CS 6630 Final Project Proposal
Nate Morrical & Justin Jensen

Basic Info
Project Title: RayTracker
- Project repository: https://github.com/n8vm/RayTracker

Team:
- Nate Morrical
  - u1137457@utah.edu, UID: u1137457
- Justin Jensen
  - u1140481@utah.edu, UID: u1140481

Background and Motivation
Both of us are in Cem Yuksel's introduction to ray tracing class, and are interested in visualizing what's going on "under the hood" of our ray tracers. In a typical ray tracer implementation, a series of rays are tracked and combined for each pixel on the screen. Multiple effects, like anti aliasing and depth of field can be created by altering how many and where these rays are generated within a pixel. As each of these ray bounce off objects, they emit a color based on a set of light and material properties. Each of these bounces can generate additional reflective and refractive secondary rays, which can in turn bounce off additional objects, collecting color as they go.

As rays travel through a scene, they must test for intersection with a variety of primitives, which can be expensive. As a result, optimizing the ray tracing process is a fairly heavily researched topic. A potential way to learn optimization insights is to study the intermediate data created during the ray traversal process. Unfortunately, the amount of data created by ray tracers is very large, and can be hard to manage and learn from.

However, If patterns can be spotted within this data, it may be possible to improve the performance of ray tracing. Rays intersecting the same geometry can be processed together for better memory coalescing. Rays that are found to have insignificant impact can be terminated early. Photon mapping has the potential to converge faster if light to camera ray paths are processed first. And scene graphs could be filtered for improved primitive intersection times.

Additionally, providing an intuitive and interactive visualization for the ray tracing process may help improve our and other students understanding of ray tracing. As raster based graphics has improved, people may be wondering what the advantages are in moving to ray tracing. Some material effects, like refraction of light, can seem counterintuitive, but could be explained with this tool. As rays bounce, subtle changes in the image could be highlighted

Project Objectives
Primary questions we would like to answer
- What rays are intersecting what geometry
- What rays are creating bottlenecks

- Which rays are most important to the final image
- Performance Cost vs Image Quality metrics
- How does texture filtering affect an image
- How does anti aliasing affect an image

We would like to accomplish
- Building a tool with high quality interaction

Benefits of this project include
- Better understanding of the ray tracing process
- Teaching the inner workings of ray traversal to students just starting to learn about computer graphics.

