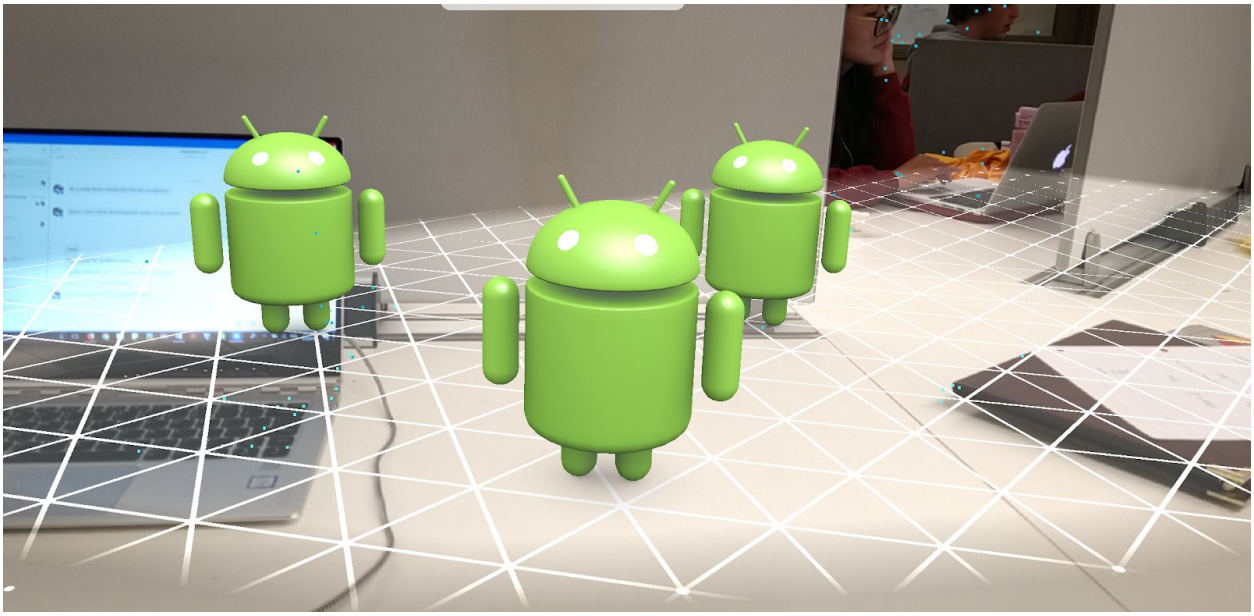


Plane Detection/Mapping with Object(s) Placement of Floor

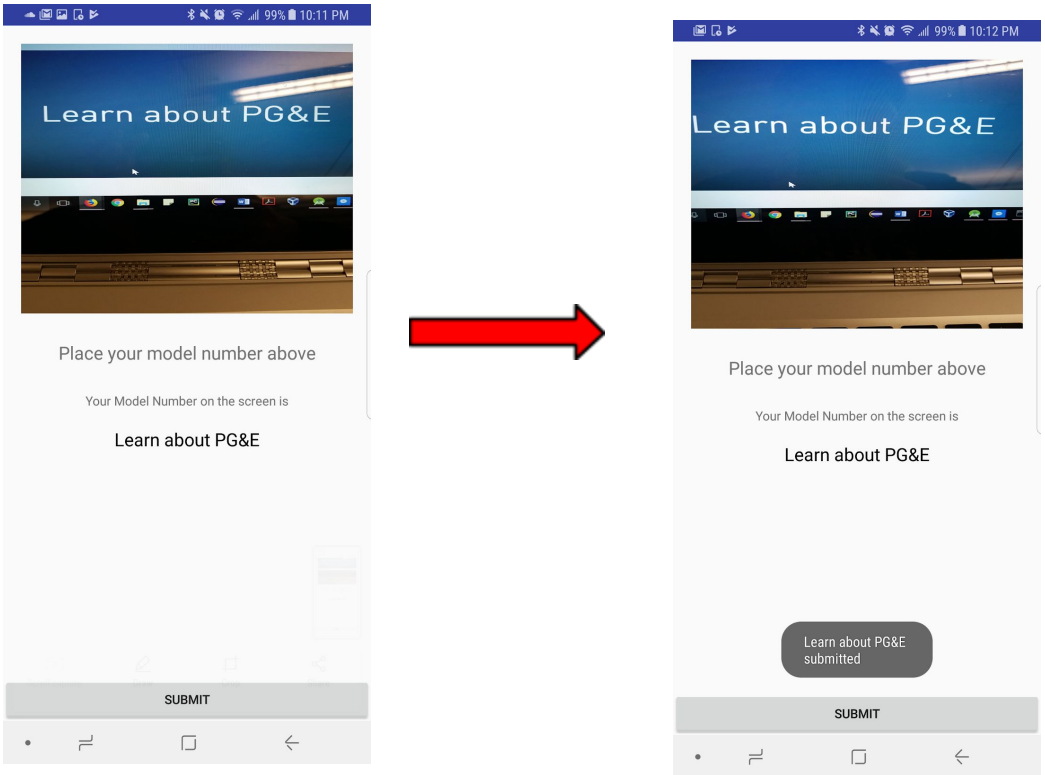




Plane Detection/Mapping with Object(s) Placement of Elevated Surface (Desk)



Text Recognition (with Button for Requests)



References

[1] A. Gosalia. *Announcing Google ARCore 1.0 and New Updates to Google Lens,* Google Blog. Feb 23, 2018. https://www.blog.google/products/google-vr/announcing-arcore-10-and-new-updates-google-lens/

[2] C. Kelley. *Daydream Labs: Experiments with ARCore*, Google Blog. Sep 11, 2017. https://www.blog.google/products/google-vr/daydream-labs-experiments-arcore/

[3] A. Ali. & J. Carpenter. *Augmented reality on the web, for everyone,* Google Blog. Jan 22, 2018. https://www.blog.google/products/google-vr/augmented-reality-web-everyone/

[4] ARCore. *AR Core for Android Studio Quickstart* https://developers.google.com/ar/develop/java/quickstart

[5] ARCore. *Run AR Apps on the Emulator* https://developers.google.com/ar/develop/java/emulator

Learning #7
Neo4J Graph Attributes

April 2, 2018

What it is

Neo4J is an open-source graph database (as opposed to a relational database). It uses a graph to store data, and data can only exist as an edge, a node, or an attribute. Graph databases can perform lookup much faster than relational databases can. It uses the language Cypher (instead of SQL).

Nodes can have multiple labels, multiple attributes (properties), and multiple relationships with other nodes. Labels group nodes into sets, and are indexed to speed up lookup. Relationships are directional and can be recursive; they can be traversed in either direction. Properties are named with strings, can be indexed (alone or with other properties) and constrained (ie forced to be a string or not null).

Properties can be Strings, integer, float, or boolean, or a list of those four. “Null is modeled by the absence of a property key.” They are modelled by property graph models, and consist of a key-value pair.

Thoughts

