

3D Data Visualizer: Using the Microsoft HoloLens 2

Team sdmay22-21: Ami Ikanovic, Ben Kelly, Elizabeth Nelson, Isaac Littler, Parth Padmanabhan, Scott Fank, Zahydee Machado

Advisor/Client: Joseph Zambreno



The Team

- Team Lead: Elizabeth Nelson
- Scheduling Lead: Isaac Littler
- Client Communication Lead: Ami Ikanovic
- Team Communication Lead: Scott Fank
- Facilities Manager: Ben Kelly
- Documentation Lead: Parth Padmanabhan
- Delivery Manager: Zahydee Machado



Project Vision

Our project vision is to develop a real-time map data AR visualization of meteorological and other data that takes as input various publicly accessible data streams and produces as user configurable map overlays. This will be created for the Microsoft HoloLens 2 using the Unity development platform.



Use-Cases

- Primarily designed as a showcase piece for the Department of Electrical and Computer Engineering at Iowa State
 - Highlight the university's commitment to research and innovation
 - Communicate the opportunities and successes of ISU to prospective students and visitors
- Can be used in the education sector to learn about the global data and patterns visualized
 - Could provide a greater understanding of problems faced around the world



Visual Sketch





Requirements

Functional

- Globe: Scaling / Rotate / Move / Stationary
- Data: Real time / Historical / 3 data streams / Multi-layer visualization
- Use: “Pick up and go”

