VISUALIZING POPULATION DENSITY

BASED ON WIFI ROUTER LOCATION

AND NETWORK USAGE

A Senior Project

presented to

the Faculty of LIBERAL ARTS AND ENGINEERING STUDIES

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

BACHELOR OF ART

By TYLER DEITZ

June, 2015

Contents

1. Introduction

1.1. ROBERT E KENNEDY LIBRARY AS A MULTICOMMON

Robert E Kennedy Library at California Polytechnic State University is shifting from being a just book checkout utility for Cal Poly, and into a hub for student life and campus culture.

This change is documented and observed by Anna Gold, University Librarian of Cal Poly, in her presentation at the 2013 Open Knowledge Conference titled “Open Culture at the Heart of the University: Libraries as Multicommons.” [1] In the presentation, Gold references a trope in science fiction of a virtual reality that is created and affected by those currently in its simulation. Gold examines this technological fantasy, and shares examples of how currently organized spaces recreate this system, and how the systems attempt to translate the impossible technologies of fiction into the actual aspects that make up organized spaces. Gold deems organized spaces that are attempting this as “Multicommons”: a space where reality is fluidly interpreted and manipulated by the consciousness of its inhabitants.

Libraries, as she proposes, are transitioning into multicommons, and its tools available for construction are its physical spaces, contained technologies, and people within it.

The academic possibilities of a multicommon are just as ambitious as the science fiction inspiring them. Gold cites the functionality of the holodeck, a virtual reality room in the TV show Star Trek, as being “a cultural repository of narrative possibilities that would normally be excluded from the ship's own sociohistorical moment.” [2] This is an exhilarating goal for a library to achieve, and denotes a transition from libraries solely being an archive of human creation into a canvas for developing changes in our world.

1.2. MY INVOLVEMENT WITH THE LIBRARY

In May of 2014 I was hired as a web developer student assistant at Kennedy Library under the direction of Conny Liegl, Designer for Web, Graphics and UX. My duties involved improving the performance, design, and usability of the library's website. My work was mainly split into two categories: (1) quick fixes and edits to the site, and (2) large-scale products to push the website beyond its current features of use.

I was happy to learn that student assistants in Conny's design team were encouraged to incorporate one substantially large-scale product into their senior project. At the time I began my position I was two academic quarters away from proposing my own senior project to the Liberal Arts and Engineering Studies department, so I engrossed myself with the technical and organizational structure of the building in an attempt to find the most suitable senior project for my academic interests.

1.3. THE MAPS APPLICATION AND ITS POSSIBILITIES

Early into my position at the library I was exposed to the ongoing maps application project. The application is structured as a simple view of each floor of the library, where the user can select the floor they want to see from a drop-down list at the top. The map had been set up as a multidisciplinary project; my student co-worker, Benjamin Keiffer, had based his senior project in the Graphic Communications department on completely redesigning the maps to follow a clean and unified design aesthetic. Carl Hunt, the library's applications programmer and integrator [3], had utilized Benjamin's map designs into a web application that currently exists today [4].

The application was designed to be a basis for more utilities to be built upon. Potential future features in the maps application can be modular and self contained, with the underlying

topography acting as a simple canvas for iteration. Different uses for the map can be turned off and turned on, the same way that filters in other maps applications can be toggled such as traffic conditions in Google Maps. [5]

Different departments of the library had conceived of different map features. Catherine Trujillo, the library's exhibits and campus arts curator [6], is excited about contextualizing library events and exhibits to their designated spaces on the map. My boss, Conny Liegl, wants to make specific areas of the map selectable and shareable so students will be able to show other people where they're located in the building, simply with a URL. The Library Information Technology department had already begun developing a feature to show computer availability for all the public PC workstations in the library. Andrew Wang, a student assistant for the Library Information Technology department, entered the 2014 Cal Poly hackathon and created the Poly Book Tracker: an application that shares the location of books on our map when a user searches for their call numbers. [7]

With all of these exciting proposals, there needed to be a standard architecture for all features to be built on. The architecture needs to pertain to the code structure of each feature, and to the design language of each feature. All features should be coded relatively the same way, building off the application's basic codebase as foundation, and all features should use similar design patterns that reflect visual cues in the application's current user interface design.

