VR4GETAR Project Report

INSTRUCTOR & CLIENT: DR. EDWARD FOX
COURSE: CS 4624 (MULTIMEDIA, HYPERTEXT AND INFORMATION ACCESS)

Jishnu Renugopal, Christian Richardson, Will Schmidt, Mattin Zargarpur, Haiyu Zhao, Kevin Zhang
VIRGINIA TECH, BLACKSBURG, VA 24061

Table of Contents

Table of Figures

Table of Tables

Executive Summary

Global Event and Trend Archive Research (GETAR) is supported by NSF (IIS-1619028 and 1619371) through 2019. It will devise interactive, integrated, digital library/archive systems coupled with linked and expert-curated webpage/tweet collections. This project will act as a supplement to GETAR by providing a Virtual Reality (VR) interface to visualize geospatial data and extrapolate meaningful information from it. It will primarily focus on visualizing tweets and images obtained from the GETAR data archive on a globe in a VR world. This will be accomplished using tools like Unity, HTC Vive, C# and Git. In order to ensure that the product meets the end user's specification, this project will use an iterative workflow with a very short feedback loop. The feedback obtained from Dr. Fox and our team members will be used to make subsequent prototypes and the final product.

Our project is intended to be used as a demo by school children interested in data analytics and data sciences. Additionally, this project can also be extended to add features to our end product.

Our final product can display images and tweets on a globe in a VR world provided that they have location information. As part of our final deliverable, we delivered a report, a presentation, a video demo and a GitHub repository containing the source code for our project. During this project, our team learned that building a 3D application is very different from building a 2D desktop application. We also learned that it is crucial to meticulously document all parts of product development to assist future development.

1.0 Overview

The focus of this VR demo is a globe and how it can be used to present information. The globe can be rotated by hand and the user is free to walk around it. On the surface of the globe are dots which represent points of data from the GETAR database. Each dot is positioned on the globe according to the location of the data that it represents. The user can filter the data present on the globe by time and topic using a menu. When the user taps a dot, it expands to display the image and text data from its position. The user can then collapse the dot, hiding its data, or drag individual pieces of data off the globe. When pieces of data from any number of dots have been dragged off the globe, they can be grouped by the user to create a custom collection of data points for data exploration. The user can name the collection, add tags to it, and save it for later viewing.

In order to make full use of the VR platform, this project will avoid the use of screens or buttons that resemble traditional 2D interfaces. Instead, the user will interact with virtual 3D objects to consume data or apply changes. When a dot is expanded, its data is displayed on prisms which the user can grab and move with their hands to create groupings. The user will interact with 3D objects representative of data points in the VR world to create and modify collections. Additionally, we also learned that

1.1 Revision History

Table 1 Revision History

Date	Description	Author(s)
02/17/17	Requirements	Jishnu Renugopal
02/18/17	Design	Mattin Zargarpur
03/12/17	Implementation	Jishnu Renugopal
04/09/17	Prototyping	Jishnu Renugopal
04/10/17	Corrections	Mattin Zargarpur, Jishnu Renugopal
04/11/17	Testing, Data explorer’s manual	Mattin Zargarpur
04/27/17	Final Report	Jishnu Renugopal

1.2 Team Roles

Table 2 Team Roles

Name	Role
Dr. Edward Fox	Client, Instructor
Christian Richardson	VR Developer
Kevin Zhang	VR Developer
Haiyu Zhao	Data Scientist
Mattin Zargarpur	Design & Developer Team lead
Jishnu Renugopal	Documentation & Team lead
Willem Schmidt	Developer & Tester

1.3 Section summary

Users' Manual:

This section contains information that a user of this application might find useful. It consists of a step-by-step walkthrough to help the user use the source code to build an application for a VR device. Additionally, it consists of a brief description of the use cases that were implemented.

Developer's Manual:

This section contains information that a developer trying to extend this application might find useful. It presents the class structure of the application, tools used and a detailed explanation of the testing techniques used.

Prototypes, Timeline and Lessons Learned:

This section presents the management aspect of this project. It contains description of the roles that each team member played. Additionally, it contains a breakdown of the events that led to the final product. Finally, it also presents an overview of the prototypes that were used to refine the final product.

Future Work:

This section consists of suggestions that a future developer could use to extend this application.

2.0 Users' Manual

This section describes the use cases that were implemented, the process of setting up the VR4GETAR project on a new machine, adding data to the project, and running the application.

The use cases outline the requirements that have a direct effect on the users' experience. The special requirements summarize the aspects of the application that could have an indirect effect on the users' experience.

The following are the use cases that were implemented as a part of this application.

2.1 Use case 1 – VR Interactions with the Globe

An interaction pattern is a high-level interaction concept that can be used repeatedly across different applications to achieve common user goals.[1] Our research revealed that Hand Selection Pattern is the most natural way for a user to interact in a VR environment.[2] Therefore, to allow intuitive interactions with the model, we plan on using the HTC Vive controller to pan and select objects. Additionally, we plan on using the users physical position combined with redirected walking (walk in a larger virtual space than available physical space) to facilitate interactions with the globe.

2.2 Use case 2 – Aggregation of Data Points

Due to the sheer volume of our data, we plan on using data science approaches such as classification and clustering to extrapolate meaningful information from the datasets provided. Visualizing too much data has many cons such as clutter, performance, information loss and limited cognition.[3]

To avoid this, we will use data aggregation – a process in which information is gathered and expressed in a summary form, for the purpose of analysis. By trying to condense/cluster data into meaningful clusters, we can avoid clutter and limited cognition. This will be done on the basis of the datapoint's location.

2.3 Use case 3 – Ability to Create and Modify Collections

To allow the user to personalize their VR environment, it is necessary that we provide a means to create custom collections of tweets and pictures. To achieve this, we will use VR interfaces to allow the user to save pictures and tweets to a collection. Additionally, we would also allow the user to modify collections by deleting or moving tweets and pictures around. Figure 1 shows how collections will be displayed in the VR world.

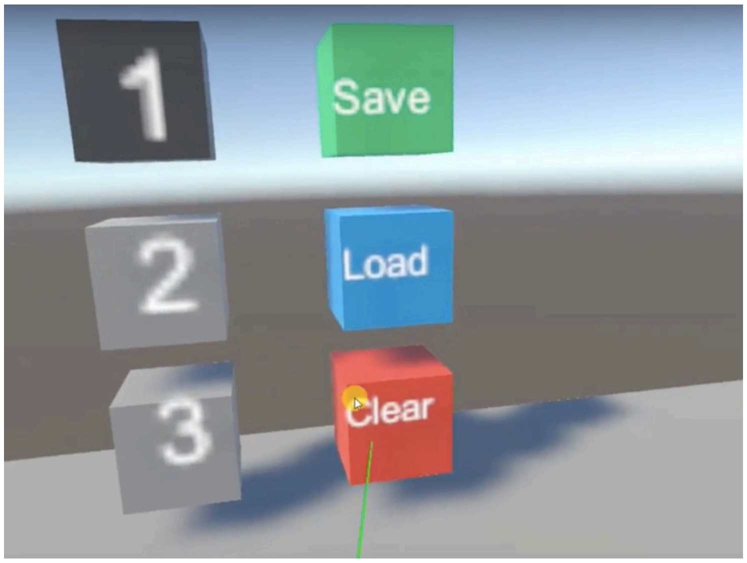


Figure 1 Collections

2.4 Use case 4 – Persistence

After our meeting with Dr. Fox, we learned that the ability to save the current state of the VR model will be crucial for its success. The need for this feature is only solidified by our feature that enables users to filter and discover data. Therefore, we will provide a mechanism to save the current state of the system and the collections. This will make the process of data exploration easier among multiple users.

