

Visualization of Earth Modeling Systems

Team MAY1701

Eli Devine, Kellen Johnson

Anish Kunduru, Julio Salinas

Faculty advisor: Dr. Johnny Wong

Project Background

- The Client
 - Professor Chaoqun Lu - Ecology, Evolution, and Organismal Biology Department at Iowa State
- The Research
 - Historical and projected compound emissions & uptake.
- The Scope
 - Create a tool which allows interested parties to view our client's data in an interactive user-space.

Project Significance

- Human influence on the climate is a widely accepted scientific theory, but it can be difficult to illustrate that to the people who can change policy.
- Increase awareness of climate change and assist policymakers and key stakeholders in related decision making.
- Our project provides a way to visually understand changes to the ecosystem as they occur over time.

The Problem

- Large Scale Datasets
 - > 1,200,000 points in a table
 - > 49,000 points to be rendered
- Slow Map Generation Times
- 2D Imaging
- The Project:
 - Automate the process.
 - Make the experience interactive.



Initial Project Plan

- Allow plotting of data in an interactive 3D space.
- End users should be able to dynamically select which dataset they wish to view.
- The solution must be platform independent and shouldn't require the installation of local software for end users.
- The process to add additional datasets should be automated.

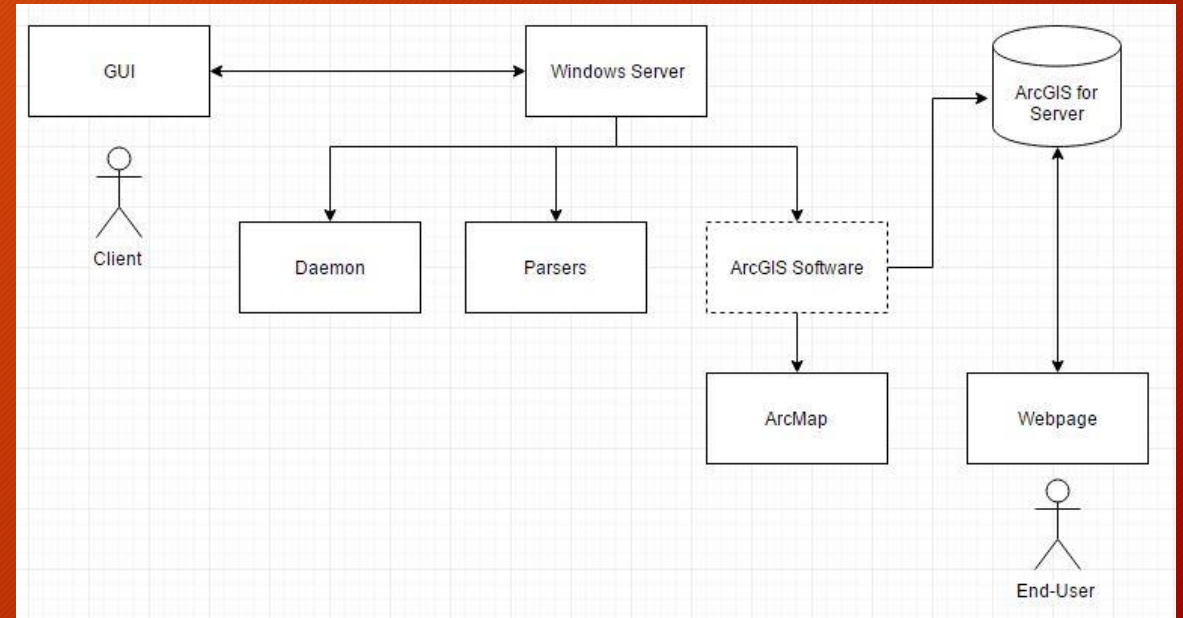
Initial Prototype: Client-Side Rendering

- Idea: Dynamically load data; render completely client-side.
- Problem:
 - Dataset too large for client-side rendering.
 - Interacting with the map causes the entire dataset to re-render.



