PERSPECTIVES DATA VISUALIZATION

FIU School of Computing & Information Sciences

Senior Project, 2013, Fall

Web-deployment of a Data Visualization Framework: Visualization Rendering and Interaction

Student: Erik Franco, Florida International University
Mentor: Dr. Radu Jianu, FIU **Instructor:** Dr. Masoud Sadjadi, FIU

PROBLEM

Data visualization customers are increasingly expecting web-accessible data visualization solutions. Unfortunately, browser technologies are still unable to cope with complex data analytics that require powerful algorithms, intricate graphics, and that can handle large amounts of data. There is currently a framework in place, called Perspectives, that provides the means to accomplish more efficient data visualization. Our task is to make the functionality found in Perspectives available on the web.

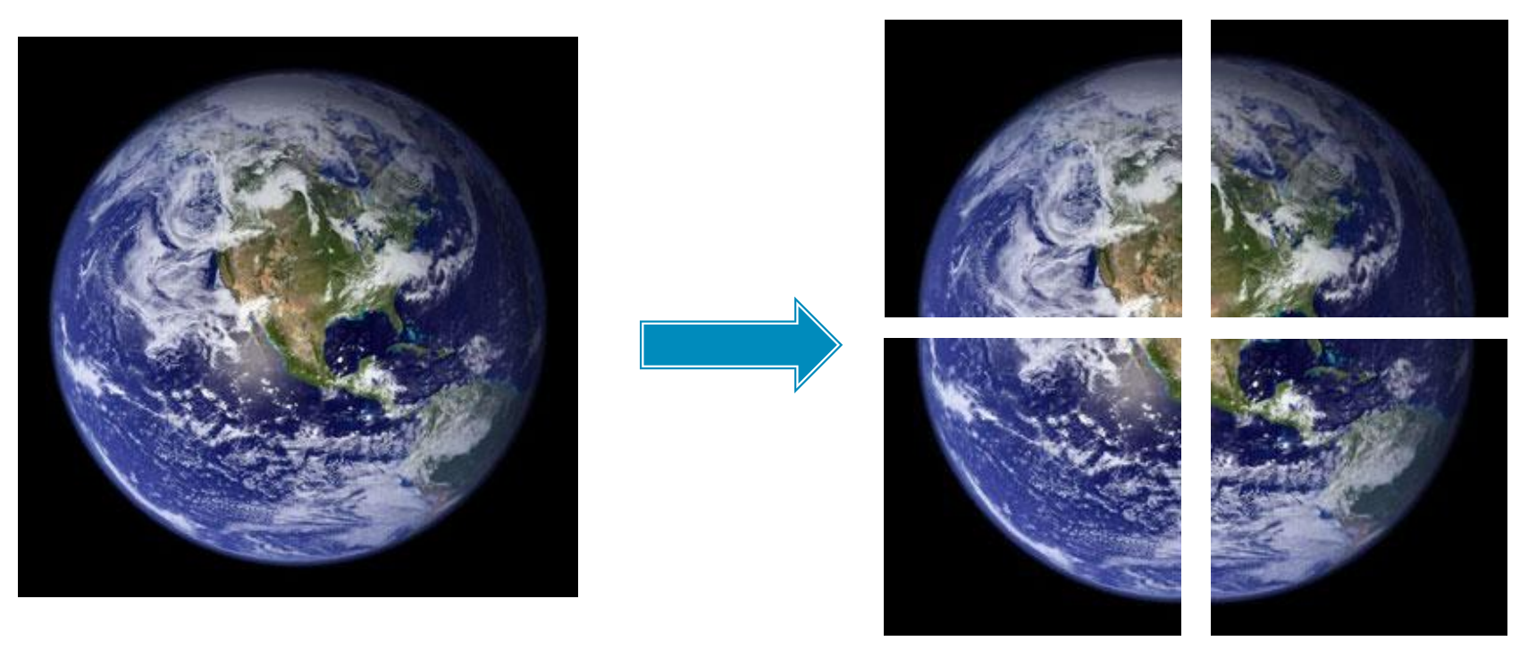
Computation and rendering will be done on the server within Perspectives and be relayed to the clients browser continuously as images. The user will be able to interact with these visualizations. The overhead associated with sending large images from the server to the client needs to be addressed in order to facilitate a smooth and responsive experience when interacting with the visualization.