2.5 Use case 5 – Color Coded Data Points

During one of the meetings, Dr. Fox suggested that we use color coded points to differentiate between tweets and images. Therefore, we will use red colored points to represent data points that contain tweets and yellow colored points to represent images. Additionally, points that have been selected will be displayed in yellow regardless of the type of data. These features will give the user additional information without having to open individual data points. Figure 2 shows how points will be color coded.

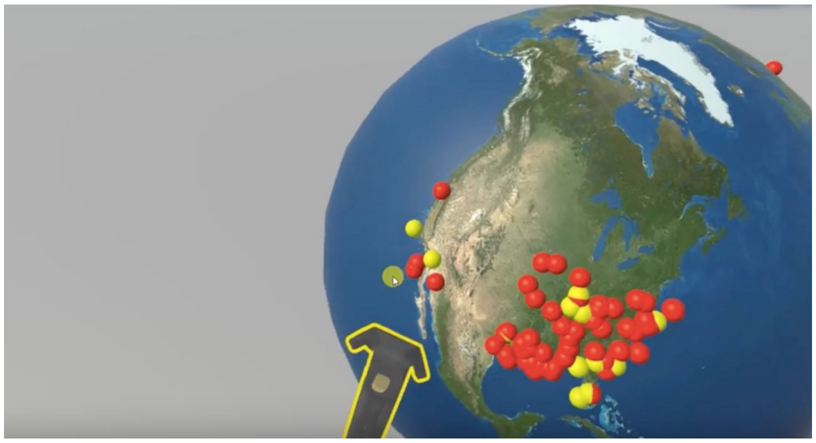
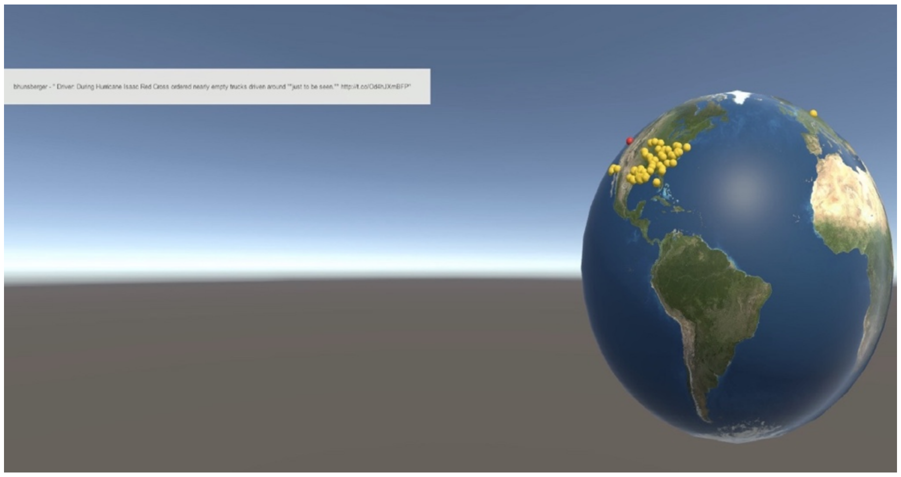


Figure 2 Color coded points

2.6 Use case 6 – Incorporate Tweets on the Globe

We plan on adding tweets to the globe using location information gathered by the VR4GETAR team. Each category of tweets will be color coded in order to make it easier for the user to visualize distinct events. Figure 3 shows a snapshot from our prototype showing how this will be implemented in the final product.



bhunsberger - " Driver: During Hurricane Isaac Red Cross ordered nearly empty trucks driven around ""just to be seen."" http://t.co/Od4hJXmBFP"

Figure 3 Geotagged tweet sample

2.7 Use case 7 – Incorporate Pictures on the Globe

Finally, we plan on adding geotagged pictures to the globe as shown in Figure 4. This will help the user in contextualizing pictures using their location. We mainly plan on using pictures from important events from different parts of the world. Figure 4 shows a sample image of an image positioned on a globe using its latitude and longitude information.

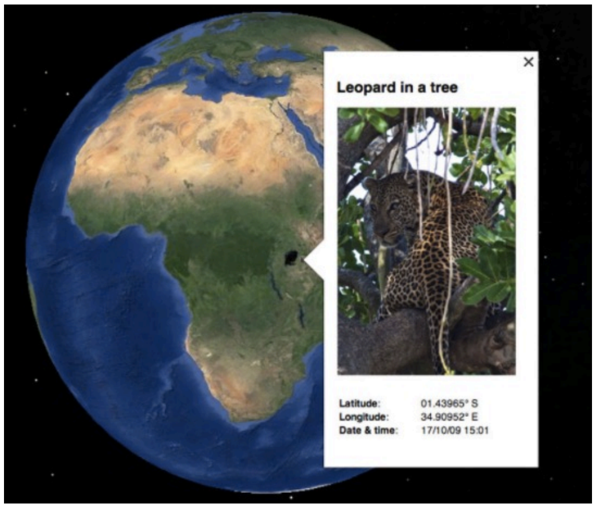


Figure 4 Geotagged picture- Sample

2.8 Special Requirements

- Since we are trying to provide an interface that mimics the real world as closely as possible, the VR environment should be able to operate without a lag that is usually attributed to low frame rates.
- One of the disadvantages of VR is motion sickness. It's associated most strongly with first-person shooters and walking games, which create a stark mismatch between your real and virtual body. We will try to eliminate this using prototypes to make the appropriate modifications.

2.9 Project Setup

The tools needed to set up the project are Git and Unity. First, the VR4GETAR repository must be cloned onto the machine. The VR4GETAR project must then be imported into Unity. This can be done from the Unity startup screen by clicking "Open", selecting the VR4GETAR project directory within the repository, and then clicking "Select Folder". After a brief loading dialog, Unity will open the project in its editor. In the future, the project will be available to open on the startup screen, without having to import it again.

2.10 Adding Data

All data for the VR4GETAR project can be found in the VR4GETAR\Assets\Data directory. Within this directory are various CSV files, JPEG files, and the Locations.txt file.

Tweets are stored in CSV files with the following format:

time,author,content,latitude,longitude

The application will attempt to read tweets from all CSV files in the directory at runtime so long as they have the ".csv" extension and follow the above format. To add a tweet, simply add to an existing CSV file or create a new one. If a new CSV file is created, do not include a header row.

Images are stored in the Data directory as JPEG files with names of the following format:

locationName_imageNameOrNumber.jpg

Since GETAR image data is not guaranteed to have location metadata, the application relies on this naming scheme and the Locations.txt file to determine the latitude and longitude values for images. To add an image to the project, place a JPEG file into the directory and ensure that the *locationName* portion of the name matches an entry in the Locations.txt file. The *imageNameOrNumber* portion of the name can be arbitrary.