Revised Project Plan

- Parse the ASCII data into a spatial geodatabase and host a map service.
- Load maps in any web browser using JavaScript.
- Provide a GUI to allow the client to enter her data easily.
- Create a daemon that handles all automation steps.



Requirements: Functional

- 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 ArcMap).
- The product shall allow end-users to dynamically select and view any generated map.

Requirements: Non-Functional

- The product shall convert raw data uploaded by the client to an ArcGIS readable format within 15 seconds.
- After conversion of a dataset, the product shall automatically generate a map service within 10 minutes.
- The product shall make a selected map service available to an end-user within 30 seconds.
- The product shall ensure that an end-user can differentiate between various points through color and/or size.
- The product shall support a minimum of 8 simultaneous users across platform, 2 of whom can be simultaneously accessing the same map.

Parsing the Data

- First, convert to a format readable by the ArcGIS API.
- Input: .txt file
- Output: .csv file

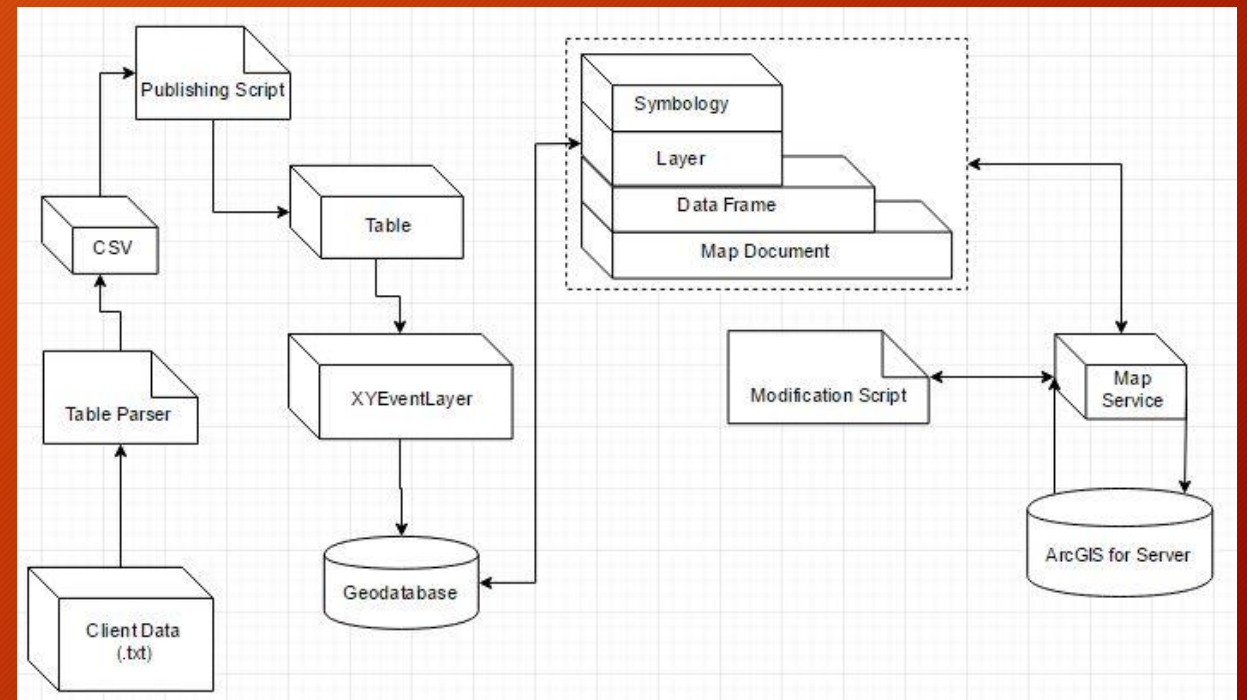
```
1  ncols      1404
2
3  nrows      924
4
5  xllcorner  -169
6
7  yllcorner   7
8
9  cellsize   0.0833333
10
11 NODATA_value -9999
12
13  -9999.000  -9999.000  -9999.000  -9999.000
    -9999.000  -9999.000  -9999.000  -9999.000
    -9999.000  -9999.000  -9999.000  -9999.000
    -9999.000  -9999.000  -9999.000  -9999.000
```



```
1  latitude,longitude,value
2  33.333322800000005,-101.3333604,-0.038
3  35.5833219,-90.3333648,2.696
4  36.333321600000005,-88.0833657,-0.029
5  35.9166551,-89.833365,2.614
6  35.999988400000001,-90.0833649,2.471
7  35.999988400000001,-90.3333648,2.454
8  36.083321700000006,-90.1666982,2.418
```

