Visualization of Climate and Earth System Modeling

PROJECT PLAN

Team Number: May1701
Client: Prof. Chaoqun (Crystal) Lu
Advisers: Prof. Johnny Wong

Team Members/Roles:
Kellen Johnson – Team Communication Leader
Anish Kunduru – Team Leader
Julio Salinas – Team Concept Holder
Eli Devine – Team Webmaster

may1701@iastate.edu
http://may1701.sd.ece.iastate.edu/
Revised: 12/2/16, version 3.0

Contents

1 Introduction

1.1 PROJECT STATEMENT

The product we are to build consists of a web application that allows users to query and view various maps that represent "natural and anthropogenic stresses" (compounds that are added to the ecosystem). Our client (Prof. Chaoqun Lu) currently develops 2D .tiff images (see Appendix 9.1) that represent maps of various compounds and would like us to create a platform that would allow dynamic loading of her various maps onto a 3D space (see Appendix 9.2). Currently, our client analyzes compounds using a proprietary algorithm and creates a graph data stored in an ASCII (American Standard Code for Information Interchange) format. The client has two issues as it stands: The first is the computation of this graph data takes several days or weeks for each model. The second is the client has no easy way to display this data for actors to view and interact with it. The goal of our project is to focus on the second issue: provide a way for interested parties to view data in a user-friendly and interactive 3D environment. If time permits, our secondary goal is to use our computer science knowledge to see if we can assist our client in speeding up her algorithm.

1.2 PURPOSE

The Visual Earth Modeling System will allow various policy makers (government, grant societies, etc.) and land managers (real-estate, governmental, farmers, etc.) to visually examine, further analyze, and understand spatial patterns of various gas compounds, water discharges, and nutrient movement. Allowing these stakeholders to further understand the environment will aid them in making decisions in reference to pollution policy, development of land to be used for various crops, grant making, etc.

1.3 GOALS

In addition to helping our client bring her work to an interactive and 3D space:

- Create a platform which allows user to view monthly mappings of various compounds.
- Create a platform which allows users to forecast the effects of various compounds on the environment.
- Provide a platform that can be used as a global learning tool to help interested parties understand the implications of human procedures on the general ecosystem.
- Build a sound product that could be further implemented in the future (by our team or other developers) if the client so wishes.
- Create a tool that automatically creates visual representations without additional effort necessary from the client.

2 Deliverables

The Deliverables to meet the Projects Goals are as followed:

- Webpage that displays various layers of compounds onto a 3D space.
- 3D Map-Service layering that can be dynamically loaded and chosen by the user for every available data model.
- Web-hosted parser to allow client to parse ASCII maps into dynamically loading webpages. This will be ultimately done through ArcGIS for Server.
- Server which will serve as host for the website, ArcGIS for Server background hosting, and ArcMap/ArcPy Processing Tools

3 Design

3.1 PREVIOUS WORK, ASSOCIATED LITERATURE, PAST DEVELOPMENT

Since ArcGIS is the platform that we have chosen to develop on, the bulk of the informational material required to map this project will be provided by the ArcGIS JS API (3) and related materials of ArcGIS for Server, ArcPy Modules, and ArcMap. Throughout the semesters, we will be able to go through and view, edit, and create various pieces onto which our client will be able to pick which data-type she prefers. Once semester two starts, we will likely have to look into dynamically editing ArcGIS layers using ArcPy. We have also contacted an Iowa State ArcGIS analyst, Josh Obrecht, who has said he would be happy to share his expertise if we require it.

Our first approach was to use Google Earth as our client and her team had suggested. The Google Earth plug-in and the associated API allows developers to seamlessly incorporate all Earth manipulation capabilities we require directly into a webpage (4). It also allows for the ability to be able to create various 3D objects, which is a feature we will be required to incorporate into our design. The API also provides the ability to be able to dynamically load your own database, which you can then display on a Google Earth globe. Unfortunately, our team found out that the Google Earth API would be deprecated by the end of the year. The reasoning behind it is that the Google Earth API was originally constructed upon an insecure plug-in which Google Chrome and Mozilla Firefox alike have decided to no longer support (5).

One of the similar products that we looked into was Landmark's DecisionSpace Earth Modeling (see Appendix 9.7.1). In a nutshell, this program not only displays 3D graphical data to the user, but also allows the user to further understand the values that are being placed (6). It also allows for the dynamic elements that we may have to incorporate in semester two, which is why this program may be of help to us moving forward. If we are able to utilize the physics and pixel-to-pixel communication of dynamic objects, this would be another possible implementation.

Additionally, there is currently similar work being done by the United States Geological Survey (USGS). Their site provides real-time data on current and past conditions and how this impacts current predictive observations of Earth. An example of data processing that is similar to work done by our client is related to water discharge. Specifically, from the USGS site one can see a map of the U.S. that shows daily streamflow percentages, which is the same type of data, water discharge, our client has recorded and would like mapped. The water discharge is the volume of water moving

down a stream or river per unit time. USGS measures their streamflow/water discharge by subsections of channel of water, where the area of the subsection is equal to the product of the depth of the subsection and the width. The discharge for the subsection equates to the product of the velocity of the stream flow and the area calculated in the previous statement (8). Dynamically loading data is provided through the ability to click on a specific state for more details (see Appendix 9.7.2).