2.11 Collections Schema

Data for collections is stored locally in text files with the ".col" extension. Each "collectionName.col" file represents a single collection with name "collectionName", and each line in such a file represents a single entry within the collection. The format of an entry is as follows:
data,position,rotation

The data portion of the entry is different for tweets and images. For tweets, it is the raw data for the tweet as found in its CSV file. For images, it is simply the name of the image. The position portion of the entry are the x, y, and z coordinate values that represent the entry's position in virtual space. The rotation portion are the x, y, z, and w values that make up the quaternion for the entry's orientation in virtual space.

An example of an entry for a tweet is as follows:

Sat Aug 15 16:16:49 +0000 2015, kate_hurricane, Just posted a photo,59.9341,30.3062,1.45,2.13,0.79,0.0,0.0,0.0,0.0

An example of an entry for an image is as follows:

```
image.jpg,59.9341,30.3062,1.45,2.13,0.79,0.0,0.0,0.0,0.0
```

This schema was chosen because it allows for simplicity of implementation and human-readability of collection data as plain text. Additionally, the ".col" files can be used to export collections between different environments.

2.12 Running the Application

Since VR4GETAR is a VR application, it requires an HTC Vive and a computer equipped with a sufficient GPU. Minimum requirements have yet to be determined for the application, but the least powerful machine that it has been tested on is equipped with an Nvidia GTX 970 GPU. The graphical simplicity of the application will likely allow it to run satisfactorily with a less powerful GPU, but VR is an inherently intensive application so this can not be guaranteed.

If the above requirements are met, the application can be run in two different ways. The first way is to use "play pmode" within the Unity editor. With the project open in the editor, simply click the large "play" button at the top of the window. The application will then be visible through the Vive. To stop the application, click the "play" button again. The second way to run the application involves building the project as a standalone executable. This can be done within the Unity editor by clicking File --> Build Settings… --> Build. Once the build process is complete, locate the new executable and ensure that its parent directory also contains a copy of the Assets\Data directory. This is necessary because the build process does not bundle the GETAR data into the executable.

3.0 Developer's Manual

3.1 Tools Used

Unity is a free multiplatform 3D video game engine that handles real-time graphics, sounds, physics, user interface, and other tasks which are necessary in creating interactive 3D environments. Developers can use its visual editor to build scenes, and its scripting frameworks allow for custom functionality.[4] Unity supports VR natively, providing a drag-and-drop solution that greatly simplifies VR development. Despite being a video game engine, Unity is flexible enough that it be used to create 3D environments for any purpose, including visualization. Unity was chosen for this project because of its ease of use and rapid development speed.[4]

C# is one of the two languages that are available for use with Unity. It will be used in this project for all programming tasks. It was chosen for this project because of its object-orientation and Java-like syntax.[4]

The HTC Vive is a VR system comprised of a head-mounted display, two controllers, and two "lighthouse" base stations.[4] Mounted high up in opposite corners of a room, the base stations allow the head-mounted display and controllers to individually track their positions and orientations in real space. The system translates that information into position and orientation in virtual space, which the user sees in stereoscopic 3D through the head-mounted display.[4] The Vive was chosen as the platform for this project because its "room-scale" tracking and detailed levels of interaction with virtual spaces provide highly immersive VR experiences. [4]

GitHub, a web-based version control repository and Internet hosting service will be used for collaboration among team members.[6] This choice was made because all team members had used a Git based version control system at one point or another in their undergraduate career. Additionally, since Git is decentralized, it is trivial to move from a development system to a demo system.[6] Finally, Git has cheap local branch tracking. This is especially important because our team will do most of the development on their laptops that do not have an abundance of internal storage.

3.2 Implementation Architecture

It is important to note that development in Unity exposes two separate concepts of objects. When writing scripts, objects in the OOP sense are present and operate as expected. When working within the Unity editor to create a virtual space, GameObjects are used to represent entities within that space. GameObjects are organized hierarchically and each can be thought of as containers of components. Components define single attributes or behaviors, such as position, physics state, or scripts.

The architecture of this project is comprised of two parts. First in this section is a description of the GameObjects, arranged in a hierarchical fashion. Second, the necessary scripts are described. This is typical of any Unity project. Game object is the most important concept in a Unity project. Every object in the application is a GameObject.[4] Therefore, the C# objects will be converted to GameObjects before being rendered.

- GameObjects:
    - Globe
    - Description: The globe upon which data points will be presented.
    - Components: transform, mesh, material, rigidbody, sphere collider
    - Children:
        - PointGenerator
            - Description: Invisible gameobject responsible for loading in the data and instantiating Point gameobjects.
            - Children: None.
            - Components: PointGenerator script
- Point
    - Description: Small sphere which represents one or more data points on the globe.
    - Children: None.
    - Components: transform, mesh, material, sphere collider, PointController script
- DataView
    - Description: Panel that appears when a Point is touched by the user, displaying a single piece of data (tweet, image)
    - Components: transform, mesh, material, IData script implementation
    - Children:
        - Text (if applicable) - Displays textual data
        - Image (if applicable) - Displays image data

- Scripts:
    - PointGenerator
        - Description: Extends MonoBehavior. Uses DataLoader to retrieve a list of IData objects, then instantiates and positions Point GameObjects according to each IData object's latitude and longitude.
        - Fields:
            - float radius - Defines the radius of the sphere to generate points on. Serialized to allow for editing in Unity.
        - Methods:
            - void start() - Monobehavior method that is called when the script component is created. All work is done/ called from here.
            - void generatePoints(List<IData> data) - Instantiates and positions Point gameobjects according to the IData objects in data.
    - Dataloader
        - Description: Static class. Collects text and image data resources, instantiating either a TextData or ImageData object for each.
        - Fields: None.
        - Methods:
            - List<IData> loadData(String resourceDirectory) - Collects text and image data resources and returns a corresponding list of IData objects.
    - PointController
        - Description: Extends MonoBehavior. Controls the behavior of Point gameobjects. Performs functions to show and hide data, as necessary.
        - Fields:
            - List<IData> data - List of data that the point represents.
        - Methods:
            - void showData() - Create gameobjects to show the user the data
            - void hideData() - Remove the gameobjects that represent the point's data

    - IData
        - Description: Interface describing a data object.
        - Methods:
            - Vector2 getLatLng() - Returns the latitude and longitude of the data.

        - GameObject createDataView() - create a DataView gameobject to represent the IData's data

- TextData
    - Description: Implements IData, represents textual data.
    - Fields:
        - Vector2 latLng - Stores the latitude and longitude.
        - String text
    - Methods:
        - TextData(String dataText, Vector2 latLng) - Constructor.
        - GameObject createDataView() - Create a DataView gameobject, using a 3DText object to display text.
- ImageData
    - Description: Implements IData, represents image data.
    - Fields:
        - Vector2 latLng - Stores the latitude and longitude.
        - Texture imageTexture
    - Methods:
        - ImageData(Texture image, Vector2 latLng) - Constructor.
        - GameObject createDataView() - create a DataView gameobject, applying imageTexture as a texture to a plane to display the image.
- DataCollection
    - Description: Represents a user-generated collection of data.
    - Fields:
        - int collectionID
        - String collectionName - User-assigned name.
        - List<String> tags - User-assigned tags.
        - List<IData> data - Data in the collection.
    - Methods:
        - DataCollection(int id) - Constructor that loads a previously generated collection.
        - void addTag(String tag) - Add a tag to the collection.
        - void removeTag(String tag) - If present, remove tag from the collection.
        - void saveCollection() - Writes the collection to disk.