Data

Primary data is coming from our ray tracer implementations from Cem Yuksel's Introduction to Ray Tracing class: http://graphics.cs.utah.edu/courses/cs6620/fall2017/
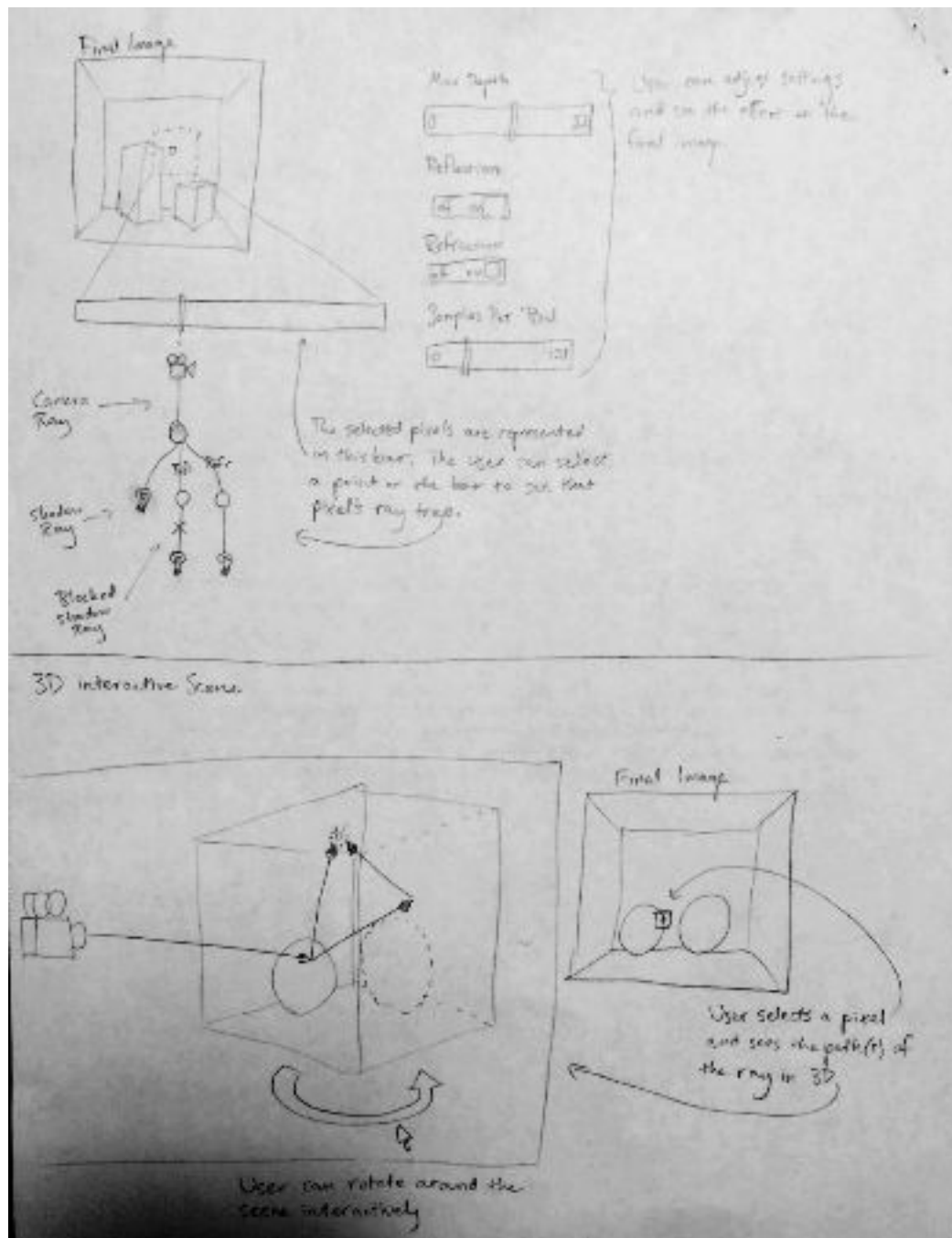
As scenes are generated, we would export intermediate results for visualization. For each pixel, we will generate a ray tree structure. Links represent rays and nodes represent intersections. Each