In our own attempts this semester, our team moved its focus to using a CSV Layer via ArcGIS for JavaScript. After parsing all of the Data into CSV files, we attempted to dynamically load these files onto a 3D map using the ArcGIS for JavaScript Web API. We would quickly determine that loading such a large number of plot points is unacceptably slow (on the order of minutes for slower laptops). Additionally, the CSV layer type that we used only allowed ~17,000 points to be loaded this way before discontinuing the plotting of points (see Appendices 9.4 & 9.5). While we were correctly reading the client's data (can be inferred by comparing shape of colored .tiff (Appendix 9.6) to shape of output from Appendices 9.4 & 9.5), this idea was scrapped due to performance and technical constraints. Due to this unsuccessful implementation, we decided to look towards ArcGIS for Server for answers.

3.2 PROPOSED SYSTEM BLOCK DIAGRAM

See Appendix 9.3.

3.3 ASSESSMENT OF PROPOSED METHOD

Our current plan is to host each month of each compound (each individual data-set) as its own individual map service. This is for stability reasons and gives us the ability to limit CPU and memory usage. We may later find it more practical to merge maps together by adding multiple layers to a single map. We will still use our parser which will take all ASCII files uploaded by our client and transform them into CSV files to be used as the datasets that are used for creation of map-services and their associated layers. These data sets will then be saved onto our local GIS server machine; and we will use these for the automated publishing of map documents to our GIS Server using ArcPy. From ESRI, this automation has been deemed a "four-step" process (7).

In summary, these steps are:

1. Create the Draft Service Definition.
2. Use ArcPy to analyze for any build errors.
3. If successfully analyzed, convert to Service Definition; else, troubleshoot and restart.
4. Publish as Service.

This process is summarized from Reference 7.

Finally, any hosted map service that was created using the above process, will be able to be called using the ArcGIS RESTful API. As a group, we wish to start off modular and can always tightly couple later if we see performance benefits during testing.

3.4 VALIDATION

Our product will be finalized when end users are able to queue up the 3D environment, interact with the globe, and dynamically load all of the data that the client currently has in 2D visuals, onto

a 3D space. The client will also be able to upload all future calculated mappings using the previous format demonstrated in her ASCII files. These mappings will be uploaded to the server and be selectable by any end user. For extended life-cycle demonstrations, the plan is to provide users with the ability to introduce various selectable compounds to the 3D environment.

4 Project Requirements/Specifications

4.1 FUNCTIONAL

Demonstrate climate change dynamics across space: A website that projects the ASCII data given to us by the client on a 3D model of a globe (visual data modeling).

- The product shall allow the client to upload raw ASCII data to the server.
- The product shall parse all uploaded data from an ASCII file into a format readable by ArcGIS.
- The product shall automate the steps traditionally taken to create a map-service (typically done via ArcGIS Desktop) by utilizing Python to take the CSV files and create a map, layers, and finally, a publish a map service.
- The product shall allow end-users to select the compound on the relevant timeline they would like to view. These datasets will be ordered by year and month. If the client would like to add more intervals or sorting methods for data, modifications must not require more than 8 staff hours from a coding standpoint.
- The product shall allow users to queue multiple datasets at the same time; meaning the end-user will be able to view multiple compounds at the same time (Example: CH_4 and CO_2 concurrently).
- The product shall be viewable from an Internet browser when accessed using the Iowa State VPN (stipulations for this are mentioned earlier).

4.2 NON-FUNCTIONAL

- The parsing of uploaded ASCII data should take less than 15 seconds, provided the dataset is smaller than 50,000 points.
- The product shall check the server for new ASCII datasets every 15 minutes; meaning datasets should be viewable by all users within approximately 15 minutes of being uploaded by the client.
- The viewing (rendering) of selected data shall be generated in less than 30 seconds; meaning the viewable data sets will be loaded by the product in less than 30 seconds.
- Once selected for creation, a dataset should be automatically created (from a CSV file) into a map service (using our automated Python scripting) in less than 10 minutes. Note that this timing is purely subjective to the speed of the server that we are given and is hard to quantify without real-world experience. Furthermore, the ArcGIS API calls cannot be modified and are not parallelizable, so multiple ASCII files uploaded in short time will simply need to be queued. As such, this requirement is flexible as will likely be changed in a later revision of this document.
- The end-user should be able to choose the color scheme to display the various datasets in.

- When values are toggled, the values should be appropriately sized to the point that they will not cascade, or billboard, toward the user. The user should be able to differentiate between various points through color and/or size.
- The system will be able to seamlessly support 8 simultaneous users. Rationale: If more than 8 users attempt to use the service at the same time, global page load times will increase up to 30 seconds (as stated earlier). After load times exceed 30 seconds, users will begin to get error messages. This requirement is largely based upon the CPU power the University is able to allocate to our server.
- 2 users will be able to simultaneously browse the same map (same compound for the same month). Rationale: If more than 2 users attempt to view the same map at the same time, page load times for those users will increase up to 30 seconds. After load times exceed 30 seconds, those users will begin to get error messages. This requirement is largely based upon the CPU power the University is able to allocate to our server.
- Web server usage will take priority over the processing of newly imported compounds. In other words, if 8 users are trying to browse maps, then processing of any additional ASCII files will slow down until web usage diminishes. This ultimately means that performance targets for the processing of newly added data (as stated in earlier requirements) is contingent upon the server not being utilized at the time.