The UML diagram (Figure 5) shows the relationship between classes. It also shows the flow of information through our application. Before the application launches, tweets exist in CSV files and images as JPEG files in a directory. These files are read by the DataLoader script which creates TextData and ImageData objects. These objects are aggregated as a DataCollection and then points are created depending on the location information the images and tweets contain. Finally, the points created are placed on the globe at the appropriate position.

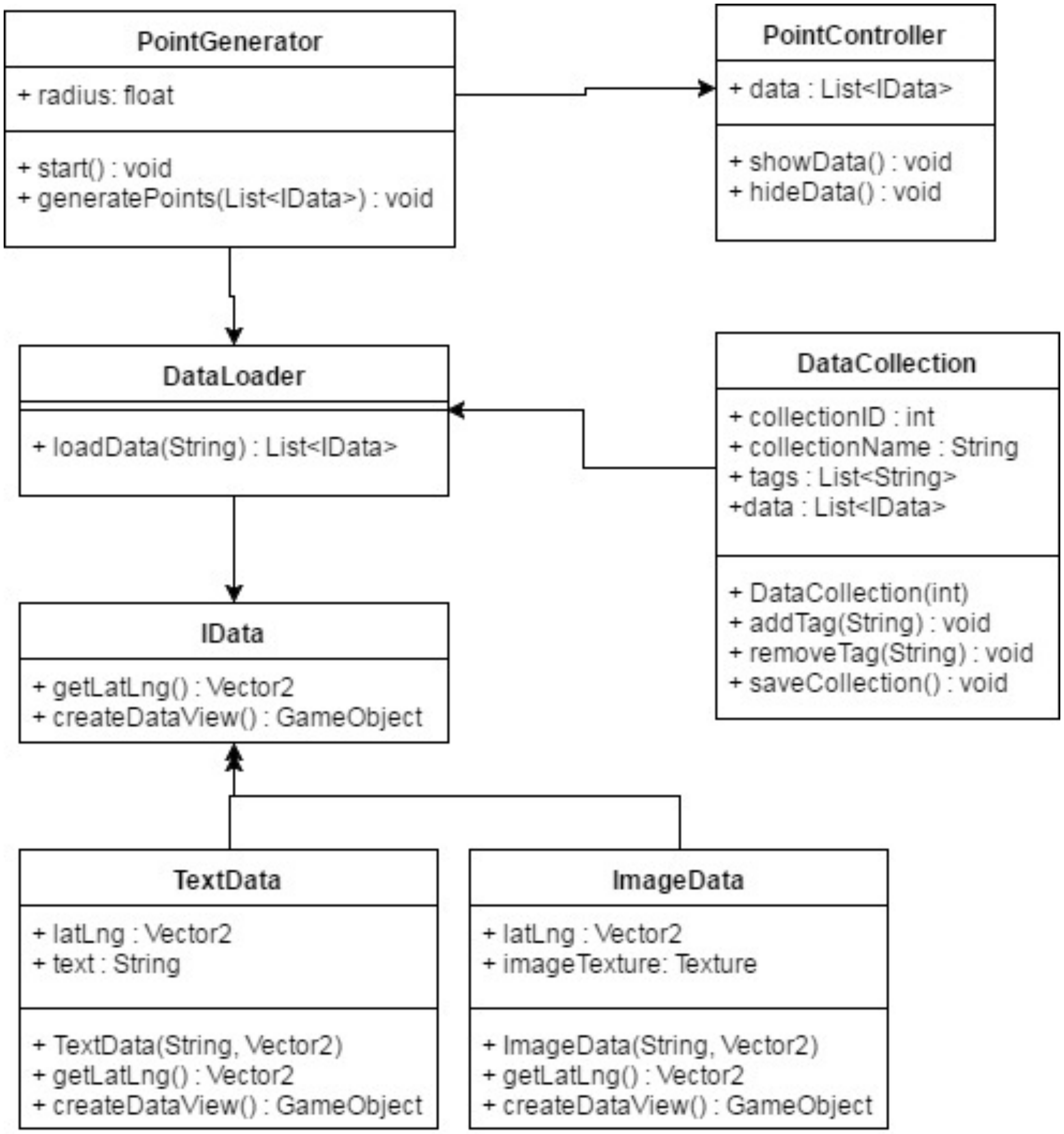


Figure 5 UML Diagram

Figure 6 shows the various transformations that the data will go through before they are displayed on the globe. Tweets start in CSV files and images start as JPEG files. They are then converted to C# objects. These objects are then inserted into an IData list which is finally run through the point generator to create data points on the globe.

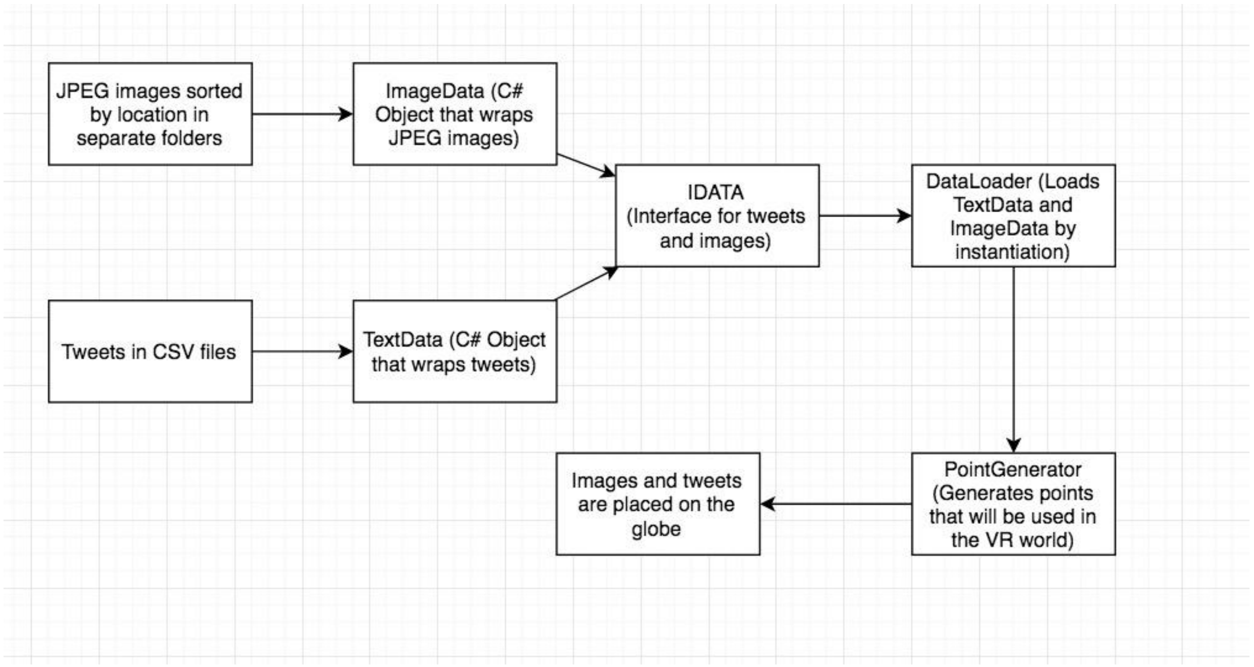

JPEG images sorted by location in separate folders
ImageData (C# Object that wraps JPEG images)
IDATA (Interface for tweets and images)
DataLoader (Loads TextData and ImageData by instantiation)
Tweets in CSV files
TextData (C# Object that wraps tweets)
PointGenerator (Generates points that will be used in the VR world)
Images and tweets are placed on the globe

Figure 6 Flow of data in the application

3.3 Testing

The application is subject to three kinds of testing: formative testing for the design of the user interface, unwritten unit tests of the logical systems, and user tests for each prototype. Presently, all tests are performed by members of the VR4GETAR team. Testing of the application is done with data provided by members of the GETAR team.