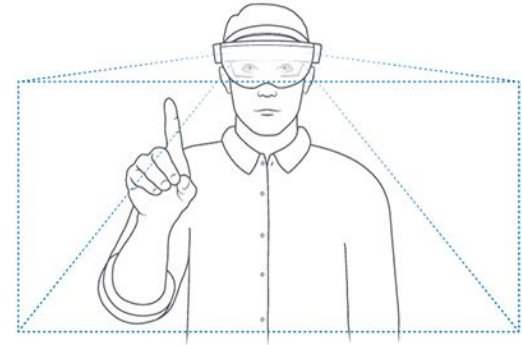
Non - Functional

- 60 FPS / Hz
- 1080p per eye
- 1 Active hour of use per charge

Requirements

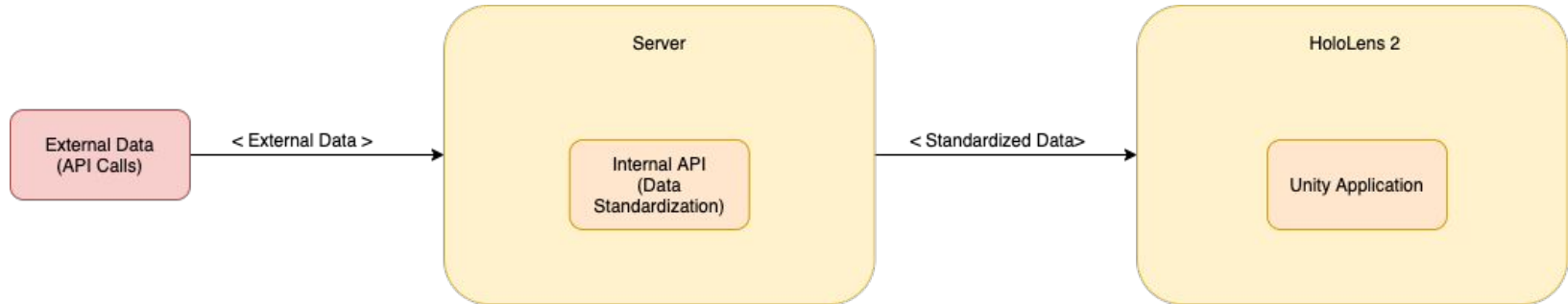
Constraints

- 6' x 6' unobstructed space
- \$1000 budget
- HoloLens 2 hardware constraints



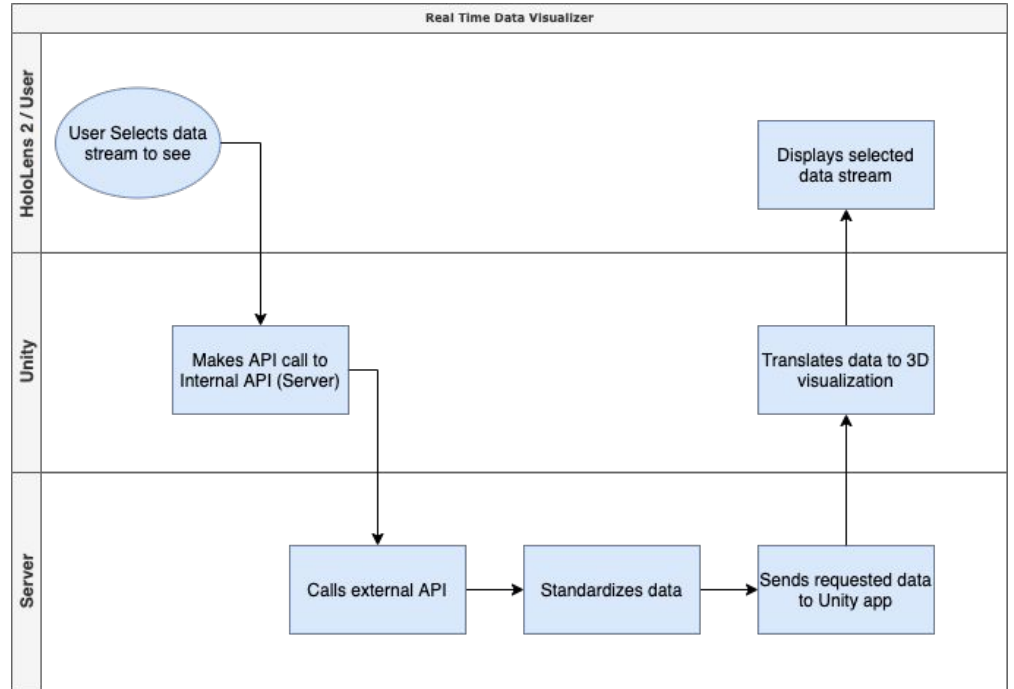


Design Diagram



System Design

Swimlane Diagram



Unity: Make API call to internal server

```
var geoNodes // Nodes list {long, lat, [weatherdata]}

function populateNodes() {
  requestDataFromInternalServer(geoNodes)
}
```



Server: External API call -> Standardize -> Send data to Unity

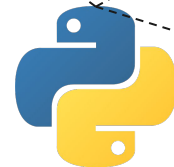
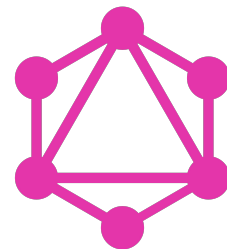
```
function requestDataFromAPI(geoNodes) {
  for each node in geoNodes {
    var data = request(node) // Returns json
    // api.openweathermap.org/data/2.5/weather?...
    // lat={node.lat}&lon={node.lon}&appid={API key}
    standardize(data) // Standardize json to our specification
    appendToJsonFile(data) // json file with all data
  };
  return jsonfile;
}
```

Unity: Translate data to 3D visualization

```
for each coordinate on earth {
  if (coordinate data exists) {
    visualizeData();
  } else if (not exist) {
    interpolation() // process of filling
                  // in missing data using
                  // surrounding coordinates
    visualizeData()
  }
}
```

Async processes:

- Unity checking what coordinates are in FOV
- Server loading data in background





Prototype Implementation


Unity app implementation:





Design Complexity

- Design Complications:
 - Manipulating real-time data
 - Mixed Reality Toolkit functionality
 - Using external APIs
- Iterations:
 - Planning:
 - Selecting the platform to use between AR, VR or a spherical display.
 - Settling on AR using the Hololens.
 - Analysis and Design:
 - Using a globe projection to plot geographic data, such as the weather.
 - Using free weather APIs to project weather data on to the AR globe.

