Requirements

Totality AweSun

Bret Lorimore, Jacob Fenger, George Harder

November 4th, 2016

CS 461 - Fall 2016

1 Eclipse Image Processor

The eclipse image processor is the piece of our project that whandles the spatial and temporal alignment of the eclipse images that are uploaded to the Eclipse Megamovie website. We have identified three pieces [1] into which the image processor can be broken down. These are: image classification and manipulation, the runtime environment, and circle detection. In order for this element of our project to operate effectively it is critical that these three pieces utilize robust and functional technologies. This section of the technology review details what options are available for implementing each of these three pieces, analyzes these options, and arrives at a determination as to which option is the best in the context of this project.

2 Image Processor Manager

3 Eclipse Simulator

The eclipse simulator will be a standalone JavaScript module enabling users to preview the eclipse. It will be designed in a stylized, 2D manner. The simulator will incorporate a time slider to allow users to simulate the eclipse in a time window spanning from 12 hours before the eclipse to 12 hours after it. As users drag the time slider, the eclipse will animate in the simulator window. The view of the eclipse which users are presented will be specific to a location that the user enters. Additionally, the time will be displayed in the simulator based on what location the user enters and the positioning of the time slider.

3.1 User Interface

The user interface for this web based simulator will use 2D animated depictions of the Sun and the Moon as they appear at a user specified time and location. Additionally, the user interface will contain background imagery such as a city or hillside landscape. There will also be a time slider, a location input, and a time display for users to interact with or view.

TABLE 1
My caption

Method	Model	Method	Difficulty of Implementation	Performance
Implement from Scratch	Images	HTML, JavaScript, CSS	May take time to start from scratch	Dependent upon implementation, requires a lot of bandwidth to send large image files
Canvas	Pixel Based	JavaScript	Built into HTML 5	Smaller surface, larger number of items
SVG	Vector Based	JavaScript, CSS	Built into HTML 5	Larger surface, smaller number of items

For the first possible method, the most basic approach would be to utilize HTML, CSS, and JavaScript for output and input. Altering the locations of the sun and moon would be done by repositioning images of the sun and moon based on location data. While this approach can be simple, repositioning images and making the interface look appealing to the user can be difficult or inefficient.

Another method would be to utilize HTML5s Canvas graphics API. Canvas is capable of rendering graphics directly to the screen via JavaScript. This is otherwise known as immediate mode graphics, which means that the rendered graphics are not saved [2]. Additionally, Canvas draws individual pixels to the screen. For this simulator, the only image objects that we need display differently depending on user input are the Sun and Moon images. Since these are relatively small compared to rest of the simulator, re-rendering the Sun and Moon most likely will not result in poor performance.

The last technology for displaying the simulator would be to use another HTML5 component called Scalar Vector Graphics (SVG). This is a shape based method for describing 2D graphics via XML. SVG utilizes objects to describe images. The browser is capable of re-rendering shapes automatically if attributes to an object are changed [3]. Adding sliders or round buttons to the simulator can also be a straightforward approach with vector graphics.

For the user interface, speed is the most important criteria. All graphics and other simulator resources will need to load in less than 500ms given a 1-10 Mbps internet connection. Additionally the animation used in the simulator should be around 60 frames per second.

REFERENCES

[1] K. Larisza and S. McIntosh, "The standardisation and sequencing of solar eclipse images for the eclipse megamovie project."

[2] Microsoft, "Svg vs canvas: how to choose," msdn.microsoft.com/en-us/library/gg193983(v=vs.85).aspx, accessed: 2016-11-09.

[3] w3schools, "Html5 svg," www.w3schools.com/html/html5_svg.asp, accessed: 2016-11-09.