3.4 Formative Testing

User interface design is an iterative process of creating a design, receiving feedback, and then improving the original design.[2] This is necessary to ensure that the interface meets the user's needs of style and functionality. VR complicates this process by adding its own constraints, such as minimum levels of discernible detail due to display technology limitations, dynamic stereoscopic cameras which provide a different sense of scale than what is seen during development, and the varying proportions of different users' bodies.[2] Formative testing addresses these issues by allowing VR4GETAR developers to design what might be an effective VR interface, assess it through use, then refine it.

An example of where formative testing was used to refine a UI feature is the size of the data points on the globe. Figure 7 is an image of the data points in their original size. They were 1.25cm in diameter in the virtual space, making them easy to click and plainly visible during development on a traditional 2D display.

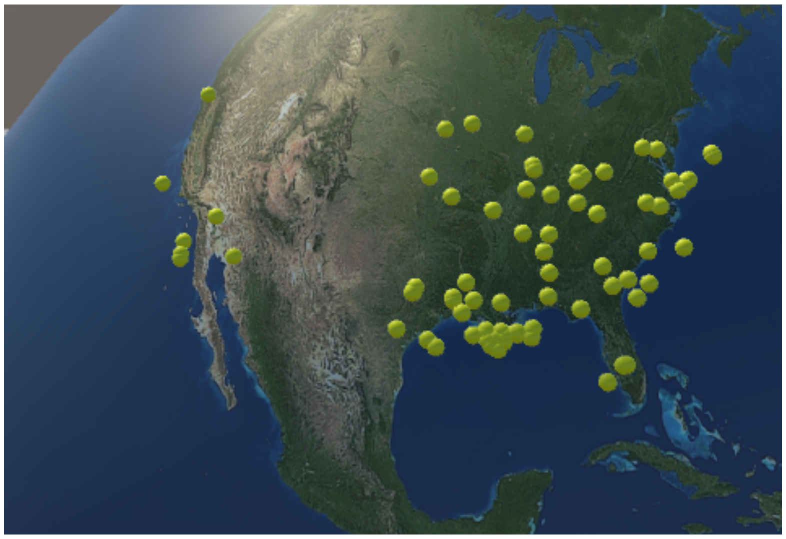


Figure 7 Points with diameter 1.25cm

Once in VR, the 1.25cm points were immediately found to be too small. They were difficult to interact with, and where many points intersected it was difficult to discern how many there were due to the low pixel density of the HTC Vive as compared to a monitor. The points were made larger, as is shown in Figure 8, but were found to be too big. Through trial and error in this way, the current point size, shown in Figure 9, was found to be the ideal compromise between space taken on the globe and ease of use.

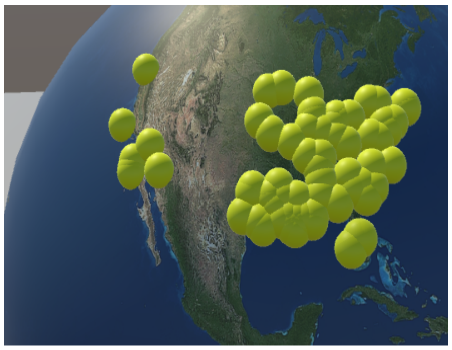


Figure 8 Points with diameter 2.5 cm

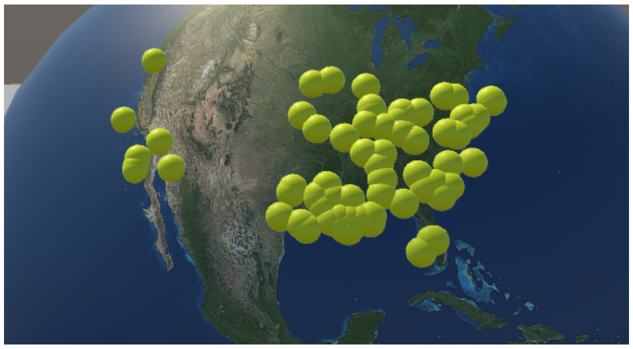


Figure 9 Points with diameter 2 cm

This approach is necessary in the development of all VR features.

3.5 Unit Tests

The logical systems of this project control how the user interacts with the virtual space, how objects are placed, and how data is processed. As features are introduced and developed, they are unit tested to ensure proper function. Unit tests for this project are not actual written test suites, however. This is because none of the features which are built in the Unity editor can be tested with written suites, since Unity does not support them. Instead, the development process involves testing with each small change. As something is being built, it can be tested in the editor using "Play Mode" to ensure that it is behaving as expected. Since this testing occurs very frequently during development, it is also effective at catching when changes have broken old features state.

3.6 User Testing

When a prototype is complete, all members of the VR4GETAR team who have access to an HTC Vive are expected to test the latest functionality of the application. This allows the team to determine the intuitiveness of the features, as well as ensure that the application runs well outside of the development environment.

3.7 Testing Data

The GETAR team has provided 200 tweets and 59 images for testing the application, with most of the data originating from the US. The tweets all relate to recent hurricanes, and the images are simply pictures of various cities. The testing data is important because it allows for testing the application at a scale that would have been time-consuming to replicate with junk data.

4.0 Prototypes, Timeline and Lessons Learned

4.1 Prototyping

In order to ensure that our final product meets the end user's requirements, we decided to adopt an iterative software development methodology which is a combination of rapid prototyping and agile development.[5] We utilized software prototyping as a practical means to develop our product whilst allowing users to try out our development ideas. Since we were not able to get actual users, we used feedback obtained from Dr. Edward Fox and our team to make modifications and add features to our prototype. Dr. Fox was helpful in providing an outsider's perspective and our team was helpful in providing the perspective of a developer who might want to add features and extend our product.

4.1.1 Prototype History

Table 3 Prototype History

Prototype Number	**Date**
1	02/13/2017
2	03/15/2017
3	04/05/2017
4	04/19/2017
Final	04/25/2017

4.1.2 Prototype 1

Status	Completed
Features	• View globe in a VR world • Rotate the globe • Pick and throw objects
Comments Received	Upon showing this prototype, Dr. Fox suggested that we use the VR world to intuitively create and modify collections for data exploration.
Refinement	After brainstorming and considering Dr. Fox's comments, we decided to place the globe in a space themed background for a more realistic appearance. Additionally, use case 3 (Create and modify collections) was added.
Snapshot(s)	Figures 10 and 11 show what the user would see using an HTC Vive. The first picture shows the globe on the table and the second picture shows the VR room in which the user would walk and interact with things.