CURRENT SYSTEM

The current system consists of a Java-based desktop application used for data visualization. It utilizes the Perspectives framework, developed by Dr. Radu Jianu. Users are able to upload data sets and choose from a given set of viewers, compatible with the uploaded data set type. The system then presents the user with an interactive interface, called viewers. Viewers allow the user to interact with the visualization.

SOLUTION

In order to implement a web environment that enables the Perspectives Framework functionality and mimics it's performance with respect to visualization interaction, a special image tiling algorithm was developed. The image was divided into 4 "chunks". Upon interacting with a specific subset of the image, only the chunk which is modified is returned back to the client.



This works to reduce the overhead of sending a large image to the client, thus increasing response time when interacting with the image.

REQUIREMENTS

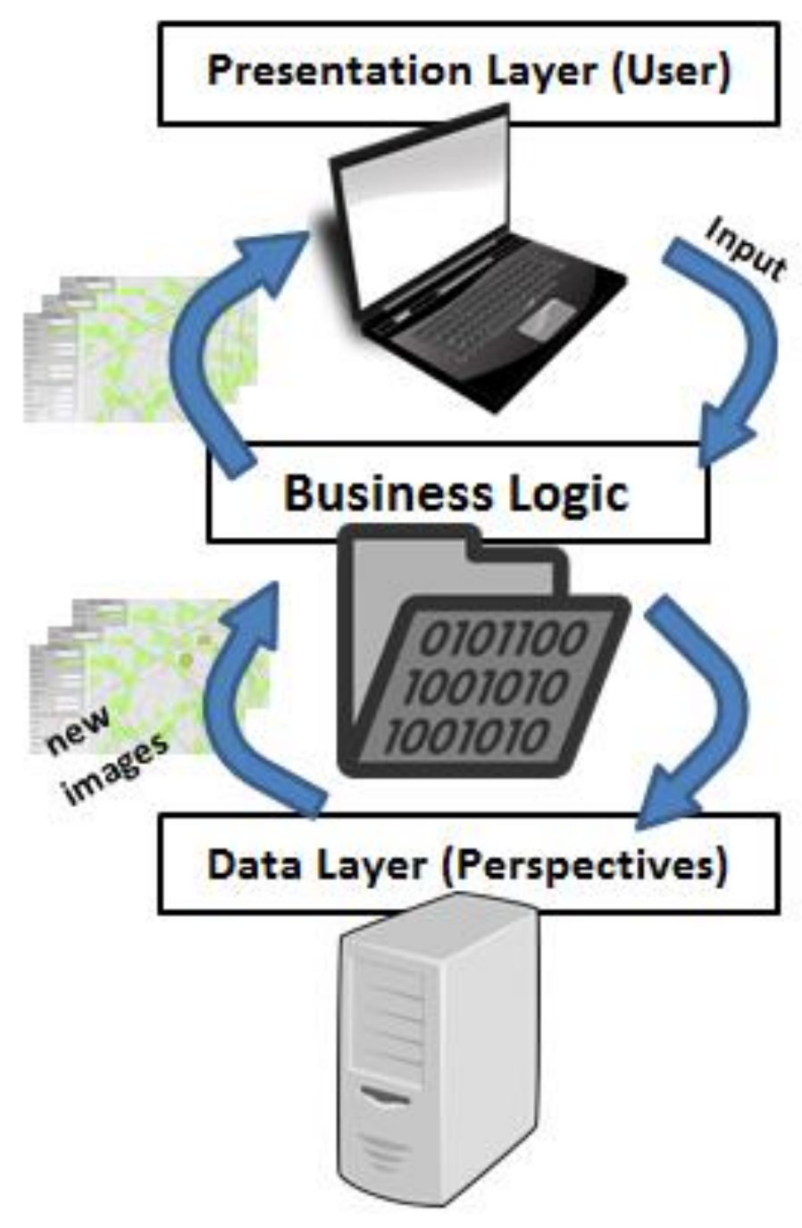
The general requirements for the system is to allow for similar functionality that is available in the desktop version of Perspectives, but made available through the web, handling most of the processing on the server and having the viewers only show images being sent continuously.