link will store its ray direction, length, depth, type (camera, shadow, reflection, or refraction), and whether the ray was blocked if it was a shadow ray. Each node will store the object that was intersected, the position in space, and the pixel color contribution.
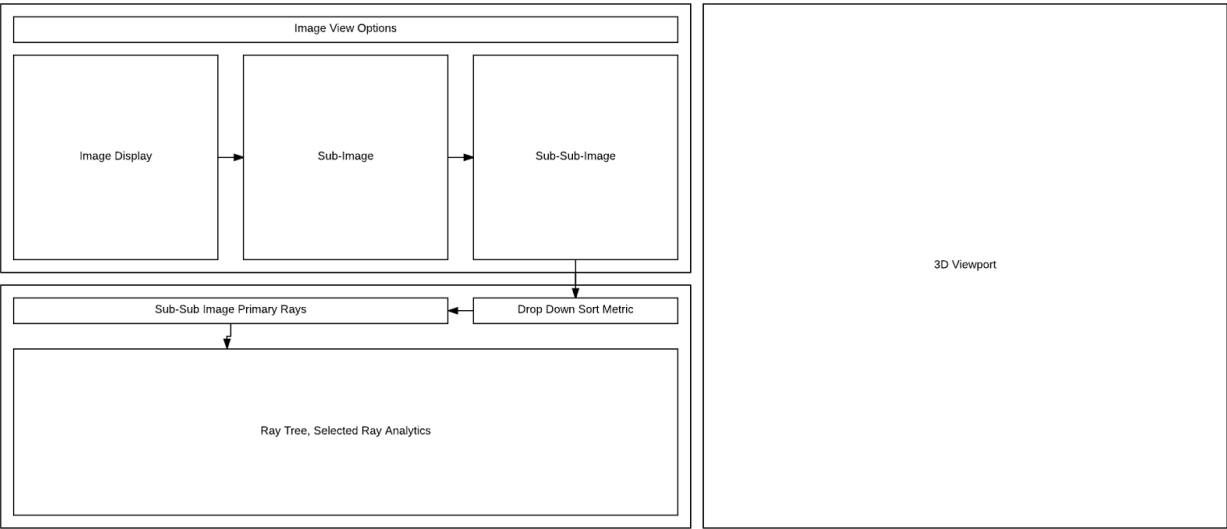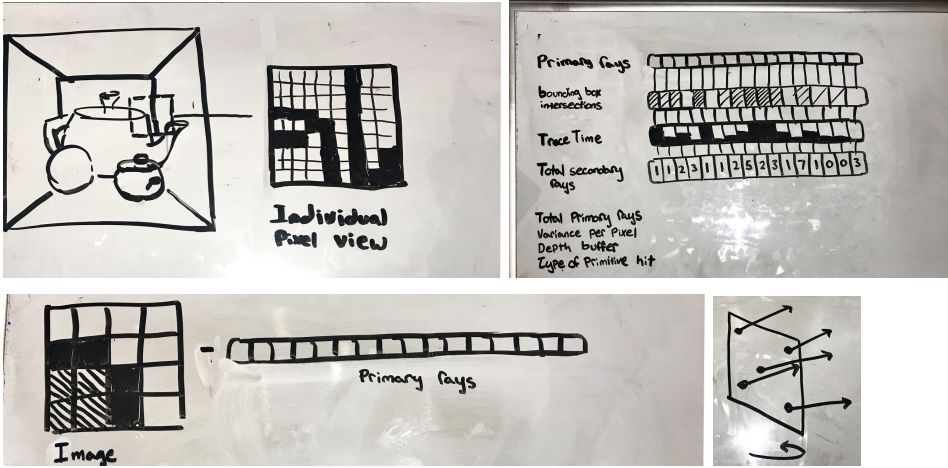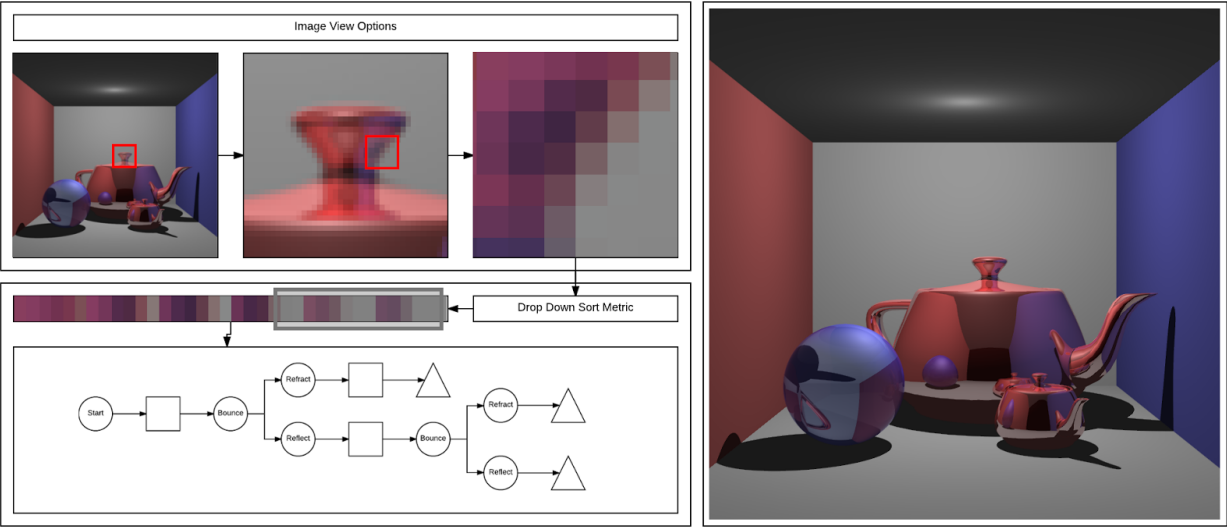
Data Processing