Generating the Map Service

- Convert the CSV into a spatial geodatabase.
- Using a template map, create a new map and link the geodatabase.
- Publish the map to the ArcGIS Server as a map service.
- Problem: Read-only values in ArcPy.



Early Idea: Cached Maps

- Idea: To maximize performance, we can cache the data contained within map services. This is how existing mapping products typically work.
- Problem:
 - Size, scaling levels, and number of maps the product must support.
- Solution: dynamic map services
 - CPU & memory usage vs storage requirements.

The Daemon

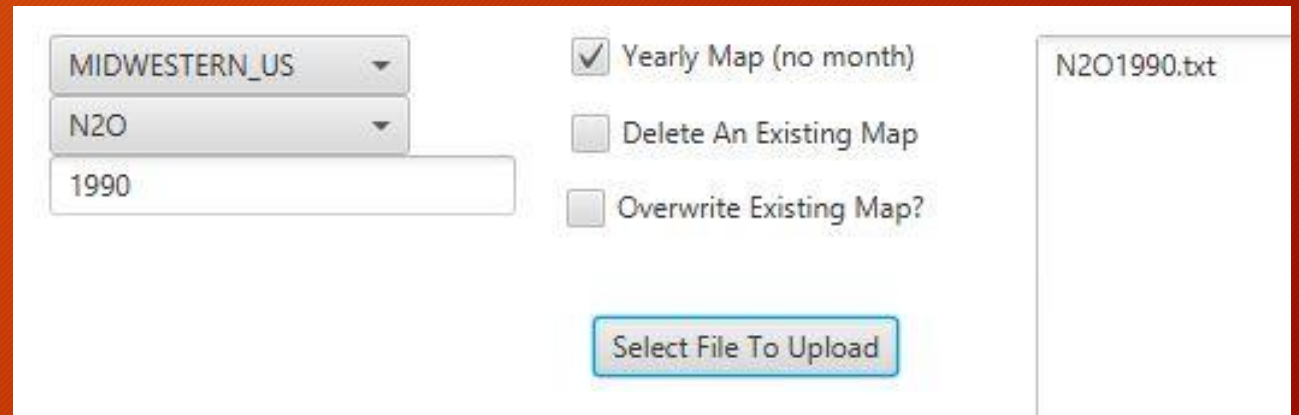
- Function: Handle communication with the client GUI and subsequent calls to automatically generate and delete maps.
- Language: Java
- Best practices:
 - Log critical output for use in debugging.
 - Strongly parse client inputted data and return validity.
 - "Clean code":
 - Readability, reuse, and testing.

How It Works: The Daemon

- Installed on the server as a Windows service.
- Handles individual parsing scripts and ArcPy component calls.
- Keeps track of all maps that have been converted and uses that to generate a JavaScript file that backs the front-end website.
 - JavaScript is minified and sent to server.
 - Faster and simpler than end-users querying another database.
 - Opens up the possibility of clients caching map tiles for future offline use.
- Talks to the client GUI via our networking framework; encrypted by TLS.

The GUI

- Function: Allow easy automation for the client and ensure well-defined input to the daemon.
- Language: Java (JavaFX)
- MVC Design
- User Operations:
 - Manually publish or delete maps.
 - Automatically publish maps.
 - View server logs.
- Deemed necessary due to:
 - Client ease of use.
 - Guard against invalid parameters being passed to daemon scripts.