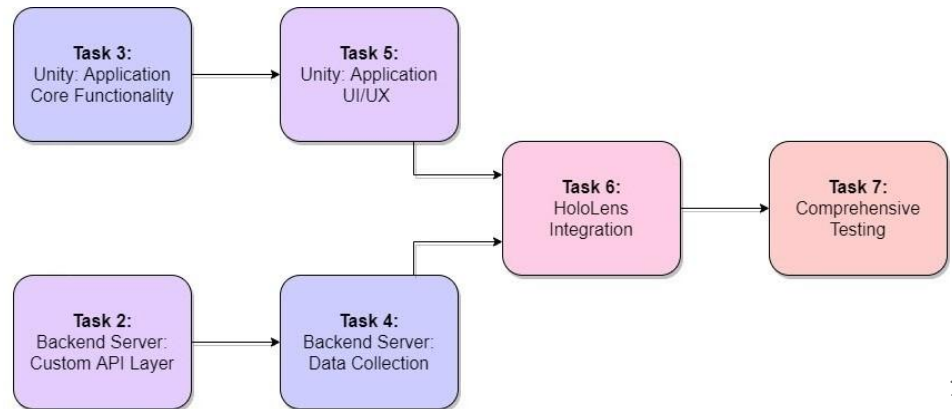


Project Plan - Milestones

- **Backend Server: Custom API Layer**
 - Deploy a server for collecting and standardizing all incoming data into a single API format that will be sent to the HoloLens
- **Backend Server: Data collection**
 - Configure the server to call and receive realtime and historical data from one or more APIs
- **Unity: Application Core Functionality**
 - Complete a Unity application with the required globe elements, navigational controls, and data receiving functionality
- **Unity: Application UX/UI (UI, modeling, etc.)**
 - Implement data visualization and user interface elements into the Unity application
- **Hololens Integration**
 - Deploy the unity program on the HoloLens with working hand gesture functionality
- **Comprehensive Testing**
 - Achieve consistently acceptable performance and confirm “pick up and go” user experience

Project Plan - Schedule

- **Week 1-2:** Backend Server: Custom API Layer
- **Week 1-6:** Unity: Application Core Functionality
- **Week 3-6:** Backend Server: Data Collection
- **Week 7-9:** Unity: Application UX/UI
- **Week 10-11:** HoloLens Integration
- **Week 12:** Comprehensive Testing





Test Plan

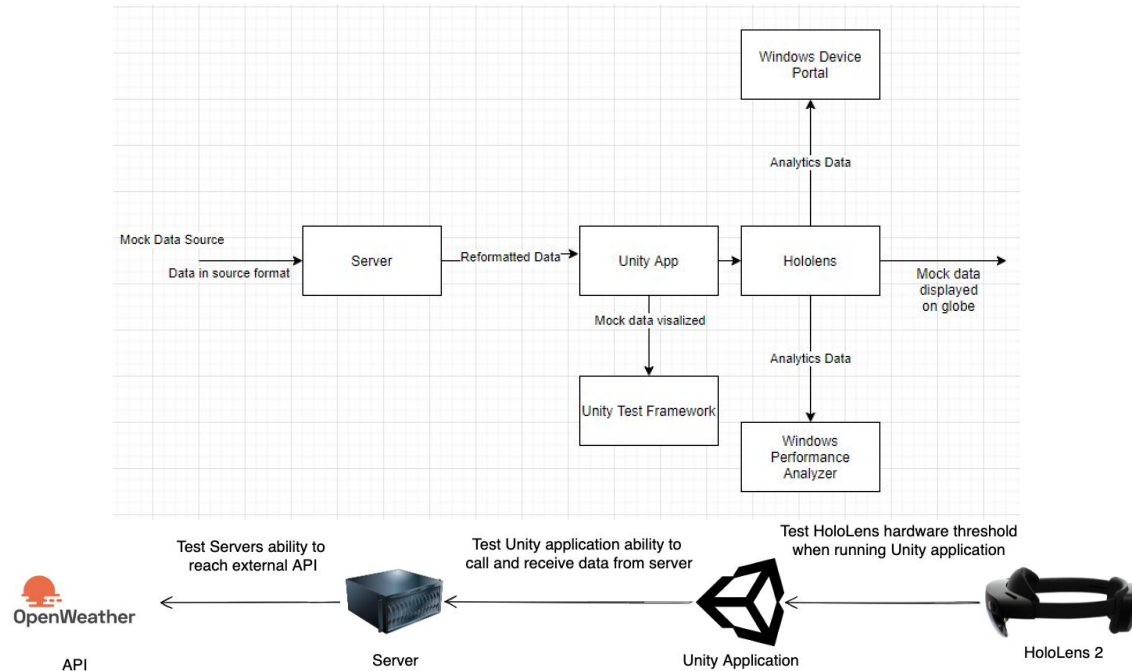
- Tests are performed on individual components (Hololens and server) before testing the software as a whole
- Unit tests are performed on the components:
 - Unity Application (Unity Test Framework):
 - Globe Manipulation(rescaling, rotating)
 - Data Overlays (Generation, Layering, Toggable)
 - Data Retrieval (Frequency)
 - Data Server (Postman, Pytest):
 - Data Retrieval(External API requests, Query building)
 - Data Processing(Manipulating external data to fit the format for the Unity application)



Test Plan: Interface Testing

- External API Interface
 - Unit tests to validate data
 - Use of Postman to make mock requests compatible with external API
- Internal API Interface
 - Unit testing API operations
 - Using Postman to make mock requests to check responses to data
- Unity API
 - Unit tests for data processing classes
 - Postman to simulate requests to ensure compatibility with the server

Test Plan: System Testing





Conclusion

- We are finishing the design phase and heading into implementation, with some progress already done in prototyping
- Next semester will begin with our implementation of the Unity Core Application, the Backend Server and API layer



3D Data Visualizer: Using the Microsoft HoloLens 2

Questions?