2. Deliverable

My final product addresses two issues with the library previously described: the growing number of occupants in the space (due to its transition into a multicommon), and the lack of standards for adding new features to the maps application.

As a proof of concept for iterating our base maps utility, I began developing a tool that solves the problem of visualizing population density in spaces of the library. The final goal of the product is to let users see the crowdedness of spaces in the library and react to that information in whatever way they would like to, whether by locating themselves to an empty area to study or to a crowded area to socialize and engage.

The population density tool for Kennedy Library's maps application will be an iterative feature of the basic mapping software. It will be presented to users as a layer over the standard topographical view, using visual cues to show the user how crowded spaces in the library are. The visual overlay will mimic how other visual data should be added to the maps app in the future.

This information will have to be delivered in an intuitive and visually aesthetic way. Users of the application will know that the layer of population density is currently being presented, and must understand which areas are crowded based on my visualizations.

The final product will be a web application based off the existing technologies of the current maps app. It will be a proposed model for the library's engineering and design team on how to technologically and visually add new information to the map in a simple and unified way.

3. How to measure crowdedness

3.1. EXISTING TECHNOLOGIES TO MEASURE CROWDEDNESS

Collecting the number of people in a space is a process that can be approached using varying technologies. To determine the technology stack the library would need to provide the crowdedness feature, I researched three current companies: Aimetis, Placemeter, and Density. These companies all use different tools to provide crowdedness information to businesses.

Aimetis People Counter is a software application [8] that is built atop the AXIS Camera Application Platform. [9] Axis is a surveillance camera company that exposes their video feed software to developers. This creates an ecosystem of different use-cases for the camera, increasing the number of prospective customers.

The Aimetis People Counter tracks the number of people walking past a digital boundary that is determined by the company. Its software is able to log the number of people passing the barrier in either direction, thus logging the amount of people going into and out of a space. Because the size of the barrier is only determined by the composition recorded by the camera, companies using the software set the surveillance at a bottleneck of consumer traffic, mainly at the entrance and exit of a building. The company can then set up automatic data reports on their traffic to, for example, show growth trends over a period of time. [10]

Placemeter [11] is a startup company similar to Aimetis, except its scope of technologies used and its points of data collected are much larger. Its prospective customer base is incredibly broad; on Placemeter's "About Us" page, they state that "Businesses use Placemeter to better understand consumer habits and optimize their operations. Cities use Placemeter to help improve transit and urban areas. People use Placemeter to create original research and generate data that make their neighborhoods safer." [12]

Placemeter's software collects more than just the amount of people in the frame of its camera's feeds. CEO Alex Winter spoke at the 2013 Websummit startup competition and shared the possibilities of what their detection can interpret. The algorithm created by his team is able to count the number of vehicles, the speed of vehicles, the number of humans in a space, and the number of humans entering or exiting a space. [13]

Placemaker relies on video feeds that capture the outside areas of buildings, rather than from the inside like Aimetis. Depending on the locations they have surveyed, they contact prospective customers and sell their analytical data to them. To increase the amount of prospective customers, Placemaker rents window views from private properties. If a person has an apartment with a good view of nearby stores or streets, they can set up a smartphone to stream their view to Placemaker and make a monthly profit. [14] Placemaker began in New York City, and in 2013 already had 20% of the entire city covered. [15]

Density [16] is a recent startup that takes a different approach to collecting crowdedness than Placemaker or Aimetis. A business owner can get a piece of hardware from them that counts the number of smartphones connected to the business's wifi network, by tracking the MAC addresses on each device.

At Launch Festival 2014, CEO Andrew Farah presented his product and the process they use to determine trends for business owners. Because not every person in a space has a device that can connect to wifi, what they aggregate is trends of population and not specifically absolute numbered values. To test that at least a good amount of people in major cities do carry devices that can be read by their product, the team tested their setup at a stadium with 20,000 attendees and gathered that 71% of people there had smartphones with wifi enabled. [17]

However, Density has recently changed their technology stack to utilize IR beams instead of wifi connectivity.[18] This is a move that could possibly be due to increasing the simplicity of

production and installation, addressing privacy concerns of tracking individuals off identifying MAC addresses, or the exposed inaccuracies conducted by their stadium test. The IR beam hardware should be put at a bottleneck entrance and exit of a space, much like the Aimetis People Counter.