5 Challenges

Some of the early challenges our team came across with was that the original plan for the project was to use Google Earth as our main platform. Unfortunately, our team quickly came to the realization that Google Earth is deprecated, and we had to look for alternative methods to model the data. Once our team found a suitable replacement came our hardest task to date: creating an Earth Visualization Model from scratch.

Originally, we attempted utilizing only ArcGIS for JavaScript 4.1 to model the data. While we had made significant progress in this regard (that our client was satisfied with), we were still working on transposing the ASCII data into a format readable by the ArcGIS API. Another issue that we were working on is making sure that the data displayed is accurate to the actual position of the data on the map. In other words, the demo that we designed creates points relative to screen pixels and we were trying to use the ArcGIS API to figure out how to create point sizes relative to map coordinates. This had something that had been a challenge for us as we were trying to avoid using an external map service which would increase costs (ArcGIS is currently free, as the ESRI CDN for ArcGIS is free to use) and cause much additional work at the automation stage. Using purely ArcGIS JavaScript became unrealistic, however, due to the large dataset size (as described in 3.2). Trying to load such a massive dataset does not allow for quick, complete, and interactive rendering, so we had to move towards ArcGIS for Server and associated published map services.

In addition to our projection system and possible system costs, we are faced with the problem of learning an entire new API/Framework (ArcGIS). As none of our group members have interfaced with this type of software, it is a bit of a challenge to learn due to the lack of examples online. In addition, most people that work on ArcGIS software use various GUIs/tools to do all of their development, so this decreases our example size even further for incremental learning. While we've experimented seemingly enough with ArcMap (Desktop GUI), and we further understand what we need to do, we still have to automate this entire process using ArcPy. Since we will be automating the process of creation and publishing, we are left with even less examples to learn from. Additionally, many of the ArcPy interfacing when editing objects using strictly Python is read-only. Because of this, our team will likely have to write a sort of plug-in of our own to allow for complete control of this process.

Finally, we will have the problem of attempting to plot very large data sets. Due to the massive size of the data samples given to us by Professor Lu, we will have to minimize the amount of objects placed on a map at any given time. Too many objects being placed will not only slow down the load time, it may crash the browser. Therefore, in addition to parsing the original input data, we will have to turn it into smaller objects. Should load time sensitivity become a greater issue later on in our project, we will have to switch map layers and publish things more intelligently on the map server. This is something that we are currently in the middle of testing and is largely dependent on the server resources allocated to our team.

6 Timeline

6.1 FIRST SEMESTER

As a team, we have decided on the following:

September – Initial Meetings with Client / Adviser / Team, Search for Usable Product

October – Initial Testing with ArcGIS Online (JS), Parser Construction

November – Installation of Needed Software, Attempt to Create 3D mapping

December – Finalize Documentation, Present to Board, Move Forward with Automation

The full timeline of the first semester can be seen in Appendices 9.8.a – 9.8.d.

In sum, we wanted to accomplish all of the following in the first semester: installation of all required software, development of the parser and team website, and creation of 3D map services by hand. Currently, we have accomplished all of the above.

As it currently stands, we have developed a prototype which will allow the user to display our client's data successfully on a 3D space (see Appendix 9.10). What we will be moving forward with in the near future and into the beginning of semester two will be the automation of this process using ArcPy. Once we are able to automatically plot the points of a set of data successfully, we will move on to plotting multiple data sets. We will create concrete map service examples to help us get familiar with the process and test this process's performance and stability in order to prepare us for the second semester's work and implementation.

6.2 SECOND SEMESTER

The second semester will consist of the following:

January – Finish Automation

February – Multiple Map Selection

March – Pixel-to-pixel Communication (User customization) and extra features

April – Finalize Product, Present to Board

The full timeline of the second semester can be seen in Appendices 9.9.a - 9.9f

Since we were able to get our prototype working, our second semester will begin with working on and finishing the automation of the creation process and the creation of a legend to allow the user to further understand their experience. After that, the second semester will largely revolve around allowing the users to customize their experience with the product. By allowing the user to toggle values, or giving various points the ability to "talk" to one another, we will be able to create an environment which allows users to use a "sandbox" mode and forecast compound readings. We will begin by allowing users to toggle values (9.9.a), and then lead into pixel-to-pixel communication (9.9.b), where we will configure automation and expansion on Dynamic Layers before we begin refinement and allowing the client to add extra features. As we cannot project how long the times

for (9.9.a) and (9.9.b) will take, we have allowed extra room in the timeline in case we need to extend our time frame. These times will likely expand in the future once more detailed implementation begins. To see the full timeline for semester two see Appendices (9.9.a – 9.9.f).

7 Conclusions

In summary, our project will revolve around plotting the work of Professor Lu and allowing various end-users to view this work in a revolutionary space, with the ultimate goal of allowing the same end-users to simulate their own data. We plan to work largely in a series of small iterative cycles, which will allow us to demonstrate each iteration to the client, in case we need to make adjustments. Additionally, working in these small iterative cycles will allow us to continue to further analyze the ArcGIS API as a reference point, give us ample time to contact experts (Josh Obrecht) in the field, and simultaneously keep our project moving forward. Our long term goal is for us to complete these iterative cycles and steps to produce a well-designed product that is clear, well-constructed, easy to access and navigate, and allow free-use for our client needs and further to the public. By the time the product is finished, users will be able to understand the effects of coinciding compounds upon the environment in a space that is visually pleasing and informative.

