An Interactive Map for the Yale Center for British Art

Vicky Tu

Advisor: Professor Holly Rushmeier

CPSC 490 Special Projects

7 May 2016

Table of Contents

Introduction

The Yale Center for British Art (YCBA) in downtown New Haven houses the largest collection of British Art outside the United Kingdom. Its 4-stories are filled with collections and exhibitions meant to amaze and spur conversation. Currently, visitors navigating the YCBA are invited to use markers located throughout the building and static floor plans on the YCBA website; however, the floor plans are non-interactive and tedious to alter as exhibitions change (Yale Center for British Art n.d.). For this project, we created an interactive map of the YCBA galleries displaying artworks currently in each bay. We built the website using HTML/CSS and JavaScript and queried from the YCBA's Linked Open Data collection using SPARQL. By

querying from the collection, the map will update itself to reflect the changing of exhibitions and the movement of artwork.

Approach

My goal was to make a product that would fit into the style of the YCBA website and the style of other museums' interactive maps. To make the project manageable, I partitioned the project into the following components:

- Background research: I researched existing museum websites to see how they represent the physical layout of their museum and its contents. In particular, I looked at the Metropolitan Museum of Modern Art and the Museum of Modern Art in New York City.
- Open Linked Data and SPARQL: Compared to Google Big Query, Open Linked Data is much more suitable for the kind of data stored and queried for art historians. I explored the intricacies of Open Linked Data and SPARQL in my quest to develop my query.
- HTML/CSS and Javascript: Since I was not familiar with these languages, I had to learn about them a great deal before I was able to start programming.
- Google Maps: I integrated a Google Maps component into my project in order to become more familiar with Google's API.

Background Research

Museums

The Metropolitan Museum of Modern Art (The Met) and the Museum of Modern Art (MoMA) are both popular NYC museums with collections similar in size to the YCBA's collection. On their website, the Met has colorful interactive maps of each floor of the museum. When visitors

click on part of a floor, a dialog box pops up. Inside the dialog box the visitor can view a representative picture of the gallery, a description of the gallery, and a link to the department the gallery exhibition belongs to (The Met n.d.). MoMA does not have floor plans displayed online. Instead, they have an online brochure which details the floor current exhibitions can be found (MoMA n.d.).

Ontologies and SPARQL

An ontology is a structure of definitions, types and the relationships between them. Schema.org is an ontology that make searching via Google or Yahoo easier (schema.org n.d.). The CIDOC Conceptual Reference Model (CRM) is an alternative ontology that provides structure to describe works in a way that captures their cultural significance (International Council of Museums n.d.). Therefore, many museums like the YCBA have opted to used CIDOC-CRM to index their data, rather than with Schema.org's vocabulary. Once entered into the database, the YCBA's data can be queried through a SPARQL endpoint (Yale Center of British Art n.d.).

Google Maps

With Google Maps API, one can display marked maps, places information, and even predict travel time. For this project we are mainly concerned with dynamically displaying the path from a visitor's selected location to the YCBA via his/her selected travel mode (Google Developers n.d.).

Features

Clickable Map

I received the floor plans in .jpg format for the second, third, and fourth floors from the YCBA. I proceeded to write code that established different clickable areas of the image. I had to either use

SVG, a language used to define vector-based graphics for the web, or HTML maps tags to define clickable area of the image. While I could have used an online resource to convert each .jpg into SVG, the result would not be useful. I could have also used SVG to recreate each floor plan, but this would not be helpful for the YCBA, who wanted to use the floor plans their graphic designer create. Thus, I opted to use maps tags, which defines clickable polygon areas by the absolute coordinates of the polygon corners. To find these coordinates, I downloaded GIMP, an image editor. To test that the maps had different clickable areas, I temporarily set the href attribute of each area to one of two images. When I clicked on an area, the first image would load, and when I clicked on another area of the same image, the second image would load, demonstrating to me that there were different clickable areas on the .jpg.

Persistent Box On Click

To orient the user, I wanted to add a persistent indication that a bay has been clicked when the user clicks the bay. I experimented with maphilight, a Javascript plugin, but realized that library was better suited for indicating when the user is hovering over the bay. Instead, I wrote code that added the clicked area into a "selected" class when clicked and removed everything else from the selected class, so that only one bay would have a box around it at any time. I also attempted to use maphilight to make each bay light up when hovered over but removed this functionality because it did not improve user experience.