3.2. CHOOSING WIFI TO MEASURE CROWDEDNESS

I proposed to determine the crowdedness of the library using a tactic similar to Density's original strategy: measuring the amount of people connected to a specific wifi router within the library. Wifi routers are already an infrastructure that is installed in the library. Also, all of the services I could find were only able to analyze the number of people *in* a building, not *where* they are inside of the space. For that to occur using focused frames such as video or IR beams, there would need to be a device at every single door for every single room in a building. As the space becomes larger and more complex, so does the infrastructure needed for focus-framed devices that just log the in/out traffic. The focus-framed devices would also not be able to determine if one section of a room is more crowded than another section. This will be detrimental for large rooms in the library such as the entire second floor study space, one of the highest traffic spaces in the building. Using wifi routers as measurement saves money and time for Cal Poly to find the right system for data collection and then paying for a company's service.

Using wifi connectivity as crowdedness data would abstract all the structural complications of bottleneck entrances and exits of building rooms. Also, more than one wifi router can be placed in a room, gathering data from more than one section.

California Polytechnic State University uses a universal wifi network named "SecureMustangWireless." When users of the network are in any building on campus, they are all connected to the same network. This is possible by having a large number of routers

throughout all spaces, each allowing users to connect to SecureMustangWireless. When users move throughout Cal Poly's property, their devices automatically connect to the strongest signal available to them. Knowing that signal strength deters over distance, I concluded that each router would map an area in the library where the area of each is closest to that wifi router.

3.3. VORONOI DIAGRAMS AND PROXIMITY MAPPING

3.3.1. Voronoi diagram description

A voronoi diagram is a visual partitioning of a plane consisting of ***N*** number of points, such that there are ***N*** number of polygons on the plane each enclosing one point, where any position on the plane whose distance to a point is closest than any other point on the plane is enclosed in that point's polygon. This means that if you are "in" a polygon on a Voronoi diagram, the closest point to you, out of all the points in the plane, would be the point enclosed in the polygon. Voronoi diagrams were first theorized by Georgy Fedoseevich Voronoy (1868 – 1908), a Russian and Ukrainian mathematician. [19]

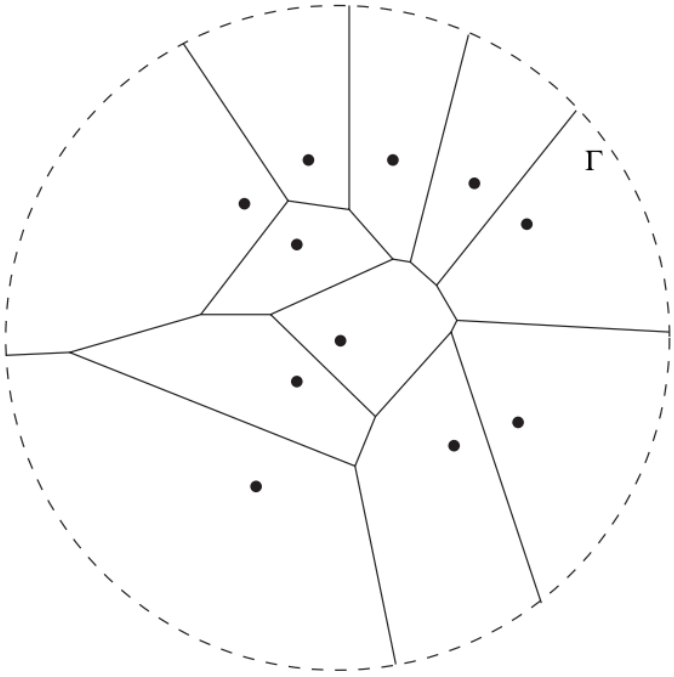

figure 1: Voronoi diagram example [20]

Voronoi diagrams are calculated using algorithms that determine edges of the polygons, using the Euclidean distance formula $d(p,x) = \sqrt{(p_1 - x_1)^2 + (p_2 - x_2)^2}$ to find the position of a graph that has the same distance between it and two or more other points.