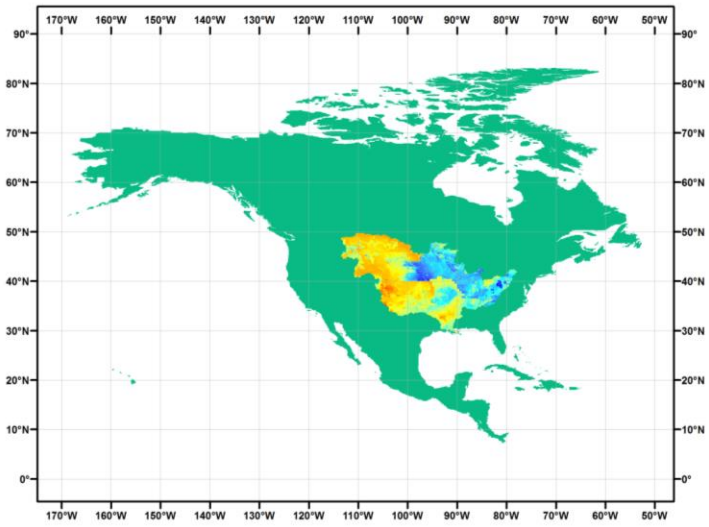
8 References

1) Landmark DecisionSpace Visual. Digital image. *World Oil*. World Oil, 1 Sept. 2010. Web. 18 Oct. 2016. <http://www.worldoil.com/news/2010/9/1/halliburton-s-landmark-announces-release-of-decisionspace-desktop-software-suite>.
2) "USGS Current Water Data for the Nation." *USGS Current Water Data for the Nation*. N.p., n.d. Web. 18 Oct. 2016. <http://waterdata.usgs.gov/nwis/rt>.
3) "ArcGIS API for JavaScript." *ArcGIS for Developers*. Esri, n.d. Web. 18 Oct. 2016.
4) “Google Earth Developers Guide” Google Developers. N.P., n.d Web. 14 Nov 2016. <https://developers.google.com/earth/>.
5) “Announcing deprecation of Google Earth API”. Google Geo Developers Blog. N.P., n.d Web 14 Nov 2016. <https://googlegeodevelopers.blogspot.com.au/2014/12/announcing-deprecation-of-google-earth.html>.
6) "DecisionSpace Earth Modeling." Landmark Solutions E&P Software. Halliburton, n.d. Web. 18 Oct. 2016.
7) "PublishMSDToServer." *ArcGIS for Desktop*. ESRI, n.d. Web. 19 Nov. 2016.
8) Perlman, USGS Howard. "How Streamflow Is MeasuredPart 2: The Discharge Measurement." *How Streamflow Is Measured. Part 2: The Discharge Measurement : USGS Water Science School*. Howard Perlman, 02 Dec. 2017. Web. 06 Dec. 2016

9 Appendices

9.1

The client's current solution to her visualization need:

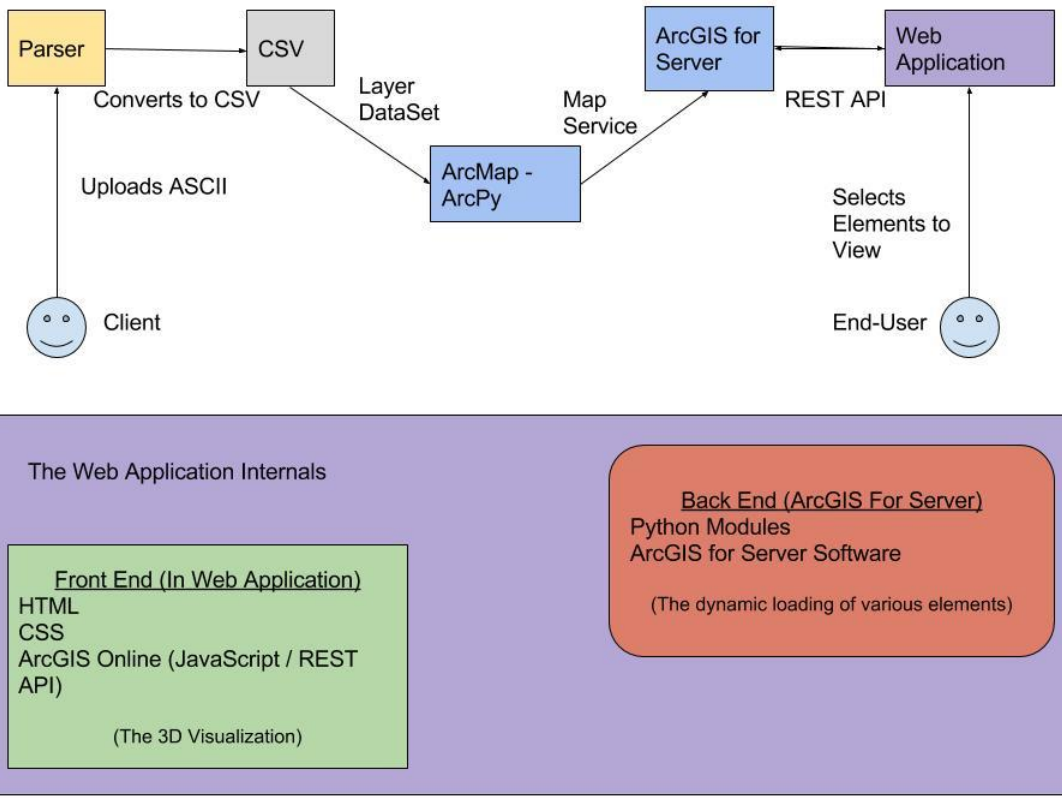


9.2

The sample 3D space onto which we'd like to project our client's work:

9.3

The proposed Block Diagram of the end product:



9.4

Attempted CSV modeling using only ArcGIS Online

9.5

Attempted CSV modeling using only ArcGIS Online (Zoom-In)

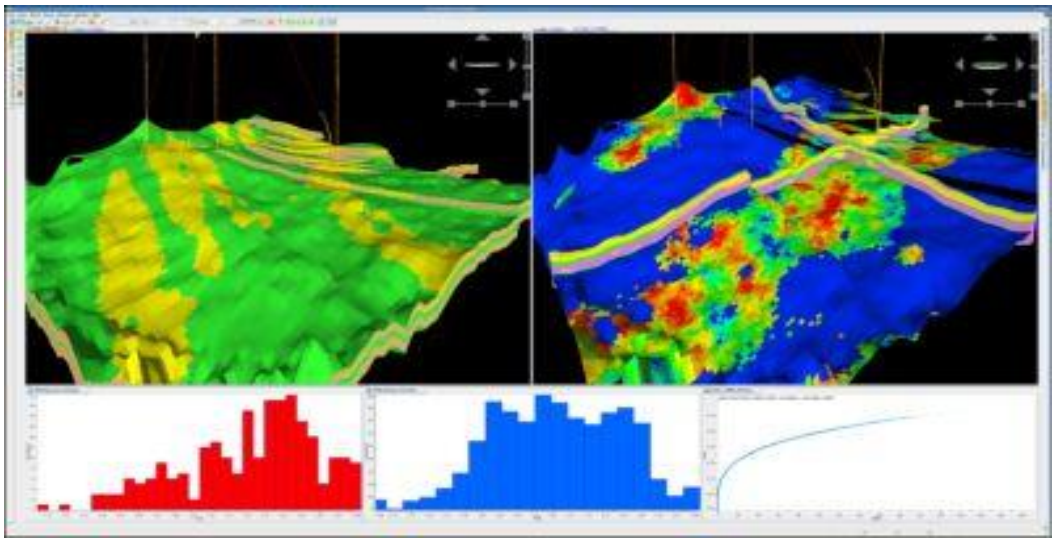
9.6

Output (CSV) of First Implementation of Parser

```
latitude,longitude,value
49.66664960000001,-110.2500235,-0.004
49.66664960000001,-109.4166905,-0.004
49.66664960000001,-109.3333572,-0.004
49.66664960000001,-109.2500239,-0.003
49.66664960000001,-109.16669060000001,-0.003
49.66664960000001,-109.0833573,-0.003
49.66664960000001,-109.000024,-0.003
49.66664960000001,-108.9166907,-0.003
49.66664960000001,-108.83335740000001,-0.003
49.5833163,-110.41669010000001,-0.004
49.5833163,-110.3333568,-0.006
49.5833163,-110.2500235,-0.004
49.5833163,-110.1666902,-0.004
49.5833163,-110.0833569,-0.004
49.5833163,-110.00002359999999,-0.006
49.5833163,-109.9166903,-0.004
49.5833163,-109.833357,-0.004
49.5833163,-109.7500237,-0.003
49.5833163,-109.6666904,-0.003
49.5833163,-109.5833571,-0.003
49.5833163,-109.50002380000001,-0.003
49.5833163,-109.4166905,-0.003
49.5833163,-109.3333572,-0.003
49.5833163,-109.2500239,-0.003
49.5833163,-109.16669060000001,-0.003
49.5833163,-109.0833573,-0.003
49.5833163,-109.000024,-0.003
49.5833163,-108.9166907,-0.003
49.5833163,-108.83335740000001,-0.003
```

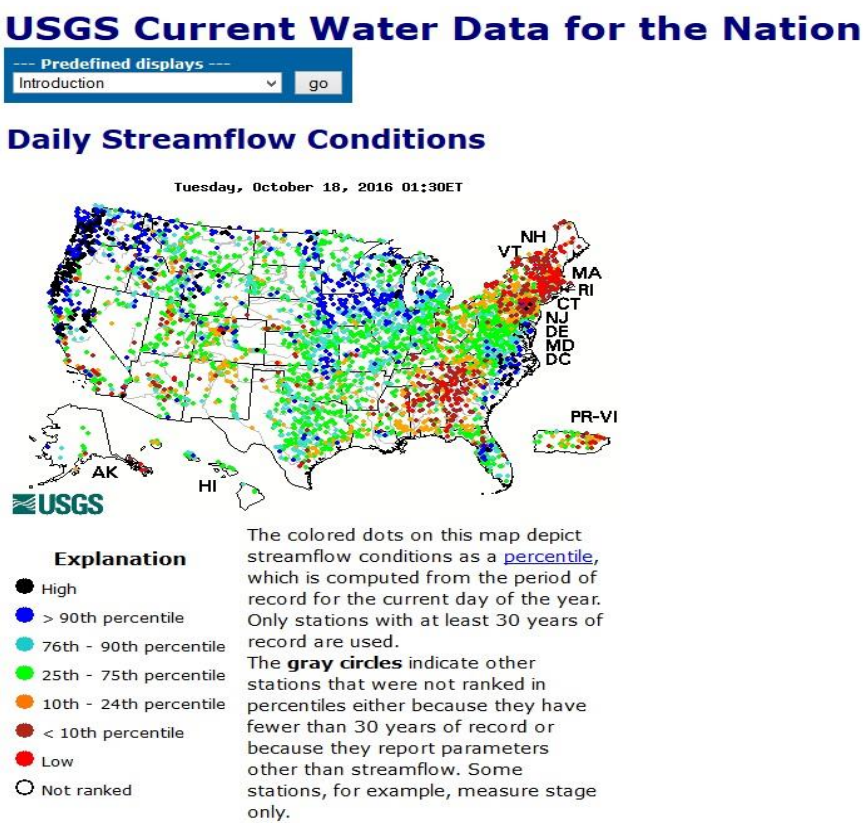
9.7.1 (1)