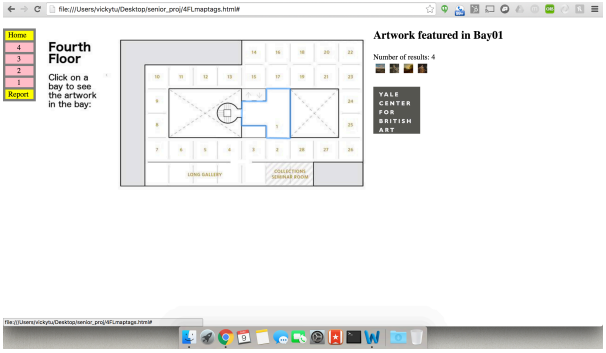


Figure 1 Box around Bay01 on click

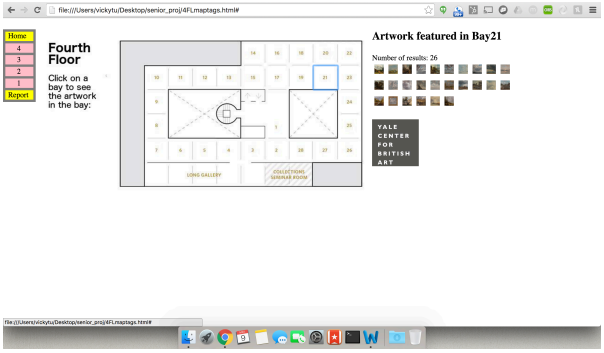


Figure 2 Box around Bay21 on click, other box vanishes

Enlarged Image on Click

Once the bay was clicked, the artwork in the bay would display on the website. At first, the results would load into a table with a name column and image column. Because there could be many pieces of artwork in each bay, this table could be very long. To save the user from unnecessary scrolling, I made the images 10 x 10 thumbnails. My code allowed the image to enlarge upon hovering and collapse back into a small image upon mouse out. But this created problems as the table would change shape and the user would accidently cause the image to collapse again just by shifting the mouse. For my second attempt, I made the images enlarge on click and collapse upon mouse out. But the table changing shape was aesthetically unpleasant. For my third attempt, I made the image expand elsewhere upon click, with each click replacing the same image object. This was the most functional, although I had to insert a placeholder image since HTML5 does not allow empty source attributes. To make it aesthetically pleasant, I decide to remove the table and only load the thumbnails. Upon clicking the thumbnail, the user would see an enlarged image of the artwork, the title of the work, and the artist.

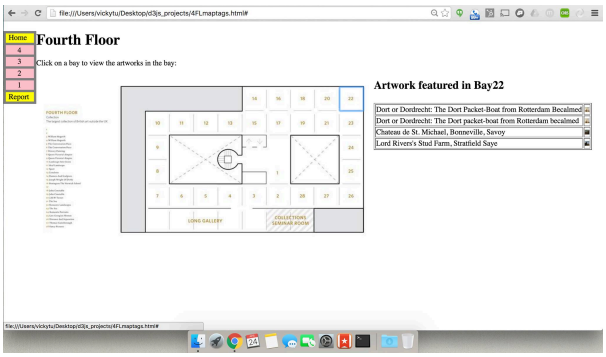


Figure 3 Before: results loaded into table

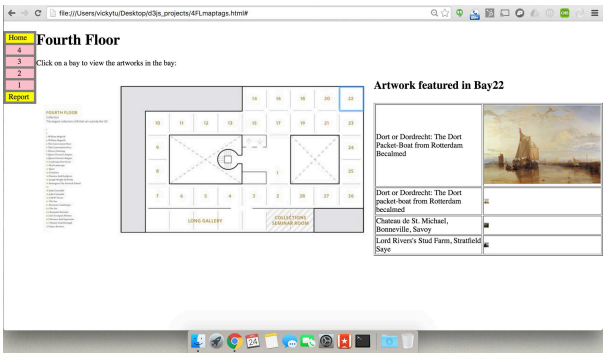
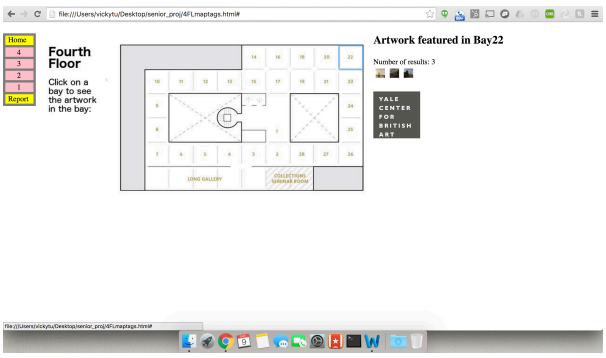