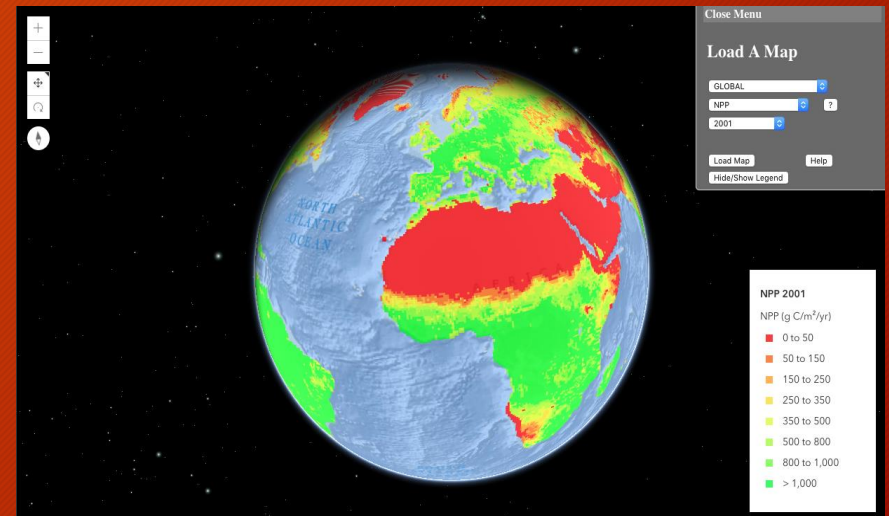


The screenshot shows a GUI interface with the following elements:

- A dropdown menu with "MIDWESTERN_US" selected.
- A dropdown menu with "N2O" selected.
- A text input field containing "1990".
- Three checkboxes:
 - Yearly Map (no month)
 - Delete An Existing Map
 - Overwrite Existing Map?
- A button labeled "Select File To Upload".
- A text field on the right containing "N2O1990.txt".

The Webpage

- Function: Display generated maps.
- Languages: HTML & JavaScript
- Loads our hosted map services via RESTful API.
- Fulfills our client's key goal of educating the public about human driven climate change.
- Took feedback for design improvements.



Testing

- Speed
 - Parsers
 - Webpage reactivity
- Accuracy
 - Verified generated map services against existing 2D images.
- Usability Testing
 - Does our intended audience find the product easy to use?
- Stability
 - Monitored server hardware usage.
- Debugging
 - Logging critical subroutines.

Demonstration: VEMS Webpage

Link to end-user webpage:

<http://may1701.sd.ece.iastate.edu/VEMS/VEMS.html>

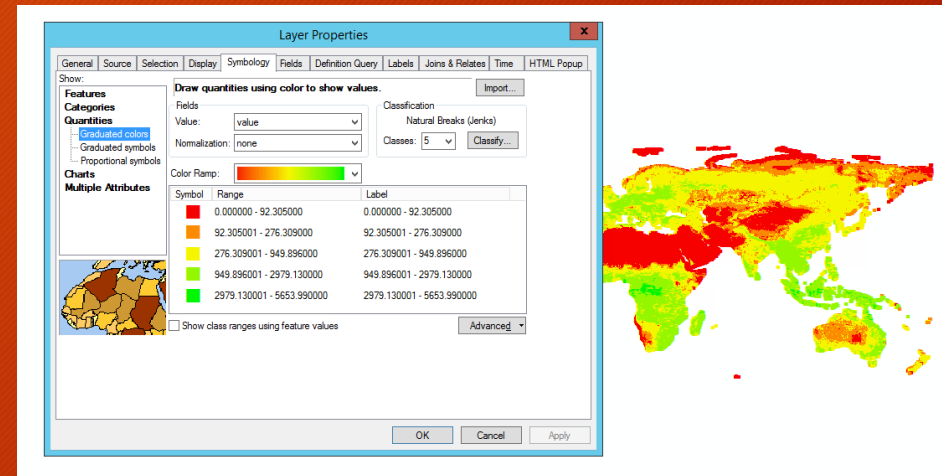
Future Development

- Multi-thread code.
- Website can support timeline maps (animations).
- Symbology and template maps generated in code.
- Regions and compounds can be added without recompiling daemon.
- Daemon can support multiple users.