Halliburton's Landmark DecisionSpace Visual Example. (World Oil):



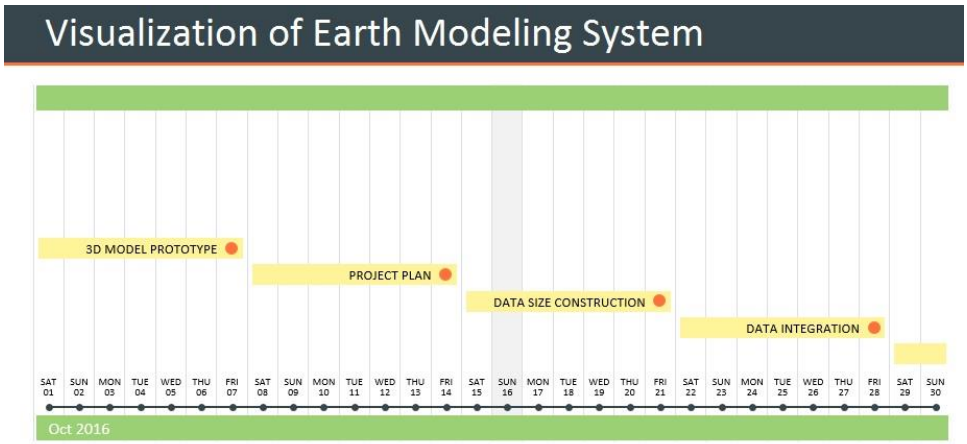
9.7.2 (2)

USGS streamflow data. (USGS):

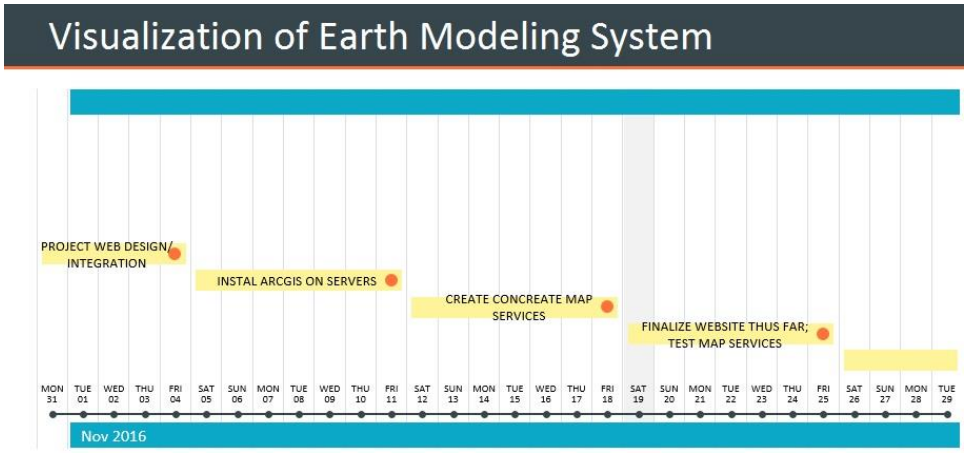


9.8

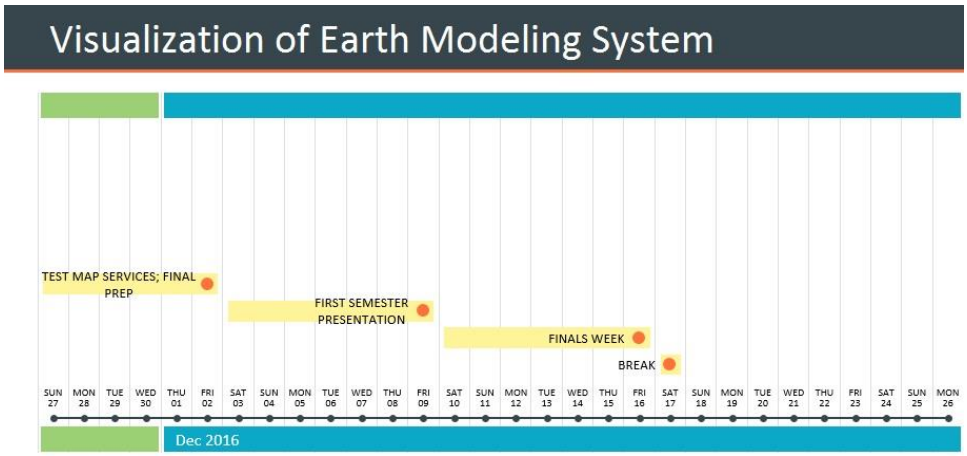
First Semester Timeline:



(a) October 2016



(b) November 2016

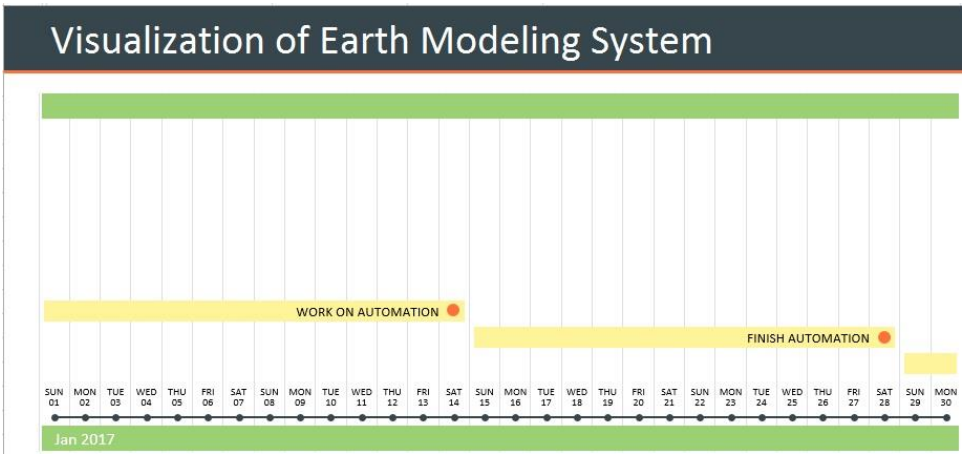


(c) December 2016

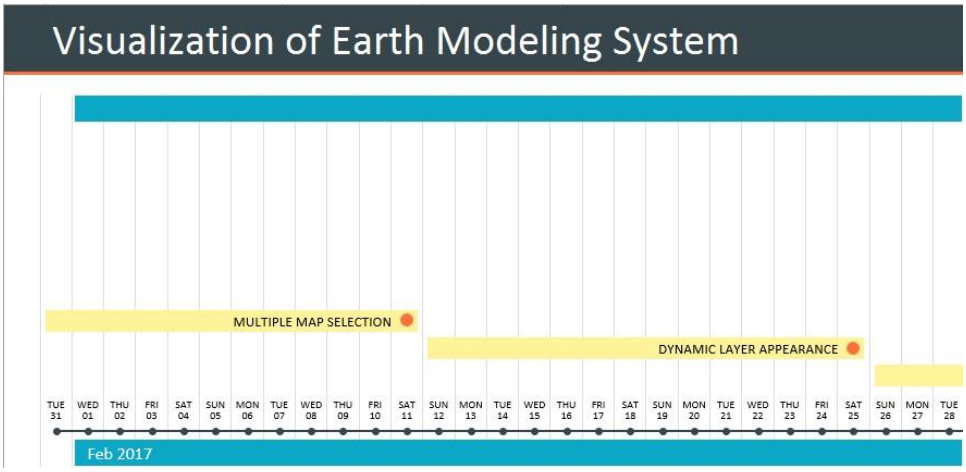
ACTIVITY	START	END	NOTES
3D model prototype	10/1/2016	10/7/2016	October: Week 1 - Began testing with arcGIS to become familiar with this program.
Project Plan	10/8/2016	10/14/2016	October: Week 2 - Write up the project plan and steps needed to complete the project. What needs to be done by the end of the first and second semester.
Data size construction	10/15/2016	10/21/2016	October: Week 3 - Determine what size to represent each value from ASCII table in order to represent a well scaled data point.
Data Integration	10/22/2016	10/28/2016	October: Week 4 - Begin to plot the points, applying the values from a 2D data set onto a 3D plane.
Project Web Design/ Integration	10/29/2016	11/4/2016	November: Week 1 - Integrate the 3D mdoel to the current project web page with data set prototypes. Create smooth transitions.
Instal ArcGIS on Servers	11/5/2016	11/11/2016	November: Week 2 - Installed ArcGIS onto your servers, begin configuration.
Create Concreate Map Services	11/12/2016	11/18/2016	November: Week 3 - Plan to create concrete map service examples.
Finalize Website thus far; Test Map Se	11/19/2016	11/25/2016	November: Week 4 - Finalize website, test map services and fix performance and stability.
Test Map Services; Final Prep	11/26/2016	12/2/2016	December: Week 1 - Continue teseting and prepare for final presentation.
First Semester Presentation	12/3/2016	12/9/2016	December: Week 2 - Give First Semester's work Presentation.
Finals Week	12/10/2016	12/16/2016	December: Week 3 - Finals Week.
Break	12/17/2016		End of Semester