The specific requirements that relate to my role are as follows:

- Render Visualization
- Interact with Visualization

SYSTEM DESIGN

The online version of the Perspectives system was developed using a 3-Tier approach as shown below.



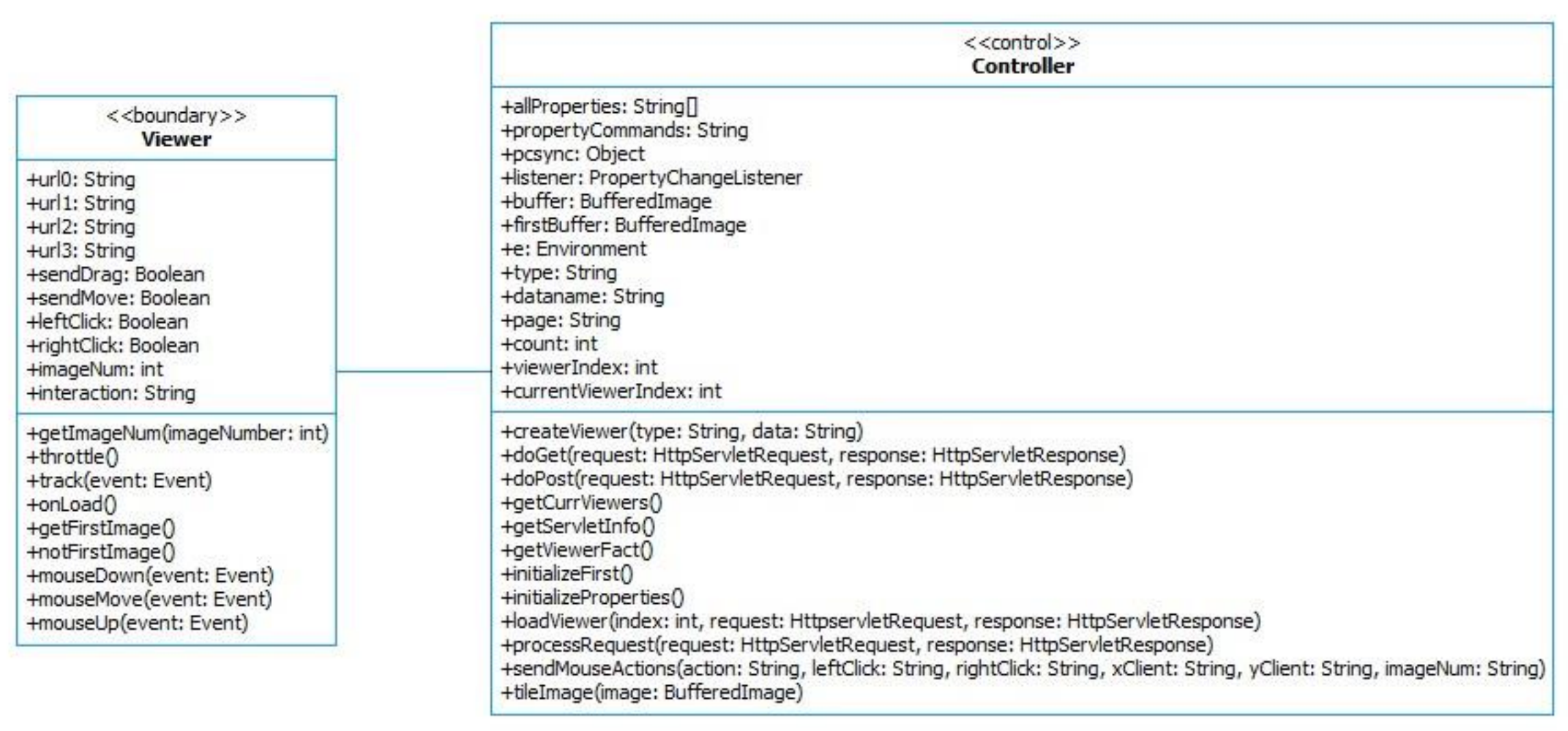
The Presentation layer represent the pages the user can see, along with client-side logic.