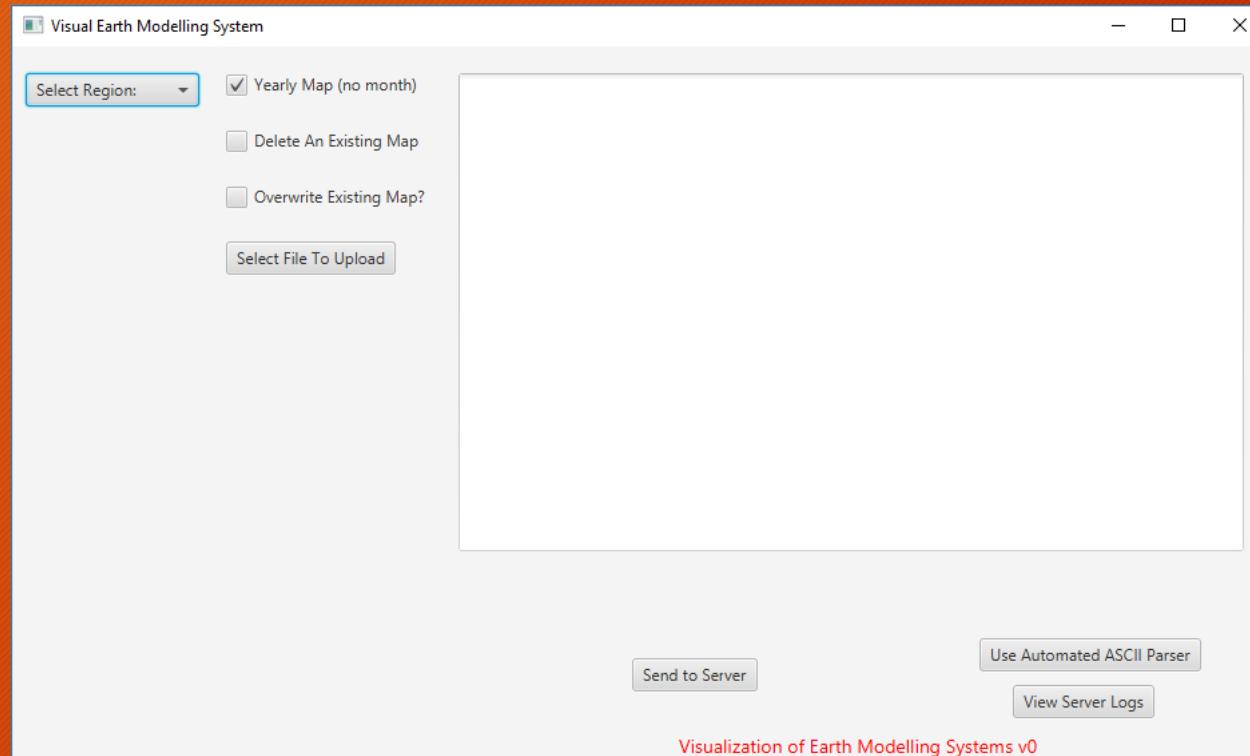
Questions?

How do we create a Template Map?