3.3.2. Use of Voronoi Diagrams In Spacial Distribution

One of the first and most famous usages of Voronoi diagrams is the water pump theory of the 1854 outbreak of cholera in London. Before germ theory, the concept that diseases are carried by microorganisms, was fully established, John Snow had mapped out the areas of cholera outbreaks in the city and overlaid the map with a voronoi diagram of the city's water pumps.

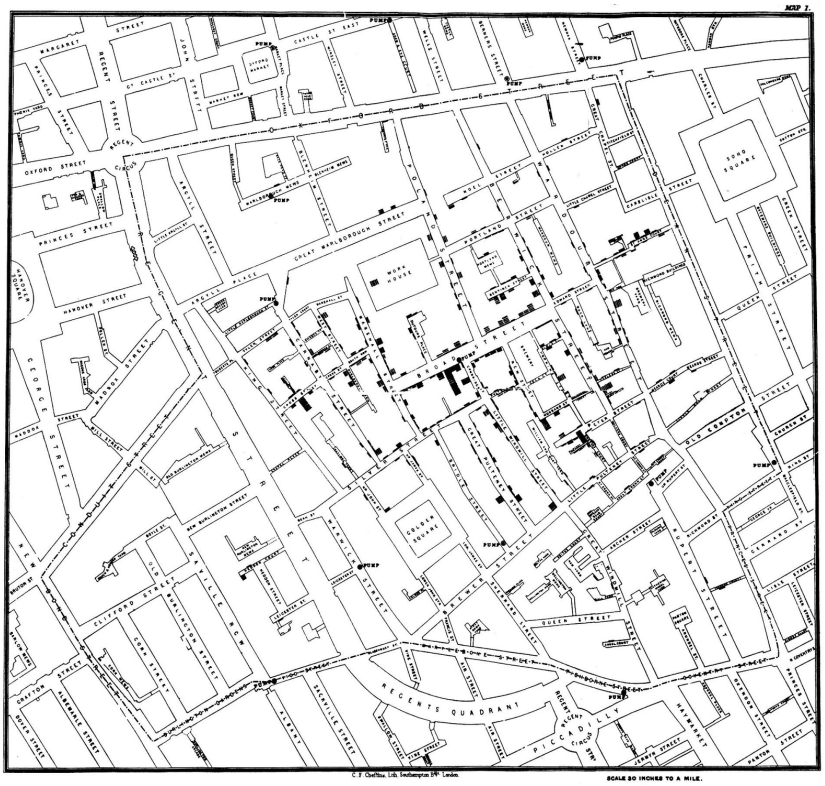

figure 2. Showing the clusters of cholera cases in the London epidemic of 1854.

Snow was able to correlate the outbreaks of the disease to being within the polygon of one specific water pump, the Broad Street pump. The pump's handle was then removed by the city, and outbreaks of the disease began to die down. [21]

3.4. USE OF VORONOI DIAGRAMS IN WEB APPLICATION USER INTERFACE DESIGN

Voronoi diagrams are also useful in the design of user interfaces, specifically concerning the relation of a user's mouse position to a set of pointed-to interactions on a page. If there are a set of buttons laid out on a page, and a user's mouse is closest to one button in the set more than any other button in the set, then the interface designer can infer that the user's mouse position is related to the action of the button. In 2014, programmer Mark DiMarco gave a presentation at

the web development conference JSConf where he reverse engineered how the New York Times had programmed some of their interactive graphs on their website. In an interactive graph by the New York Times called “Long-Lived Greatness”, the New York Times plotted 185 professional baseball players and created a Voronoi diagram to map out the space on the graph to where the closest baseball player’s plot was. [22]