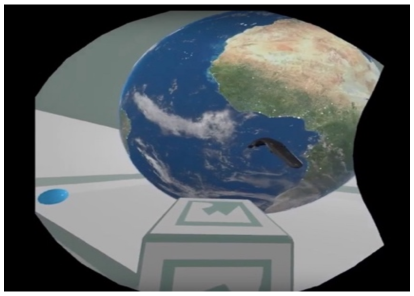


Figure 10 Globe

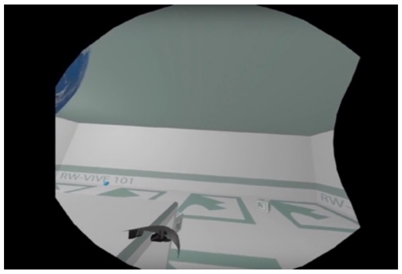


Figure 11 VR Room

4.1.3 Prototype 2

Status	Completed
Features	• The globe is placed in a space themed background • Data points are represented on the globe using dots • Tweets associated with data points can be viewed by clicking on the corresponding point
Comments Received	Upon showing this prototype, Dr. Fox suggested that we use different colors to represent the type of data (tweets or images).
Refinement	After considering Dr. Fox's comments, we added use case 5 (color coded data points). This use case will be implemented in prototype 4.
Snapshot(s)	Figure 12 shows a tweet that was accessed by clicking on a data point on the globe. The color red is used to show data points that have been opened and the color yellow is used to represent all the other points.

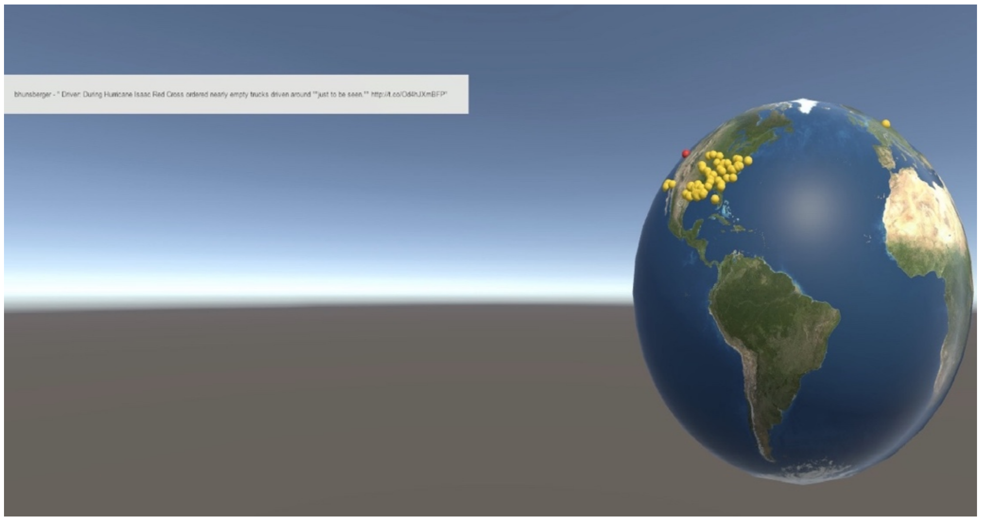

bhunsberger - " Driver: During Hurricane Isaac Red Cross ordered nearly empty trucks driven around ""just to be seen."" http://t.co/Od4hJXmBFP"

Figure 12 Snapshot of our second prototype

4.1.4 Prototype 3

Status Completed

Features
- VR interactions to select data points on the globe and to walk around in the VR environment
- View images in a VR world
- View tweets in a VR world

Comments Received None

Refinement In our next prototype, we used color coded points to represent the type of data i.e. distinct colored data points will be used to differentiate between tweets and images.

Snapshot(s) Figures 13, 14 & 15 show how the user would interact with the globe to view tweets and images. Figure 13 shows how the user will use HTC Vive's controller as a laser pointer to select data point. Figures 14 & 15 show how tweets and images are displayed in the VR world.

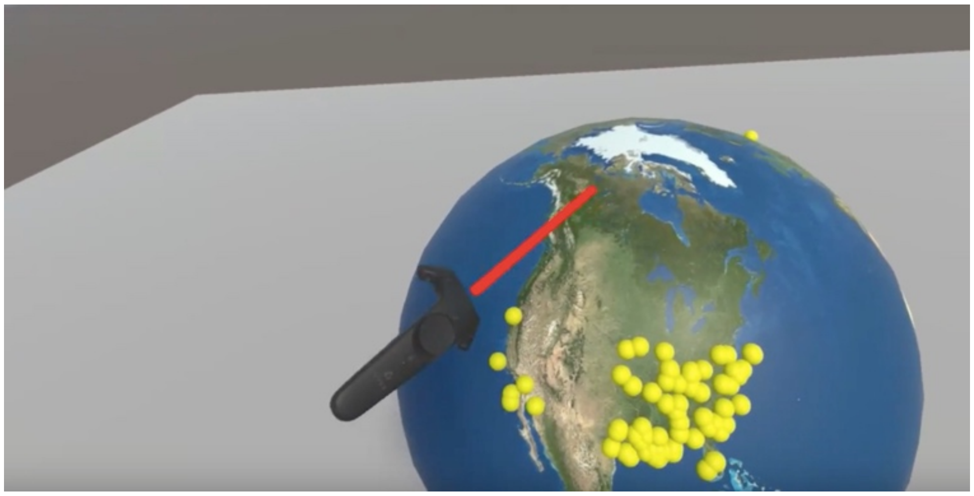


Figure 13 Selection using HTC Vive's controller

Figure 14 Picture displayed in VR

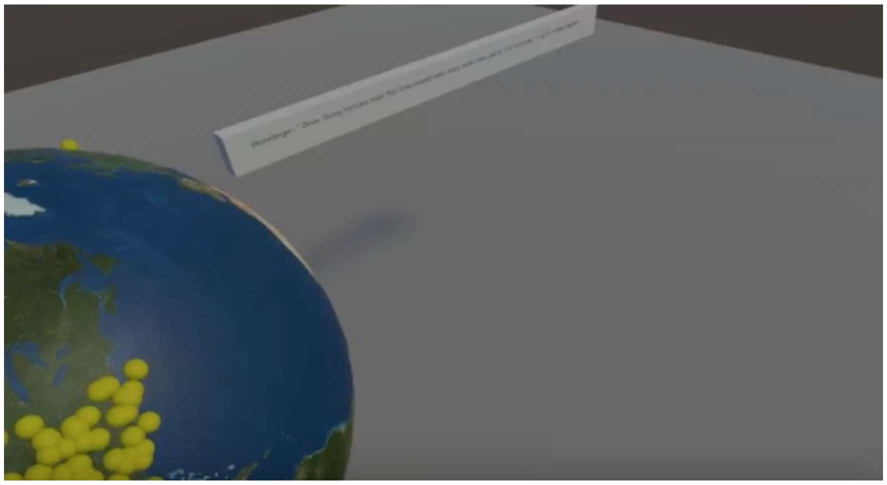


Figure 15 Tweet displayed in VR

4.1.5 Prototype 4

Status	Completed
Features	• Ability to create and modify collections • State persistence to save the state of the system when the application is shut down • Color coded data points
Comments Received	Upon showing the prototype to Dr. Fox, he suggested that we include a demo video with our final deliverable. Additionally, he suggested that including labels would let the user know that the numbers 1, 2, 3, etc. represent collection numbers.
Refinement	After considering Dr. Fox's comments, we will include an annotated demo video hosted on YouTube. Furthermore, we will include labels to provide information about the use of a 3D object in the VR world.
Snapshot(s)	Figure 16 shows the 3D objects in the VR world that will be used to select a collection and modify them. Figure 17 shows how the data points in a given collection will be displayed in the VR world.