We will be generating all of the data with our ray tracer, so there should not be substantial data cleanup; we can generate exactly what data we need and organize it how we need it.

Visualization Design
Sketch 1:

Sketch 2

Image View Options

Image Display → Sub-Image → Sub-Sub-Image

Start → Bounce → Refract / Reflect → Bounce → Refract / Reflect

Drop Down Sort Metric

Individual Pixel View

Primary rays

bouncing box intersections

Trace Time

Total secondary Rays

1 2 3 1 1 2 5 2 3 1 7 1 0 0 3

Total Primary Rays
Variance per Pixel
Depth buffer
Type of Primitive hit

Image

Primary rays

Image View Options

Image Display	Sub-Image	Sub-Sub-Image

Sub-Sub Image Primary Rays	Drop Down Sort Metric

Ray Tree, Selected Ray Analytics

3D Viewport

Must-Have Features

*List the features without which you would consider your project to be a failure.*

- Selecting groups of pixels in the rendered image to visualize the types of rays that passed through those pixels.
- Visualizing the scene traversal of individual rays as a tree.
- Visualizing which rays hit which objects, eg through parallel coordinates
- Visualizing distance traveled per ray
- Changing rendering parameters and visualizing the effect on the final image

Optional Features

List the features which you consider to be nice to have, but not critical.

- 3D interactive visualization of the scene
- In 3D, visualize the paths of one or more rays through the scene
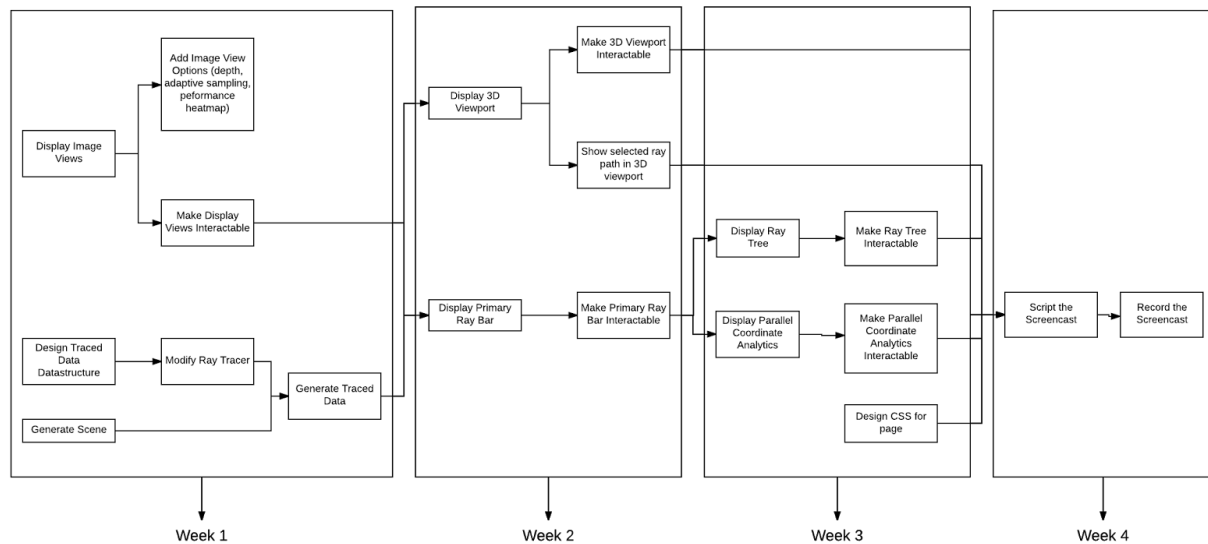- The ability to bookmark different rendering parameters and compare the resulting images quickly