The business logic layer translates the information sent from either direction, along with some additional computation.

The Data Layer is the Perspectives Framework that, among other things provides the logic to create viewers, as well as creates new images based on the users interactions.

OBJECT DESIGN

The classes I was mainly focused on were the Viewer class and the Controller class.



The Viewer class is where all of the client side visualization interaction tracking occurred. Ajax functions were created that tracked mouse down, mouse up, drag, and mouse over. Also, to reduce the load on the server, a throttle method was introduced to limit the number of requests sent from the client to the server during image interaction.

The Controller class is responsible for controlling the communication between the Viewer class and Perspectives. When a new image is created by Perspectives, it gets sent to the tiling method which returns an array of sub images. This works together with the send mouse actions method to return only the modified sub image to the client.

IMPLEMENTATION

The system was implemented using Java, JavaScript, HTML and AJAX technologies.

Below is a snippet of the tiling method

```
    //Image array to hold image chunks
    BufferedImage imgs[] = new BufferedImage[chunks];
    for (int x = 0; x < rows; x++) {
        for (int y = 0; y < cols; y++) {
            //Initialize the image array with image chunks
            imgs[count] = new BufferedImage(chunkW, chunkH, image.getType());

            // draws the image chunk
            Graphics2D gr = imgs[count++].createGraphics();
            gr.drawImage(image, 0, 0, chunkW, chunkH, chunkW * y, chunkH * x, chunkW * y
                    + chunkW, chunkH * x + chunkH, null);
            gr.dispose();
        }
    }

    return imgs;
}
```

Below is a snippet of the method which sends actions to Perspectives and returns the image chunk that was modified

```
    // if mousedown is detected
    if (action.equals("mousedown")) {

        if (lClick)
            e.getViewerContainers().get(currentViewerIndex).mousePressed(x, y, MouseEvent.BUTTON1);

        if (rClick)
            e.getViewerContainers().get(currentViewerIndex).mousePressed(x, y, MouseEvent.BUTTON3);
    }

    // if drag event occurs
    if (action.equals("drag")) {

        if (lClick)
            e.getViewerContainers().get(currentViewerIndex).mouseDragged(x, y);

        if (rClick)
            e.getViewerContainers().get(currentViewerIndex).mouseDragged(x, y);
    }

    return imageNum;
}
```