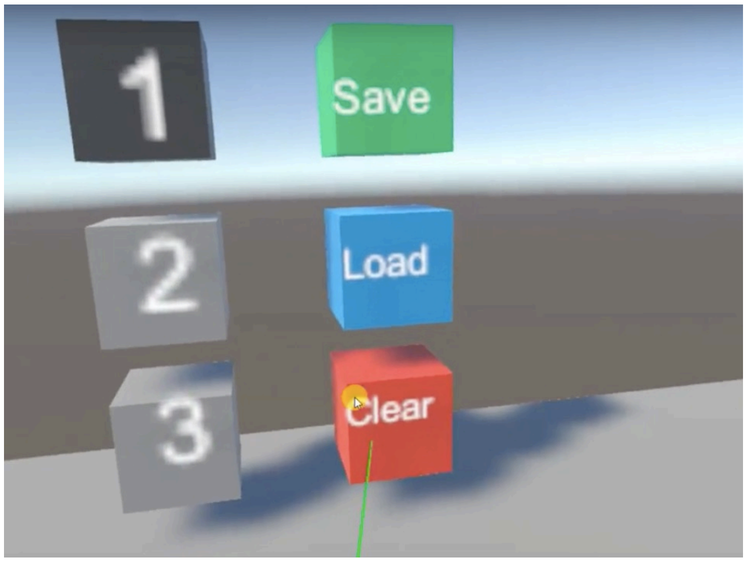


Figure 16 Collections and Action Buttons to Modify Collections

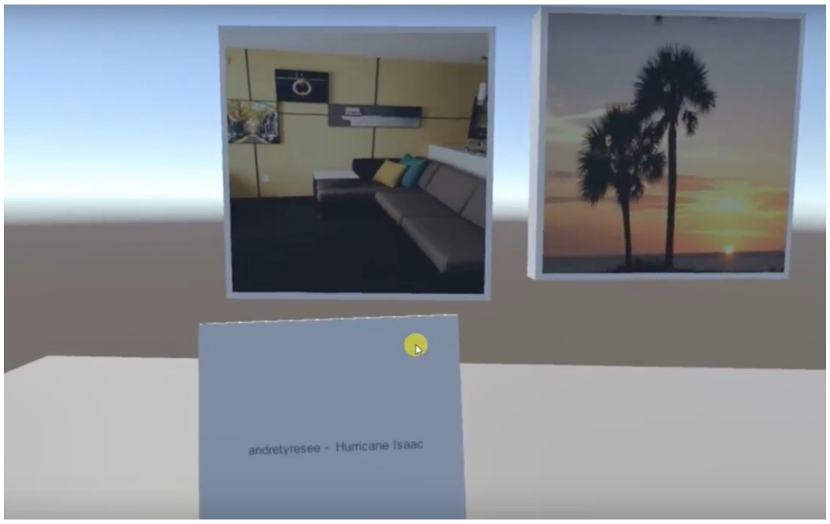


Figure 17 A Sample Collection displayed in the VR World

4.1.6 Final Product

Status	Completed
Features	No new features were added. In this stage, prototype 4 was refined to meet the end user's and Dr. Fox's specifications.
Comments Received	None
Refinement	N/A

4.2 Timeline

In order to ensure the success of any team project, division of responsibility is paramount. This ensures autonomy and responsibility among team members.[5] Therefore, utmost consideration was given to the strengths and the weaknesses of each team member during the process of delegation.

Since we chose to follow an iterative software development methodology, we got feedback from the client after every prototype to allow for frequent improvements and modifications to our model. This is illustrated in the Gantt chart below.

The Gantt chart (Figure 18) shows a high-level breakdown of tasks along with a timeline for the tasks.

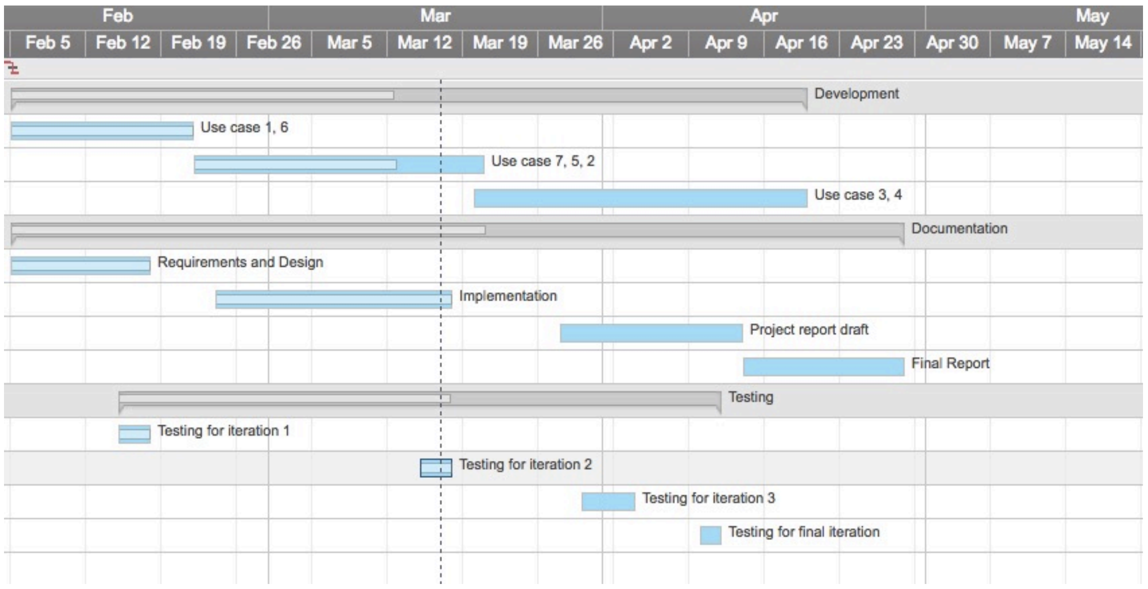
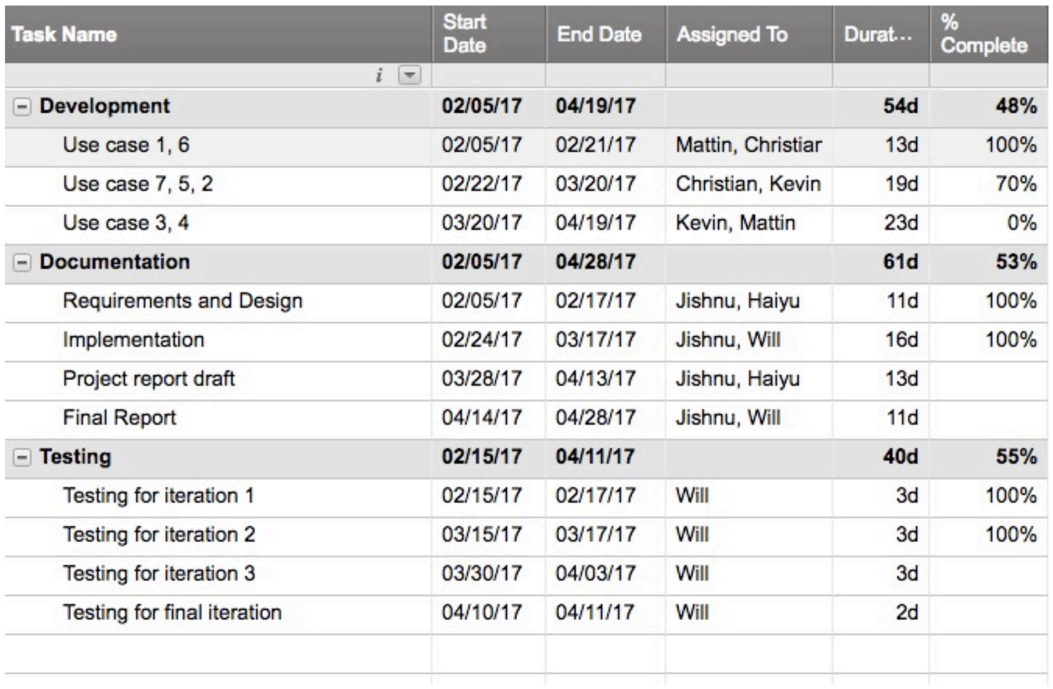


