AtlasCMS: Final Report

Marshall Mattingly
Department of Computer Science,
University of North Dakota
Grand Forks, ND 58202
Email: marshall.p.mattingly@my.und.edu

Michael Marti
Department of Computer Science,
University of North Dakota
Grand Forks, ND 58202
Email: michael.marti@my.und.edu

Abstract

As North Dakota approaches its 125th anniversary of statehood, there is a wealth of demographic, economic, and social data available relating to the diverse religious, ethnic, and social groups of North Dakota. However, this data is scattered across many fields, including anthropology, history, religious studies, sociology, and American Indian studies, without any single source location to view multiple narratives from the many fields. Creating a content management system that allows students from these various fields to create digital interactive maps and graphs with accompanying narratives will allow students and the general public to learn about multiple aspects of North Dakota, creating connections between the multiple disciplines that may not have been apparent otherwise. By using a content management system that is integrated with common technologies used to create physical atlases, AtlasCMS allows the content managers to reuse created content, greatly simplifying the process of creating interesting and interactive online maps.

I. Introduction

There are several examples of interactive atlases or maps online, each with their own unique set of features. For example, Atlas of the Historical Geography of the United States [1] utilizes static images as maps, while allowing hiding of the legend, narrative, and table of contents, while also incorporating a slider and animation functionality for year-on-year map changes. The historical geography atlas is an excellent example of an interactive atlas, but struggles with dynamic data because of its use of static images. For a further evaluation of more online atlases, please see the Related Works section of this paper.

Another consideration is that the related works discovered appeared to be created and maintained by a web developer. A major goal of AtlasCMS is to create a content management system (CMS), instead of a hard-coded website, that allows for content managers, rather than web developers, to generate and modify the content. This also allows AtlasCMS to be expanded into other projects that wish to represent geographical data sets with an interactive user experience, elevating it above a single-use application.

This paper will examine several online atlases or interactive maps in the Related Works section, explaining how other projects overcame the transition from physical atlas to digital atlas. The approach taken during development of AtlasCMS, including considerations and rational for user interface (UI) elements, is defined in the Approach section. Along with the higher-level approach, the Implementation section provides a technical breakdown of the AtlasCMS project, including software design diagrams, algorithms, and specific application programming interfaces (APIs) and frameworks integrated into AtlasCMS.

The paper then transitions into the Results section, which documents how well AtlasCMS accomplishes it goals, primarily of usability for both users and content managers. The Future Work section provides ideas for progression of the project beyond what was completed. Finally, the Conclusion discusses the impact and importance of the results of AtlasCMS.

II. Related Works

During our search for related works, we focused primarily on websites that present geographic data over periods of time. This led to the discovery of several interactive elements that add to enhance the consumption of the maps, creating a clear benefit to a digital atlas compared to a traditional, physical atlas. We also noted how similar website handled providing basic map information, such as a legend and compass, allowing us to compare and contrast the different UI design decisions.

There were three main related works that we used when designing AtlasCMS: *i*) Atlas of the Historical Geography of the United States [1], an interactive atlas with basic animation functionality and narratives that covers many topics, including presidential elections, state boundaries, and other data; *ii*) Urban Layers: Manhattan's Urban Fabric [2], an interactive single-map that shows the buildings built in Manhattan during a dynamic, user-defined time period; and *iii*) Washington DC: Our Changing City [3], an interactive atlas with more advanced animation functionality and narratives that covers changes in demographics, schools, and housing in Washington DC.

A. Interactive Maps

Interactive maps that allow the user to easily compare similar maps over a period of time, both by animation and manual selection, is paramount to the digital experience of the atlas. The historical geography atlas uses hand-drawn maps, which appear to be taken from figures in a physical atlas, and places them atop a map element layer provided by Leaflet. For sections have multiple reference points, such as presidential elections, as slider that encompasses the year-span is provided at the bottom of the map, allowing the user to either slide to a specific year, or hit a play/pause button that pre-caches all of the images for each map within the section and ticks through each year automatically, allowing the user to pause on any specific map. [1] The slider is an intuitive UI element to show multiple maps that scales well to multiple viewports and gives responsive feedback while animating; however, the use of static maps limits the interactivity of the individual maps.

The Washington DC atlas primarily uses charts and static geographic maps to present data. However, there is an interactive map in the Housing section under the title ”Building boom transforming DC neighborhoods“ that automatically animates through several years and updates the map, presented via MapBox, automatically. By default, this map is not interactive; however, the user can press a button to allow a fully interactive version of the map to popup in a new window or tab. This interactive version allows the user to click a button corresponding to a specific year to pause the animation, and then resume the animation if desired. [3] The buttons are a simple and intuitive UI element to show multiple maps and gives somewhat responsive feedback while animating by highlighting the current button; however, buttons take up significant space, limiting the amount of data points possible, especially on smaller viewports.