USC offers a class on databases, which covers RDBMS (specifically MySQL). Three of the four team members have taken or are taking this class, so we will be using RDBMS in combination with our prior knowledge of graphs as a reference point for understanding graph databases.

Compared to a relational database, the structure of Neo4J, which groups individual nodes by amorphous labels rather than in rigidly defined tables, seems freer. In a graph database, each data point is its own object, rather than a row in a table, or several rows across multiple tables.

How does this impact our design? What considerations are there to make?

Labels seem to not be affected by which properties are defined for a node, or what properties a node is lacking. In that case, it could be possible to have an object labelled a “pipe” that lacks important “pipe” properties with other objects labelled “pipe” (e.g. lacks a diameter). Since a “null” value is represented by an absence of the property, the program cannot assume that certain properties exist for certain nodes, and it will always interpret the absence of an property as “null.” This means that a node’s properties are not strictly defined -- the program could very logically ask a sewer pipe for its voltage.

In extension, could it be possible that properties in general are not well defined? Could asking for the value of a node’s property while making a typo in the name of the property return without error despite being technically invalid, since it is interpreted as a valid property with a null value?

References

- https://neo4j.com/
- https://neo4j.com/docs/developer-manual/current/introduction/graphdb-concepts/
- https://github.com/opencypher/openCypher/blob/master/docs/property-graph-model.adoc

Learning #8
Shapefiles and GeoJSON Files

April 16, 2018

ESRI ArcGIS Shapefiles

ArcGIS interfaces with a map that *they* create first, and then you place GIS files into this map that they've already created. This means we cannot use ArcGIS' map with AR as provided, and instead must read the GIS files manually to use its information.