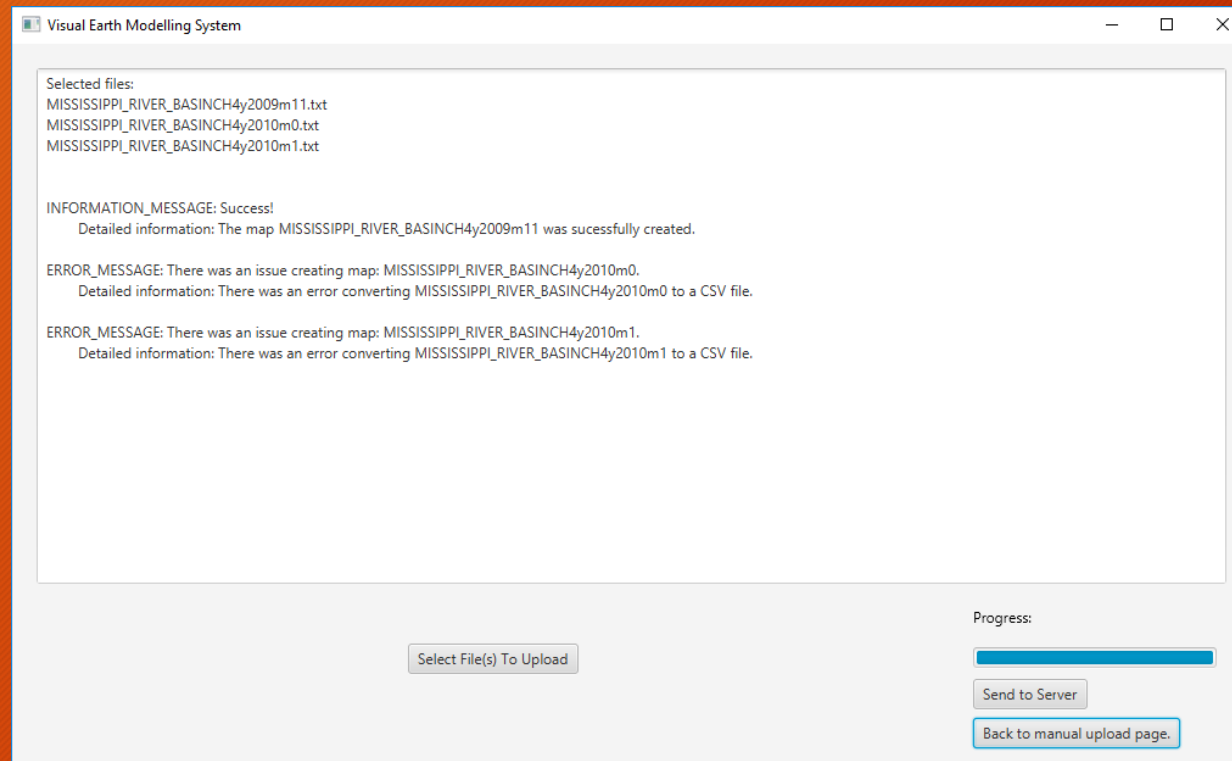
- We've provided our client with sample files to edit.
- Process summary:
 - Open a file in ArcMap.
 - Modify the name of the layer (appears in the legend).
 - Modify the symbology:
 - Color ramp (map colors)
 - Number of legend classes
 - Range & labels
 - Modify the reference scale within the data frame.
 - Publish the map to the server.