The urban layers map is the least interactive of the maps in the related works, providing no means of animation. However, instead of having a slider or buttons that allow a user to select a single data-point, urban layers allows the user to select both a start and end data for the map, filling in the map with all corresponding data. [2] While the urban layers map does not have any animation functionality, since all data points in a given time-span are shown, the ability to select and start and end point using a slider is intuitive and allows for basic filtering by the user to limit the data points to those the user is interested in.

B. Interface Elements

There are many elements that make up a typical atlas, including: *i*) a legend to show define the icons on the map; *ii*) a narrative to describe what the map (or collection of maps) represents; *iii*) and a table of contents to allow easy navigation to the different sections or chapters in the atlas. The historical geography atlas starts with the legend displayed, hovering over the map in the bottom-right corner, and the narrative and table of contents hidden. There is a floating element in the top-right corner that allows the user to hide the legend and show the narrative or table of contents by clicking a check box. If the user checks either the narrative or table of contents, the respective elements pops in from the right-hand side, shrinking the map to allow the new element to fit. [1] Using the master-element and checkboxes are both intuitive, and allow the user to easily turn elements on and off to either view a larger map or read the narrative.

The Washington DC atlas present all of its data with the narrative on the left and any visual element on the right using a two-column approach whereby the visual element takes up more space than the textual element. Legends are presented on the visual element in the top-right corner and cannot be hidden. There is no full table of contents; however, the user is able to click a box in the top-middle of the atlas to change chapters or pages within a chapter. This atlas is obviously more inspired by traditional atlases, presenting in a static format, which hinders the interaction within the website by limiting how the user can consume the data and, especially, making it difficult to select a specific page from any other page.

III. Approach

While there are several examples of online atlases, as described in the related works section, there is no common content management system for online atlases that allows for interactive elements to be created by non-technical persons. Therefore, aside from the usability for users reading the atlas, a primary consideration is the usability of the content management system.

To simplify the programming and provide a familiar interface, we will utilize several robust, well-tested open source web application programming interfaces (APIs) and frameworks. On the client-side we will use Bootstrap and JQuery. Bootstrap is a widely used CSS and JavaScript framework that allows for simple and consistent stylization over multiple viewports, from phones to desktops. While there are no current plans to create a phone interface, using Bootstrap will allow one to be developed without any additional modification by content managers. JQuery is a JavaScript framework that provides an easy way to access and modify elements on the webpage dynamically and allow AJAX functionality to update data, as needed.

On the server-side, we use Node.js as the package manager for all of the elements of our server. This allows us to use JavaScript, a familiar programming language for both primary developers, to handle all website requests and integrate other JavaScript projects into our project, such as ¡BLANK¿ for the actual web requests and ¡BLANK¿ for the web page template system. This allows for more rapid development of the web pages themselves without needed to focus on complex server configurations.

For the design we incorporated several elements from each of the related works and refined them into a new UI that focuses on multiple viewports. There is an upper navigation bar that allows for easy access of the different chapters and sub-chapters and scales well to multiple viewports. The current map takes up the rest of the available space, with a legend floating above the bottom-right of the map and a slider, if appropriate, below the map, allowing the user to select a specific date for a series of maps. To the right of the map is a small column with an arrow pointing inward, toward the map; if the user clicks the arrow, the narrative pops out, automatically shrinking the map and flipping the arrow 180-degrees to indicate that clicking the arrow again will hide the narrative. While extremely small viewports, such as phones, haven't been fully developed, it is intended that the narrative pop-out would instead fill the screen, and allow the user to exit the pop out to again view the map.

The slider at the bottom of the page indicates that the map can be animated and is used for maps that change over time. The user can select a specific year, and the narrative and map will automatically adjust to the context of that year. The user can also click a play button to the left of the slider to automatically animate the map and narrative, as if the user had slowly moved the slider herself. If anything catches the user's eye, they can pause the animation at any time. Since the slider is keyed up with the map and narrative, as the user reads the narrative, jumping to a new section of the narrative, we can automatically move the slider and alter the map, ensuring that the map and narrative are always consistent.

While each of the related works cited have elements similar to AtlasCMS, we believe that by tying our maps and narratives together with the animation and slider, AtlasCMS allows for a more immersive experience. Additionally, developing with scalability and multiple viewports in mind allow AtlasCMS to remain functional on a wide variety of devices, rather than just desktops.