(d) First Semester Summary

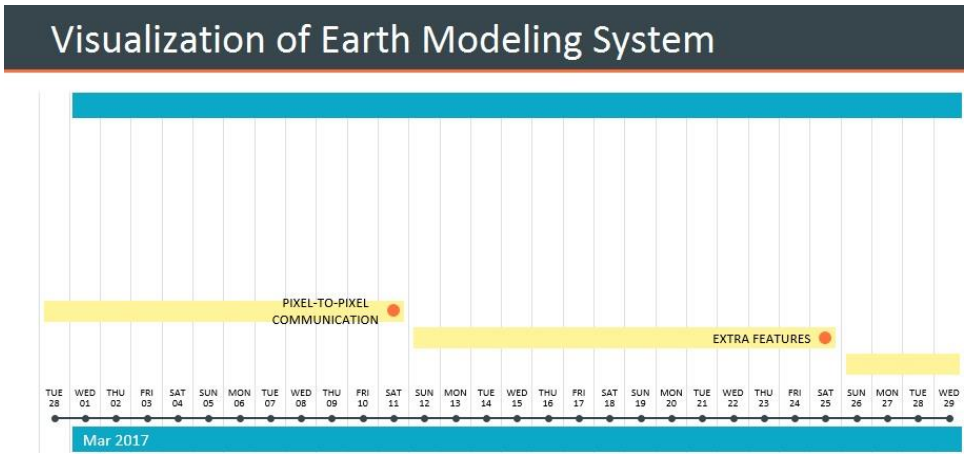
9.9 Second Semester Timeline:



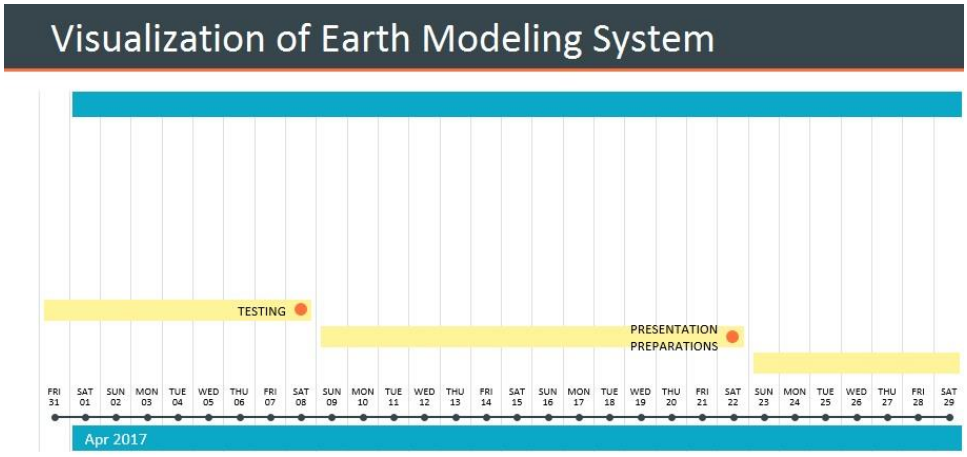
(a) January 2017



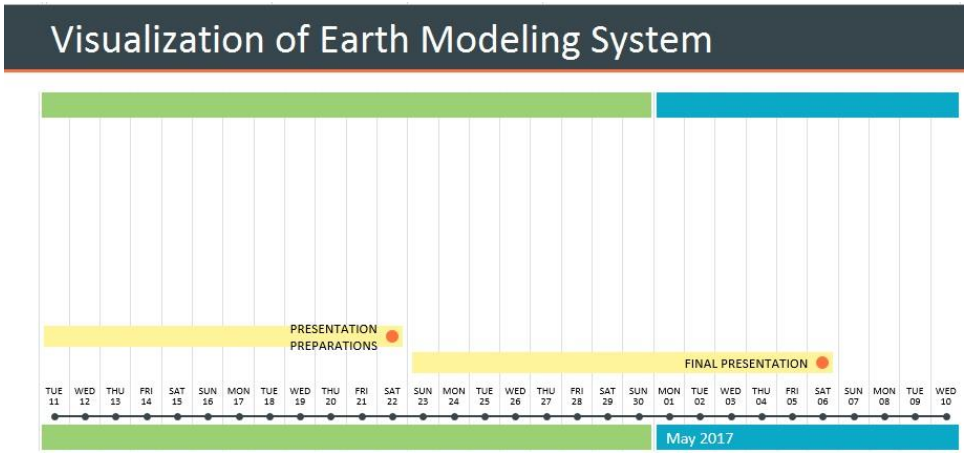
(b) February 2017



(c) March 2017



(d) April 2017



(e) May 2017

ACTIVITY	START	END	NOTES
Work on Automation	1/1/2017	1/14/2017	January: Week 1-2 - Regroup with the team and pick up where we left off.
Finish Automation	1/15/2017	1/28/2017	January: Week 3-4 - Finishing on the automation of the creation process
Multiple Map Selection	1/29/2017	2/11/2017	February: Week 1-2 - Create and layer different mappings onto a single 3D space and select which to display.
Dynamic Layer Appearance	2/12/2017	2/25/2017	February: Week 3-4 - Allows users to dynamically change layers appearance in the map services we create.
Pixel-to-Pixel Communication	2/26/2017	3/11/2017	March: Week 1-2 - Create an interactice flow of pixel communication for smoother data points.
Extra Features	3/12/2017	3/25/2017	March: Week 3-4 - Come up with extra features if possible.
Testing	3/26/2017	4/8/2017	April: Week 1-2 - Test project and web design on public computers to see if everything works fine and users are capable of freely working on it.
Presentation Preparations	4/9/2017	4/22/2017	April: Week 3-4 - Preparations include, finsihed project, documentation, presentation prep.
Final Presentation	4/23/2017	5/6/2017	May: Week 1 - End of Semester, present project

(f) Second Semester Summary

9.10