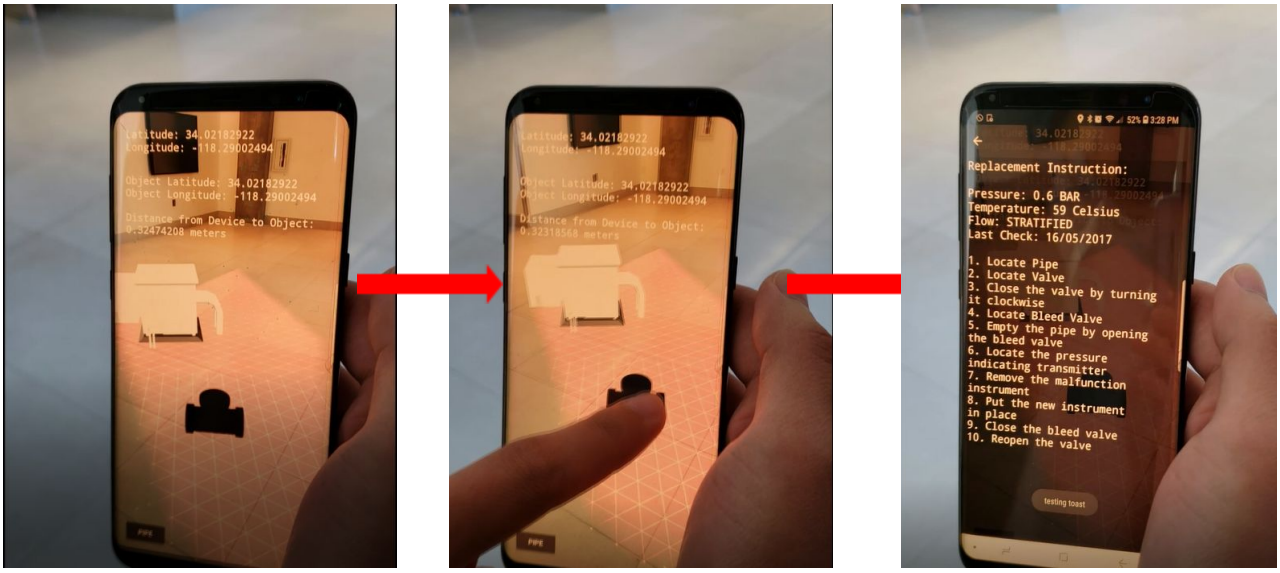
A shapefile is a hierarchy of files: an entire ZIP folder with a header and smaller files inside, which themselves may include other files in them, and each of which contain different information. EIA.gov provides several shapefiles.

GeoJSON

The example given by Pete Baker was that GeoJSON has feature classes e.g. point, line, polygon that define the shape, and then has a very flexible set of user-defined attributes that describe what it is, like "steel pipe", "4-inch diameter", "installed in 1918."

GeoJSON is an open source extension of the JSON format that is widely supported and very flexible.

Tap the Object to Display information



Whenever the device is tapped, it creates a hit point which contains x, y, and z values of the object's position. These values are compared with the x, y, and z values of each object currently being tracked by the device. If these values are similar enough within a certain margin of error, we open a new activity to show details about the specific object being tapped. If no object matches the tapped location, then we create a new object at that location instead.

Implications of Learnings

Purpose

The following outlines the implications of the use of Augmented Reality in the field of utilities and also outlines our concluding thoughts about its possible local and global impacts. Also included is a reflection on the technical details noticed during the development of our prototype.

Current State of Technology

Augmented reality (AR) is an up-and-coming tech that can be used to obtain, display, and interact with information related to the real world. It has become a simple and intuitive way for users to visualize and interact with data. AR development has only begun to receive its due respect and investment only recently. As a result, the technology is still only sparsely available and even more so, barely accesses the tip of its great potential.

Currently, the AR industry suffers from fragmentation of solutions, each providing different features for different platforms. Google's AR Core alone supports Android, Unity, Unreal, and through a set of custom web browsers, the Web, each one with a slightly different API and vastly different capabilities. Furthermore, the industry is so young that few solutions provide support for common features such as obtaining coordinates of a virtually placed object, finding the orientation of the camera, or streamlining information reticles next to an object.

The answer to the latter is to pick a more powerful platform, to wait until a more flexible platform catches up, or invest in furthering a platform. In terms of platforms, we suggest looking into and tracking the development of web-based AR solutions like Google AR Core for Web, and AR browsers like Firefox Reality. The reason is that web browsers are independent of the devices they run on, and AR solutions targeted towards the Web will allow programmers to leverage many of the already-existing libraries and frameworks for the web along with reaching a wider user base.