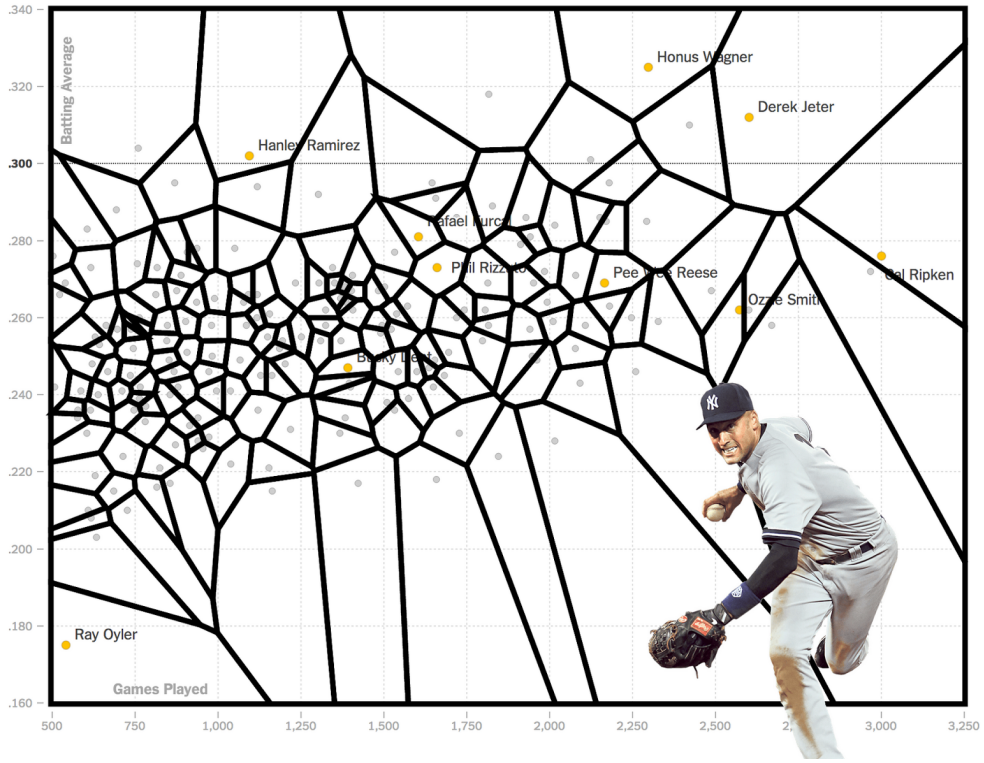

figure 3. Overlayed Voronoi diagram of the “Long-Lived Greatness” infographic. Based on a New York Times infographic. [23]

These web applications show that the use of dynamically created Voronoi diagrams in web application design is both possible and beneficial. This is due to the increased performance of the programming language Javascript, which is used by web developers to provide dynamic user experiences to otherwise static HTML web pages. Algorithms for creating Voronoi diagrams

have been implemented in javascript by Raymond Hill, creating the plane and polygons within an HTML “Canvas” element [24] and by Alex Beutel. [25]

4. Technology and design overview

4.1. BASIS OF THE APPLICATION

My library population density tool is built off the current technology stack of the library's maps web app. Each of the five floors of the library have an SVG illustration correlated with it. SVG files are "Scalable Vector Graphics" files that contain data points about vectors - meaning shapes and their features. SVG was a great format to design the maps in, because, unlike image files like JPG or PNG, SVGs can be presented in any size format with perfect resolution. This is because the map won't have any pixelated artifacts as it would have with a standard image file on a larger screen format; SVGs just contain information about vector paths, and are dynamically reconstructed to display the embedded shapes in any resolution.