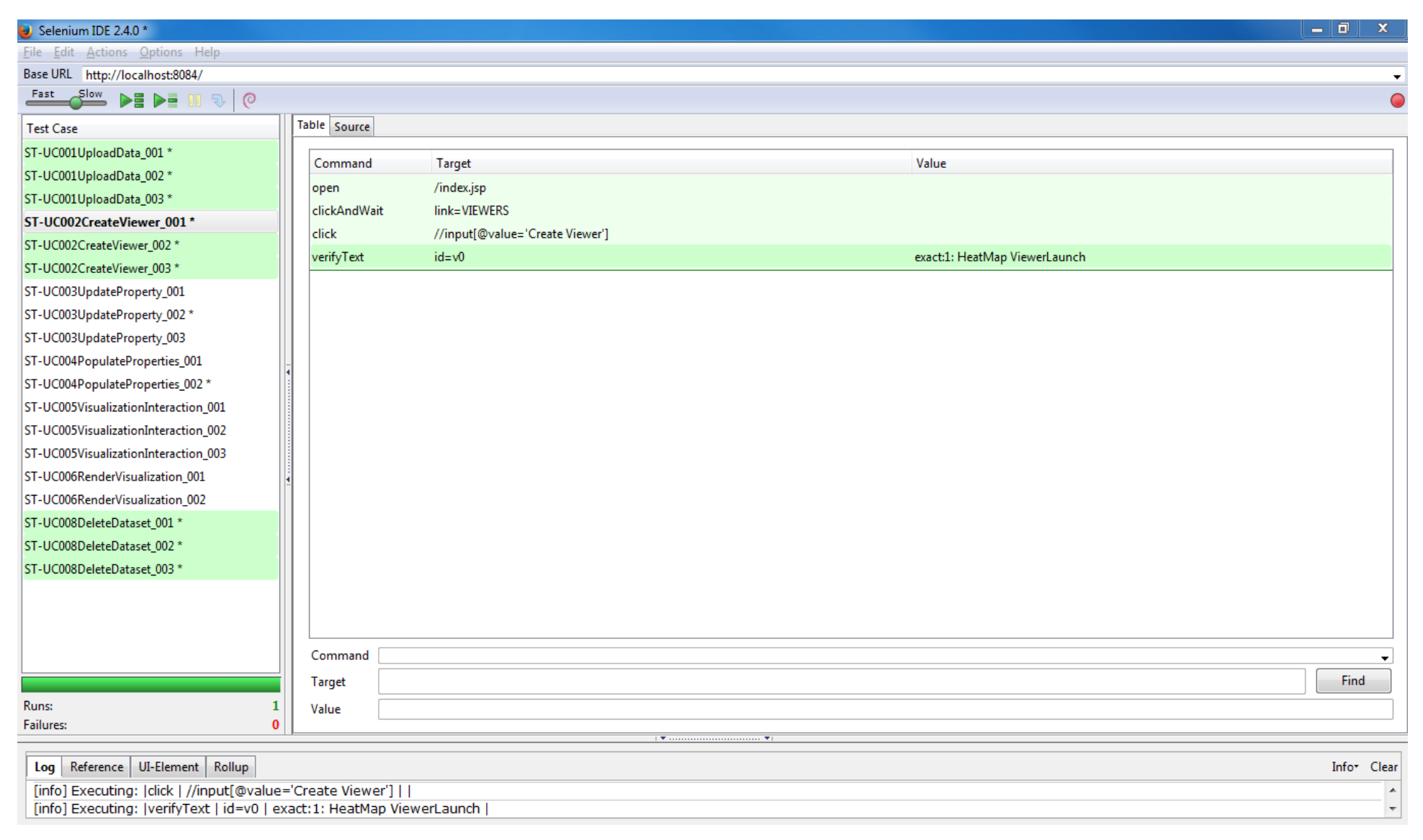
Below is a snippet of the method that controls whether to update a specific image chunk. This method works in conjunction with the previous two methods.

```
if(buffer != null){

    BufferedImage [] tiledImage = tileImage(buffer);

    if(tiledImage != null){

        // if no drag occured, update specific image chunk
        if(imageNum != -1){

            ImageIO.write(tiledImage[imageNum], "png", response.getOutputStream());
        }
        else{   //update all image chunks

            if(request.getParameter("slot") != null){

                ImageIO.write(tiledImage[Integer.parseInt(request.getParameter("slot"))], "png", response.getOutputStream())
            }
        }
    }
}
```