Figure 18 Gantt chart

Table 4 shows the division of responsibility of tasks among team members along with the percentage of the tasks that have been completed.

Table 4 Division of responsibility



Task Name	Start Date	End Date	Assigned To	Durat...	% Complete
Development	**02/05/17**	**04/19/17**		**54d**	**48%**
Use case 1, 6	02/05/17	02/21/17	Mattin, Christiar	13d	100%
Use case 7, 5, 2	02/22/17	03/20/17	Christian, Kevin	19d	70%
Use case 3, 4	03/20/17	04/19/17	Kevin, Mattin	23d	0%
Documentation	**02/05/17**	**04/28/17**		**61d**	**53%**
Requirements and Design	02/05/17	02/17/17	Jishnu, Haiyu	11d	100%
Implementation	02/24/17	03/17/17	Jishnu, Will	16d	100%
Project report draft	03/28/17	04/13/17	Jishnu, Haiyu	13d	
Final Report	04/14/17	04/28/17	Jishnu, Will	11d	
Testing	**02/15/17**	**04/11/17**		**40d**	**55%**
Testing for iteration 1	02/15/17	02/17/17	Will	3d	100%
Testing for iteration 2	03/15/17	03/17/17	Will	3d	100%
Testing for iteration 3	03/30/17	04/03/17	Will	3d	
Testing for final iteration	04/10/17	04/11/17	Will	2d	

4.3 Lessons Learned

4.3.1 Development

Features Implemented	• Ability to interact with the globe in an intuitive manner (walk around, rotate the globe, touch data points). • View JPEG images with location data as data points on the globe. • View tweets with location information as data points on the globe. • Ability to create and modify collections by the user. This feature will aid data exploration. • State persistence to save the current state of the application before it shut down. • Use color coded points to relay additional information about the data points.
People involved	Mattin Zargarpur – Development Jishnu Renugopal – Feedback Willem Schmidt, Haiyu Zhao, Kevin Zhang – Assisted development
Lessons learned	• Use of color coded points will convey additional information to the user without having to open each data point. • Adding state persistence and the ability to create and modify collections will make the application more useful for data exploration. • Placing the globe in a space themed background will make the model more realistic.

4.3.2 Documentation

Work done	This project was documented in the form of a project report • A developer's manual which is intended to be used by developers who might be interested in extending this project by adding more features. • A users' manual intended to be used by people using this application for data exploration. • A documentation of the structure of the code base and the data flow. • A detailed explanation of the tools used to develop the application • Used Dr. Fox's comments on the interim report to polish the final report. • Updated the prototype section to reflect the final product. • Updated the testing section to reflect the final product.
People involved	Jishnu Renugopal – Documented design, project overview and methodologies used. Mattin Zargarpur – Documented implementation and testing procedure
Lessons learned	It is crucial that the project report meticulously document all parts of the product development. This ensures that the end user is able to set up the application with ease. Additionally, this also helps subsequent developers working on VR4GETAR to use the report to gain some context about the project without much effort.

4.3.3 Testing

Work done	• Formative testing was done to ensure that the VR interface works as expected without any issues like lag, low frame rate, etc. • Unit testing was done to ensure that the internal logic of the application is performing as expected.
People involved	Mattin Zargarpur – Lead tester Willem Schmidt, Haiyu Zhao, Kevin Zhang – Assisted testing
Lessons learned	Applications developed for a 2D interface such as a desktop might not transfer well onto a 3D interface such as VR.

5.0 Future Work

Due to time constraints, the user-generated collection feature is limited to three total collections. Support for an arbitrary number of collections would provide an improvement to the application. Further, tracking collection metadata such as name, author, and time of creation, would add polish.

Another possible improvement to the application, given sufficient development time, would be to add more visualizations. Currently, the application allows users to view data directly, but showing the relationships between various pieces of data could also be insightful.

Of course, being that VR4GETAR is a 3D application made without any artists, the visuals could be improved.

Acknowledgements

This research was supported by Virginia Tech. We thank the VR4GETAR team which provided insight and expertise that greatly assisted the research. We would specifically like to thank Liquing Li who provided us with data for our prototypes.

We thank Dr. Edward Fox – our instructor and client for his assistance and for the comments that greatly improved the project report. He can be contacted via email at fox@vt.edu. Additionally, we would like to thank him for his feedback on our prototypes which greatly helped us in making our final product better. This project is based upon work supported by the National Science Foundation under grant numbers IIS-1619028 and 1619371.

References

1. **Erdogan, I., & Campbell, T. (2008). Teacher Questioning and Interaction Patterns in Classrooms Facilitated with Differing Levels of Constructivist Teaching Practices. *International Journal of Science Education,30*(14), 1891-1914. doi:10.1080/09500690701587028**
2. **Lin, J., & Schulze, J. P. (2016). Towards Naturally Grabbing and Moving Objects in VR. *Electronic Imaging,2016*(4), 1-1. doi:10.2352/issn.2470-1173.2016.4.ervr-415**
3. **Data Visualization and High-Dimensional Data Clustering. (2015). Retrieved March 17, 2017. *Clustering,* 237-261. doi:10.1002/9780470382776.ch9**
4. **Technologies, Unity. (2016). Unity User Manual (5.5). Retrieved March 17, 2017, from https://docs.unity3d.com/Manual/index.html**
5. **Doorewaard, H., Hootegem, G. V., & Huys, R. (2002). Team responsibility structure and team performance. *Personnel Review,31*(3), 356-370. doi:10.1108/00483480210422750**
6. **GitHub. (2017, April 10). Retrieved April 11, 2017, from https://en.wikipedia.org/wiki/GitHub**