Local Impact

At PG&E, the AR can have a profound impact, if it is incorporated as part of the daily routine of its technicians on the field. The most important roles that it would play includes providing greater safety through situational awareness and cost effectiveness through reducing costs of managing information retrieval.

How does AR make this possible? A full fledged AR application would help provide information about specific electrical or gas line assets that its technicians work with on a daily basis. Technicians on site would find it difficult to quickly assess the situation of every asset, due to the

rapid advancement in technology and the vast amount of technologies present, that span a few decades. To combat this, AR would be available through a mobile device and its visual aid of streamlining information about assets that technicians require.

The hope is for the application to assist in providing a risk analysis or quick diagnosis of situations or devices in need of maintenance or to provide a live feed of existing pipelines or electricity lines that run through the ground.

Technicians would normally operate at a much longer duration manually, maybe hours or days, to achieve the same amount of precaution that AR would provide them in just a look through the lens of the camera of their mobile devices.

Global Impact

To begin thinking on a national level of the impact of AR in this field is daunting, even more so on a global scale. Globally, the amount of pipelines and electricity lines that run around the developing countries of the world is numerous. The amount of resources that this world transports around is overwhelming. With the help of AR, the world can become much safer through providing a greater awareness in situations that they might find themselves in, providing a better understand of the technology used around them. Unfortunate accidents involving people interacting with dangerous and risky assets like transformers or live electric lines could all be avoided through the use of AR and the information that it could provide.

Information could be revolutionized into ways the future has always been envisioned it to be. Data can be seen everywhere and the interaction between people could be altered in both positive and negative ways.

But some of the risky implications that would come with utilizing AR in this field could be the risk of the security of information and the slight inaccuracies that may come with interacting with data that may not be currently at its most accurate.

Development of AR in this field could bring into view, a greater and more innovative way of interacting with data in general.

Reflection

The biggest blocker to the development of the project ended up being the platform that we chose to work on. AR Core on Android is extremely limited compared to ARKit or even AR Core on Unity, which ease the placement of objects by location, provide reticles to display pop-up information in, and provide ways to calibrate the device or correct for the device's orientation. Furthermore, there wasn't much documentation, open source support, or projects for ARCore, and the ones that did exist were difficult to understand.

We had a slow start partially due to the fragmentation of platforms available, each of which provided different capabilities and were compatible with different devices. We ended up weighing which platform to use without knowing which capabilities we would need.

Our own inexperience with the both AR and the workings of technicians and GIS also slowed us down, making it difficult for us to pin down a feasible MVP. Researching AR was relatively straightforward; it was comparatively more difficult to research field work before we received pointers about where to go or what to look for.

We made a lot of progress on the project and were able to turn out a working demonstration of augmented reality. Our app is able to place points according to coordinates and display information for each point. However, shortcomings include only a half-implemented calibration, since we ran out of time to implement it manually, as AR Core for Android lacks support for this functionality. A general suggestion towards future developers and advancement in this technology is to pursue development on iPhones because of the great capabilities and calibration that Apple has done with their phones and API of ARKit. Pursuing this would be a better option than any other platform for mobile devices currently because of ESRI's development collaboration of their ArcGIS and ARKit which has proven itself capable of AR.

Overall, we learned a lot from this collaborative project. It pushed the limits of our ability to engage in new technologies such as augmented reality and tools such as ArcGIS, ESRI, GeoJSON, OpenGL. From working directly with vectors and matrices, we were able to expand on our understanding of various topics of advanced math and learn about computer graphics as well.

The team has definitely seen the potential of what AR can achieve for an organization, and we plan to continue to look into AR during our respective careers. There will be endless opportunities and possibilities for AR, which can only evolve from here on out.

We thoroughly enjoyed working with Pacific Gas & Electric in this challenging and rewarding venture. We would like to personally thank Matt Rettagliata, Pete Baker, John Nichols, the field workers, and Professor Miller for allowing us this opportunity, and we hope that this project will continue to develop further, in a way that contributes to PG&E's EPIC 3-37.

Links

Final Project Github Link

https://github.com/whl827/ARTeam

Final Project Video Link

https://www.youtube.com/watch?v=pVtT7PD4RWE&feature=youtu.be