Work Breakdown Structure
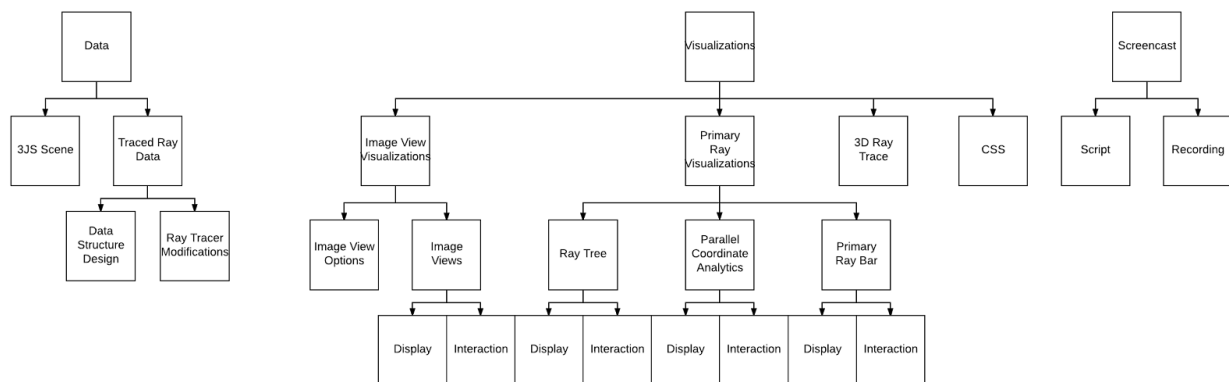
Project Network Diagram

Project Schedule

Week 1	Week 2	Week 3	Week 4
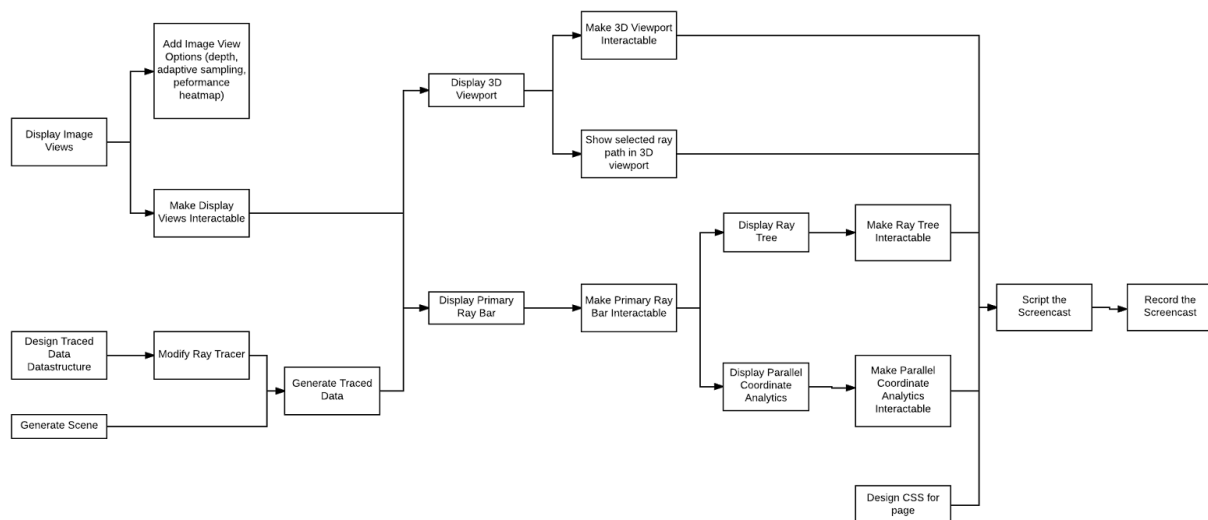
November 3rd - Get Week 1 Deliverables Done
November 10th - Get Week 2 Deliverables Done, Stubs for other Visualizations. (Milestone)
November 17th - Get Week 3 Deliverables Done.
November 24th - Catch Up, Final Details
December 1st - Finish Screencast (Project Deadline)

Milestones
- Code for gathering data written
  - Demo scene modeled, shaded, and textured
  - Demo scene rendered and data collected
- Raw data processed and organized
- Visualizations completed
- Optional visualizations coded
- Final product completed
- Screencast script written
  - Screencast recorded
- Project website completed
- Process Book completed

Overview and Motivation: *Provide an overview of the project goals and the motivation for it. Consider that this will be read by people who did not see your project proposal.*

Related Work: *Anything that inspired you, such as a paper, a web site, visualizations we discussed in class, etc.*

Questions: *What questions are you trying to answer? How did these questions evolve over the course of the project? What new questions did you consider in the course of your analysis?*

Data: *Source, scraping method, cleanup, etc.*

11/10/17: The data generation is proving more difficult than we previously thought. In order to continue the development of the project, we created a small set of artificial test data. The purpose of this was to establish the data structure so that we could start writing D3 code and expect a specific organization of the data. The actual data gathering is being worked on concurrently. Once that is in place, we can generate as much data as we need very quickly.