VERIFICATION

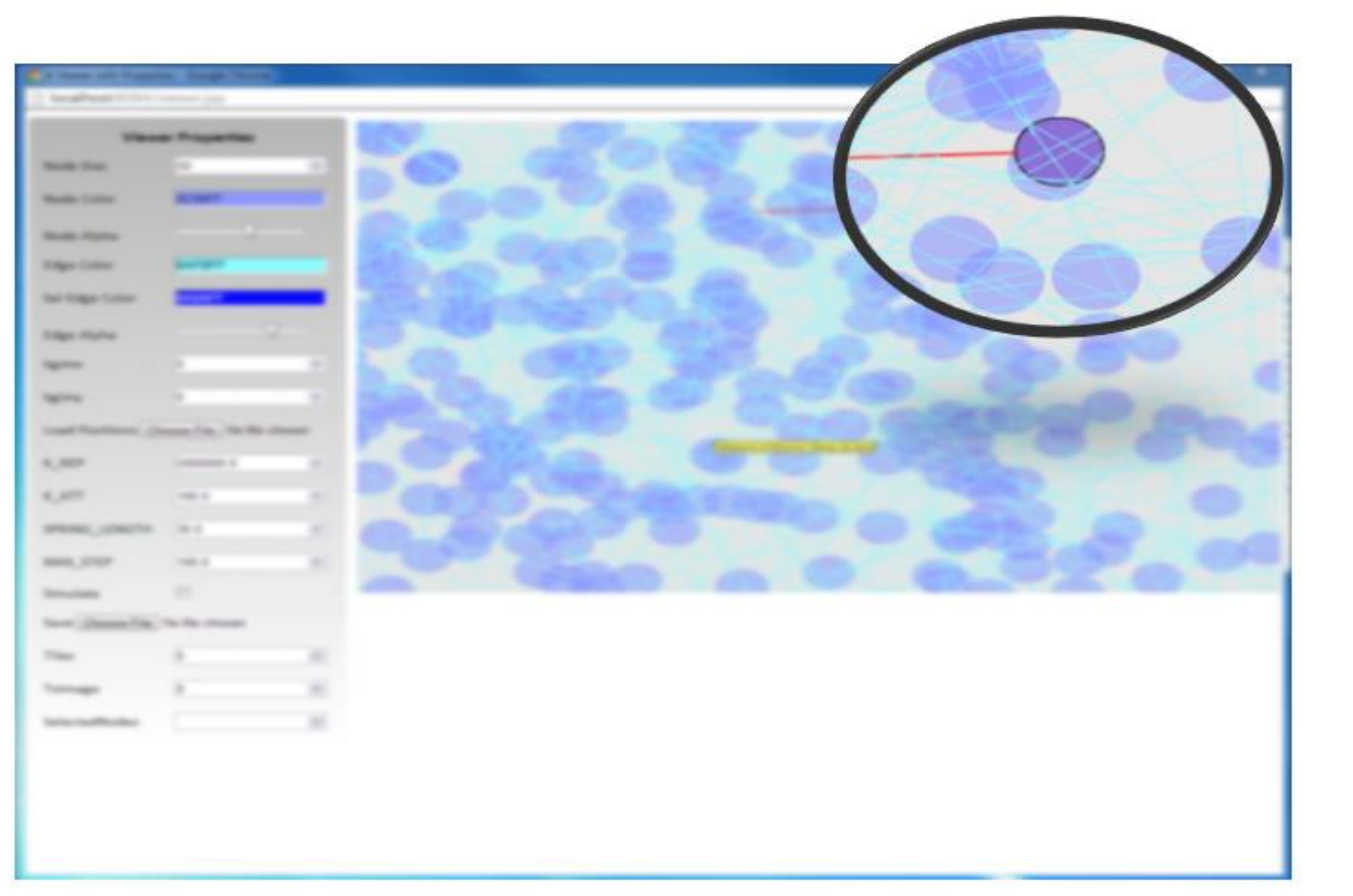
The system was tested using JUnit, Mockito, and Selenium.

Selenium was used to perform System Testing. The tests were conducted to validate each use case. Three test cases were done per use case. Two of the test cases tested sunny day scenarios, while the other tested a rainy day.