Figure 4 Before: thumbnails expand in place on click



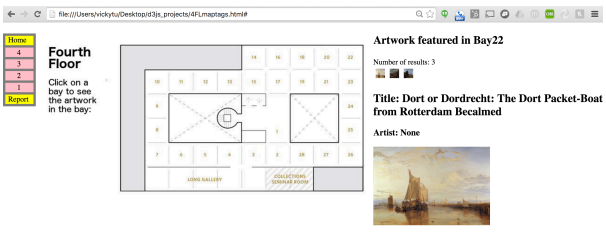


Figure 5 After: results loaded as thumbnails

Figure 6 After: enlarged image, title, artist on click

Slim Menu

I started with a basic bullet point menu since I needed to quickly test each page as I developed. After studying the Met's interactive map, I decided I wanted a similar narrow menu that consumed minimal page real estate for the final product. After finding no suitable images online to make into a clickable menu, I decided to create my own using SVG. The menu I created looks like a 4-story building, so that when the user clicks on a floor in the image, they will be taken to the YCBA floor plan for the corresponding floor. An additional benefit of making the menu an image was that I could make the image float to the left, allowing the map image to appear next to the menu, thus saving page real estate.

Google Maps Component

Initially, I only had a Google Maps of the area surrounding the YCBA. Then I added a drop-down menu that allowed the user to chart the driving path from some preselected locations to the YCBA. Finally, I added another drop-down component that allowed the user to designated one of 4 modes of travel.

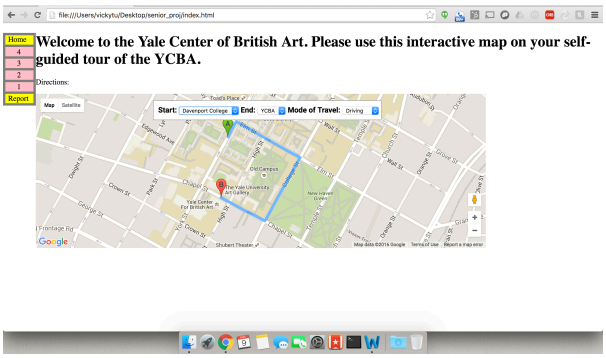


Figure 7 After: dynamic Google map to YCBA

Obstacles and Solutions

Querying on SPARQL Endpoint

I wanted a query that, given the bay number, would return the title of the artwork, the artist name, and the thumbnail image url. At first my query only returned the title and image url when given the bay number. But I modified this query to return the object ID instead of the title because several artworks in the YCBA's collection had multiple titles. I created a second query that returned the title and artist name when given the object ID. However, I had to break this up into two queries because some artworks do not have artist information, which interfered with my correct query for the title.

Querying from HTML

After creating the correct query, I tried to query from an HTML page. I discovered example code that allowed the user to query from a different endpoint using just HTML/CSS and Javascript. I modified the query and endpoint to make it execute the query I wanted on the YCBA SPARQL endpoint. This did not work until I found code on GitHub, in which they specified a different url as the endpoint (International Image Interoperability Framework n.d.). I changed the url in my code and the query worked.

Querying on Chrome

I choose to develop on the Chrome web browser since I could use Chrome Developer Tools to debug my code. While my .html page loaded perfectly, when I clicked to query Chrome threw an error: "No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access." HTML communicates through many headers. This error meant the response from YCBA's SPARQL endpoint did not have a header that Chrome required. Since I could not easily intercept the response message before Chrome received it, I

installed a Chrome extension called “Allow-Control-Allow-Origin:*” which allows one to request from any site by adding to the response the “Allow-Control-Allow-Origin:*” header. In the future, this can be remedied if my code is hosted on the same server the collection is hosted on.

Page Real Estate

Despite creating a slim menu, my code wasted considerable page real estate due to the title of the page occupying an inch of the top of the page. I decided to slightly modify the floor plan images by replacing the legend with the title of the page. I accomplished this using GIMP.