Exploratory Data Analysis: *What visualizations did you use to initially look at your data? What insights did you gain? How did these insights inform your design?*

Design Evolution: *What are the different visualizations you considered? Justify the design decisions you made using the perceptual and design principles you learned in the course. Did you deviate from your proposal?*

As of 11/10/17, our design has been modified slightly. We received feedback via peer review from classmates and have incorporated their suggestions into an improved design. The first suggestion is that we remove the "3-level zoom" layout. It's redundant to have three levels. So we removed one of them which freed up space for the other aspects of the layout. Another suggestion from our classmates was to overlay the various image visualizations (rendered colors, time to render each pixel, number of ray-primitive intersections, etc.) over the interactive 3D viewport. This way, that space can serve both functions. It is unlikely that the user will want to see the interactive viewport and the image visualization simultaneously. The following image is our redesigned layout:
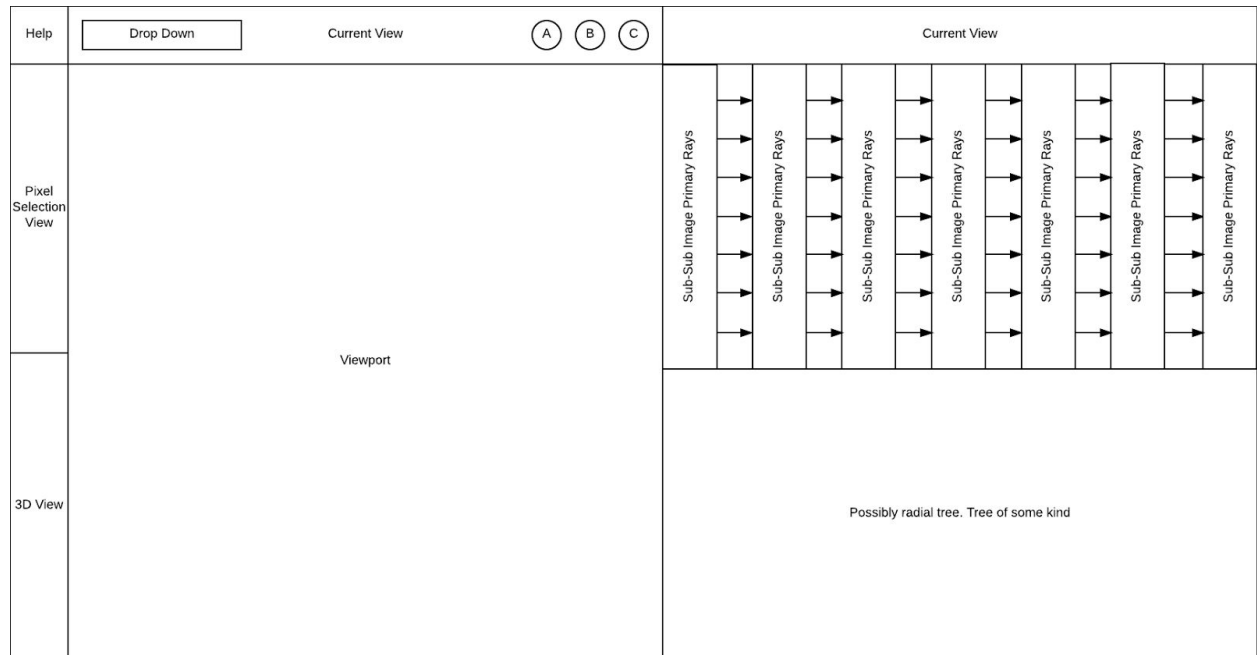
Another suggestion was to implement some sort of Magic Lens feature to allow the user to compare multiple image map visualizations quickly. We have yet to decide whether to incorporate this suggestion into our design and how it can be done.

Implementation: *Describe the intent and functionality of the interactive visualizations you implemented. Provide clear and well-referenced images showing the key design and interaction elements.*

Evaluation: *What did you learn about the data by using your visualizations? How did you answer your questions? How well does your visualization work, and how could you further improve it?*