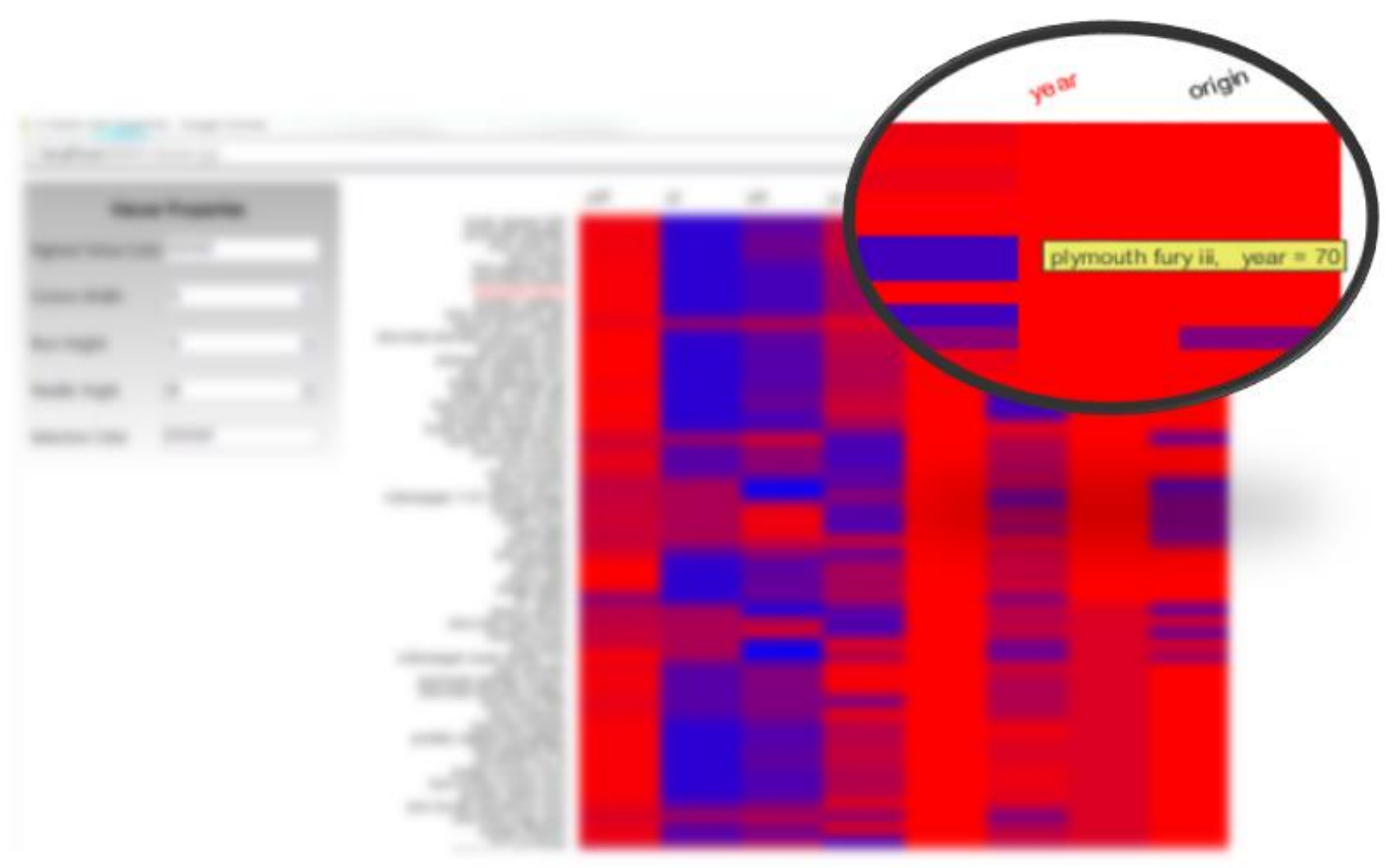


SCREENSHOTS

Example Graph Viewer, after a user has clicked on a node:



Example HeatMap Viewer, after a user has hovered over an element of the image:



SUMMARY

The new system that was created extends the Perspectives framework, developed as desktop application, by making it available online, while still having the interactive visualizations along with all of its functionality.

It relies on simple technologies available in most modern browsers in order to make it accessible to most people.

My focus was to allow for the visualized data to be displayed to the client, as well as to allow for the user to interact with the visualization; with special emphasis on performance to allow for responsive visualization interaction.

ACKNOWLEDGEMENT

I am thankful for the help that I received from my group members Raul Tobo and Santiago Pintos as well for the assistance provided by Mershack Okoe.