To allow non-technical content managers to maintain the system, a simple management system with user authentication and administration was created. The administration system allows administrators (users with a higher level of credentials) to create user accounts and groups, create new sections, and give, remove and modify rights to specific content managers or groups on specific sections. This basic functionality allows on or more persons to control rights to the content management system without the need for developer interactions.

Content managers can then log into the admin portal and add, delete, or modify pages within sections for which they have rights. When a user creates a new map, they are able to specify whether or not the map is static or dynamic, which means that the map covers multiple years which will be animated. The user then defines the name of the map, URL of the map, inserts the narrative (in one or multiple sections), and creates the map-narrative key points that ensure the map and narrative remain in sync when the map is dynamic. The key points are then further refined to allow the user to define which layers are turned on and off when the key point is triggered, allowing the map to actually perform the updates. Content managers can also modify any of the features at any time.

IV. Implementation

This section is still in development. Current documentation pending to change.

Originally, development for AtlasCMS started as an extension of DjangoCMS, a content management system written in Django, a Python framework. However, given the time-constraints and unfamiliarity with writing modules in DjangoCMS, it was decided that creating a basic CMS from the ground up using more familiar frameworks would allow for a faster and more consistent development experience. Therefore, Node.js, a JavaScript package manager, is used to integrate several JavaScript packages into AtlasCMS, including a basic web server, database APIs, and templating system.

One of the primary concerns during development was ensuring that the content managers would be able to reuse resources they have already created using ArcGIS, a common geographic data creation tool. During early development, the idea of converting the ArcGIS data to the open source GeoJSON standard was tested. However, during the conversion, only the basic geometric shapes and user-defined variables for the shapes were retained, removing extremely important features such as symbology for legends and colors to distinguish shapes. To use GeoJSON from ArcGIS data would require the content manager to upload the ArcGIS data and then edit the GeoJSON, in a tool that would have to be developed and integrated into AtlasCMS, creating a sizable increase in work load and adding error potential to the content.

To avoid have content managers reenter data in GeoJSON format that has already been created in ArcGIS, it was decided to use connections from AtlasCMS to ArcGIS Server to render the maps. ArcGIS Server is a standalone ArcGIS project that allows content managers to upload their ArcGIS data, with all defined layers, symbology, and legends, to be rendered by the server. External sources, such as AtlasCMS, can then send requests to ArcGIS Server using the JavaScript ArcGIS API to embed the map, including any layer filtering, on webpages. Using ArcGIS Server to render the maps simplifies the process for content developers who are already using ArcGIS dramatically; however, this is an additional cost that must be incurred, and developing support for GeoJSON is a future goal of AtlasCMS to allow it to be used in more projects.

Here we will discuss how we went about allowing the user to create map-narrative key points, and how we used those to filter the layers on the map with the narrative (and vice versa). This feature hasn't been fully implemented and will be added for the final draft.

Here we will discuss the authentication system in place, including basic access control and content management features. This feature hasn't been fully implemented and will be added for the final draft.

Here would be the high-level use-case diagram describing the system in full. Will be added for the final draft.

Here would be the database design. Will be added for the final draft.

V. Results

Results pending.

VI. Future Work

A major distinction of AtlasCMS is the ability to scale to multiple viewports. However, given time constraints, the interface and functionality was developed with tablets and laptops in mind. The next step in the design would be to implement the interface and functionality for phones, including rendering to a smaller viewport and more touch-enable features such as swiping.

While AtlasCMS is custom made for a situation in which an ArcGIS server is available, creating an interface that uses fully open standards such as GeoJSON and Open Maps, while allowing the same level of interaction would be ideal. This would allow AtlasCMS to be used be a wider variety of projects at zero cost. However, to incorporate this functionality, a specific set of expected variables in the GeoJSON must be defined by the program and created by the content managers, removing some of the ease-of-use of AtlasCMS. Optionally, a GeoJSON editor can be incorporated directly into AtlasCMS, allowing for the alteration of required variables to the provided data within AtlasCMS itself.

VII. Conclusion

Conclusion pending.

VIII. TODO

1) Expand the Introduction
2) Add additional citations for the different APIs, frameworks, and programs used
3) Complete the Implementation, Results, and Conclusion sections

References

[1] “Atlas of the historical geography of the united states,” http://dsl.richmond.edu/historicalatlas/, accessed: 2014-10-21.

[2] “Urban Layers manhattan urban fabric,” http://io.morphocode.com/urban-layers/, accessed: 2014-10-21.

[3] “Washington DC our changing city,” http://datatools.urban.org/features/OurChangingCity/, accessed: 2014-10-21.