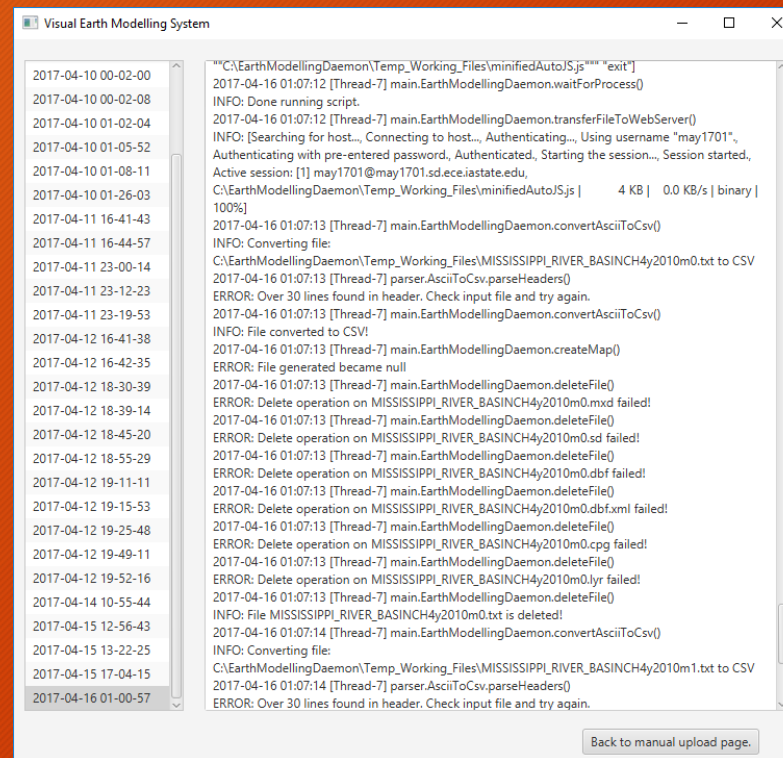
GUI Screenshot: Single Upload Page



GUI Screenshot: Multi-UL Page



GUI Screenshot: Logs Page



Visual Earth Modelling System

```
2017-04-10 00-02-00 ""C:\EarthModellingDaemon\Temp_Working_Files\minifiedAuto\$.js"" exit"]
2017-04-10 00-02-08 INFO: Done running script.
2017-04-10 01-02-04 2017-04-16 01:07:12 [Thread-7] main.EarthModellingDaemon.transferFileToWebServer()
2017-04-10 01-05-52 INFO: [Searching for host..., Connecting to host..., Authenticating..., Using username "may1701",
2017-04-10 01-08-11 INFO: Authenticating with pre-entered password, Authenticated, Starting the session..., Session started,
2017-04-10 01-26-03 Active session: [1] may1701@may1701.sd.ece.iastate.edu,
C:\EarthModellingDaemon\Temp_Working_Files\minifiedAuto\$.js | 4 KB | 0.0 KB/s | binary |
100%]
2017-04-11 16-41-43 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.convertAsciiToCsv()
2017-04-11 16-44-57 INFO: Converting file:
C:\EarthModellingDaemon\Temp_Working_Files\MISSISSIPPI_RIVER_BASINCH4y2010m0.txt to CSV
2017-04-11 23-00-14 2017-04-16 01:07:13 [Thread-7] parser.AsciiToCsv.parseHeaders()
2017-04-11 23-12-23 ERROR: Over 30 lines found in header. Check input file and try again.
2017-04-11 23-19-53 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.convertAsciiToCsv()
2017-04-12 16-41-38 INFO: File converted to CSV!
2017-04-12 16-42-35 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.createMap()
2017-04-12 18-30-39 ERROR: File generated became null
2017-04-12 18-39-14 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-12 18-45-20 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.mxd failed!
2017-04-12 18-55-29 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-12 19-11-11 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.sd failed!
2017-04-12 19-15-53 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-12 19-25-48 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.dbf failed!
2017-04-12 19-49-11 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-12 19-52-16 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.dbf.xml failed!
2017-04-14 10-55-44 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-15 12-56-43 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.cpg failed!
2017-04-15 13-22-25 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
2017-04-15 17-04-15 ERROR: Delete operation on MISSISSIPPI_RIVER_BASINCH4y2010m0.lyr failed!
2017-04-16 01-00-57 2017-04-16 01:07:13 [Thread-7] main.EarthModellingDaemon.deleteFile()
INFO: File MISSISSIPPI_RIVER_BASINCH4y2010m0.txt is deleted!
2017-04-16 01:07:14 [Thread-7] main.EarthModellingDaemon.convertAsciiToCsv()
INFO: Converting file:
C:\EarthModellingDaemon\Temp_Working_Files\MISSISSIPPI_RIVER_BASINCH4y2010m1.txt to CSV
2017-04-16 01:07:14 [Thread-7] parser.AsciiToCsv.parseHeaders()
ERROR: Over 30 lines found in header. Check input file and try again.
```

[Back to manual upload page.](#)

Timing Results

- Data Transformation (Client Data to CSV)
 - 1 to 5 seconds
- Entire Map Creation
 - 45 seconds to 1 minute
- Initial Map Load by End-User
 - 10 seconds