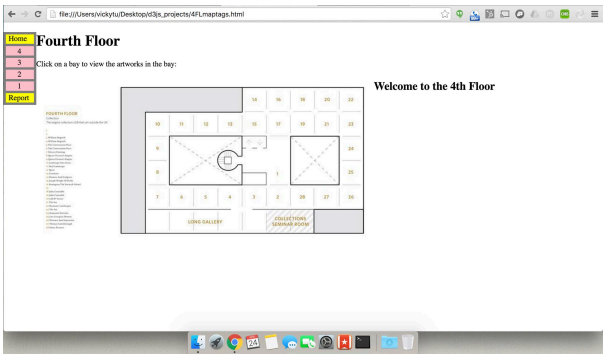


Figure 8 Before: wasted page real estate at top of page

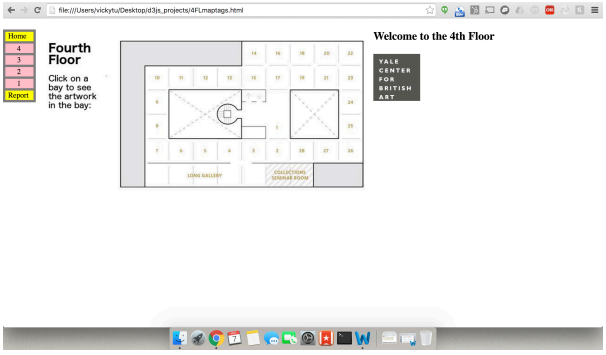


Figure 9 After: Shifting title down saves page real estate

Resources

To learn HTML/CSS and Javascript, I used W3Schools tutorials (W3Schools n.d.).

For the SPARQL query, I referenced the W3C Recommendation on SPARQL (W3C Recommentation n.d.).

To find the coordinates of the bays in the floor plan images and to modify the floor plans, I used GIMP (GIMP: GNU Image Manipulation Program n.d.).

For Google Maps, I referenced the Google Maps Javascript API (Google Developers n.d.).

For the menu made from SVG, I used the corresponding W3Schools tutorial (W3Schools n.d.).

Although the final product does not use maphilight, I used David Lynch's documentation on maphilight when testing my code (Lynch n.d.).

Final Result

The final product is located on the CPSC 490 page. Every page features a modified floor plan. When a bay is clicked, a small thumbnail of each artwork in the bay is displayed. When the thumbnail is clicked on, an enlarged image of the thumbnail replaces the YCBA logo placeholder. Above the enlarged image, the title and artist information for the artwork is displayed.

Future Work

Since this is a prototype of the final interactive map that the YCBA would like, there is still much to be done. Although the Google maps component is being used to chart the path from a user's location to the YCBA, it could be used to show the user the location that the artwork was. Because of load time, the enlarged image is the thumbnail image, but the enlarged image could be much bigger if the larger image was queried. Lastly, the persistent box should not vanish when the user clicks away from the floor plan.

Acknowledgements

I would like to greatly thank Professor Holly Rushmeier for her endless support. As my CPSC 201 professor to my senior project advisor, she has seen me at both ends of my Yale Computer Science career, and I hope I have made her proud. I would also like to thank Emmanuelle Delmas-Glass, Collections Data Manager of the Yale Center of British Art, for helping me

tremendously with my project. A final thank you goes out to Professor Richard Yang, who inspired me to major in computer science.

Bibliography

Google Developers. *Google Maps API.* https://developers.google.com/maps/ (accessed 05 01, 2016).

International Council of Museums. *CIDOC CRM Home page.* http://www.cidoc-crm.org/index.html (accessed 05 01, 2016).

International Image Interoperability Framework. *Presentation-API.* https://github.com/IIIF/presentation-api (accessed 05 01, 2016).

Lynch, David. *jQuery maphilight documentation.* http://davidlynch.org/projects/maphilight/docs/ (accessed 10 2016, 03).

MoMA. *Brochure.* http://www.moma.org/momaorg/shared/pdfs/docs/visit/MoMA_English.pdf (accessed 05 01, 2016).

schema.org. *Welcome to Schema.org.* http://schema.org/ (accessed 05 01, 2016).

The Met. *Museum Map.* http://www.metmuseum.org/visit/museum-map (accessed 05 01, 2016).

W3C Recommentation. *SPARQL 1.1 Query Language.* https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (accessed 02 10, 2016).

W3Schools. *HTML(5).* http://www.w3schools.com/html/ (accessed 02 28, 2016).

—. *SVG.* http://www.w3schools.com/svg/ (accessed 04 10, 2016).

Yale Center for British Art. *Yale Center for British Art Floorplans.* http://britishart.yale.edu/visiting/floorplans (accessed 05 01, 2016).

Yale Center of British Art. *SPARQL Endpoint.* http://collection.britishart.yale.edu/sparql/ (accessed 05 01, 2016).