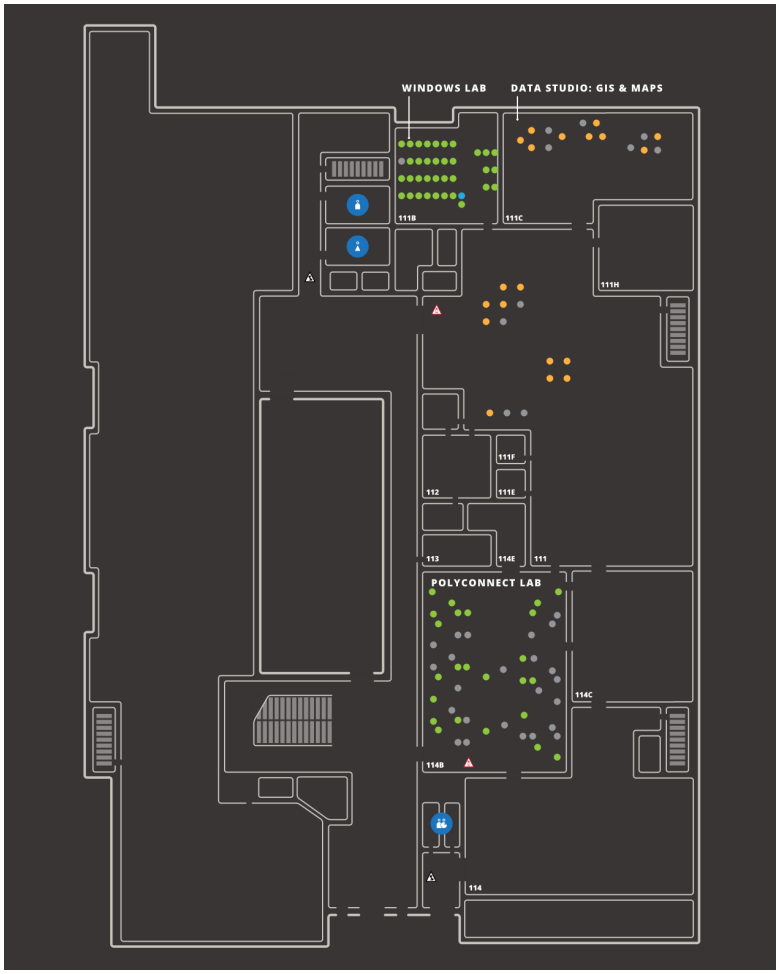

figure 4. SVG map of the first floor of the Kennedy Library.

The SVG files can be thought of as the "static" portion of the application. The library's engineering team combined the five SVG floor illustrations into one web application using the Javascript application framework AngularJS. AngularJS controls "states" that an otherwise static HTML page can be in, and can change these states depending on how the user interacts with a website (eg. clicking a button). When users go to the maps application, they can select what floor they would like to look at by clicking one of five buttons (one for each floor). AngularJS then changes the state of the HTML page to display the SVG image of the coinciding floor the user had selected.

To prototype the crowdedness feature, I stripped down the existing webapp into just showing a webpage with one floor, not using AngularJS to switch views for the floors. This is because I was only focusing on building an extension to the maps, and didn't want to increase the complexity of my work by building the crowdedness view into the application. The additions I created are flexible enough, to be able to work with any floor.

4.2. DYNAMICALLY CREATING VORONOI DIAGRAMS OFF WIFI ROUTER LOCATIONS

The SVG file containing the illustrations of the maps needed to be edited for my work. The original SVG was organized as a five-layer illustration. Each layer contained a different floor to the library. To add further contextual data to the file, I created a new layer called "Floor_1_routes." Inside that layer, I added ten red circle elements that represent the locations of wifi routers on the first floor.

Storing the locations of the wifi routers in the SVG file itself was a conscious design decision of mine. If in the future any routers were moved, added, or removed, anyone knowledgeable in Adobe Illustrator would be able to update the location information for the app. This increases the number of library staff members who would be able to conduct technical maintenance, because there is no programming involved at this point. It logically follows that more contextual information about the map would be included in the map SVG itself. Like I previously stated, AngularJS functions as a state changer for what layers are or aren't on. This functionality could be extended to showing a wider array of layers beyond the basic floor plans.

I used the D3.js javascript library to load and read data from the map's SVG file. D3.js stands for "Data Driven Documents", and is described by Cal Poly student Stephanie Friend in her paper "Visualizing Relationships between Related Variables: Improving Physics Education

through D3.js Network Visualizations" as being "intended for easy data manipulation and visualizations." [26]

When the maps SVG is loaded, the program traverses the SVG file looking for a layer that is titled "Floor_1_routes." After that, it adds every circle element to a list as a pair of x coordinates and y coordinates, representing the location of each wifi router on the browser screen.

I then used the D3.js built in function "voronoi" to create a set of SVG polygons that make up a voronoi diagram using the wifi router coordinates as points of generation.

4.3. DESIGNING CROWDEDNESS

Because my project focused on the implementation and design of the crowdedness feature of the maps, I didn't research how to algorithmically determine the crowdedness of areas using the amount of users connected to wifi routers. Instead, I set up static data on a scale of one to ten that would act as a finalized crowdedness attribute which such a program would determine. For every wifi router in the SVG map, there is an associated "crowdedness" scale correlated with it.

I initially envisioned my product to display crowded in a color-coded format, such as how heat maps show a heat intensity from red to green.

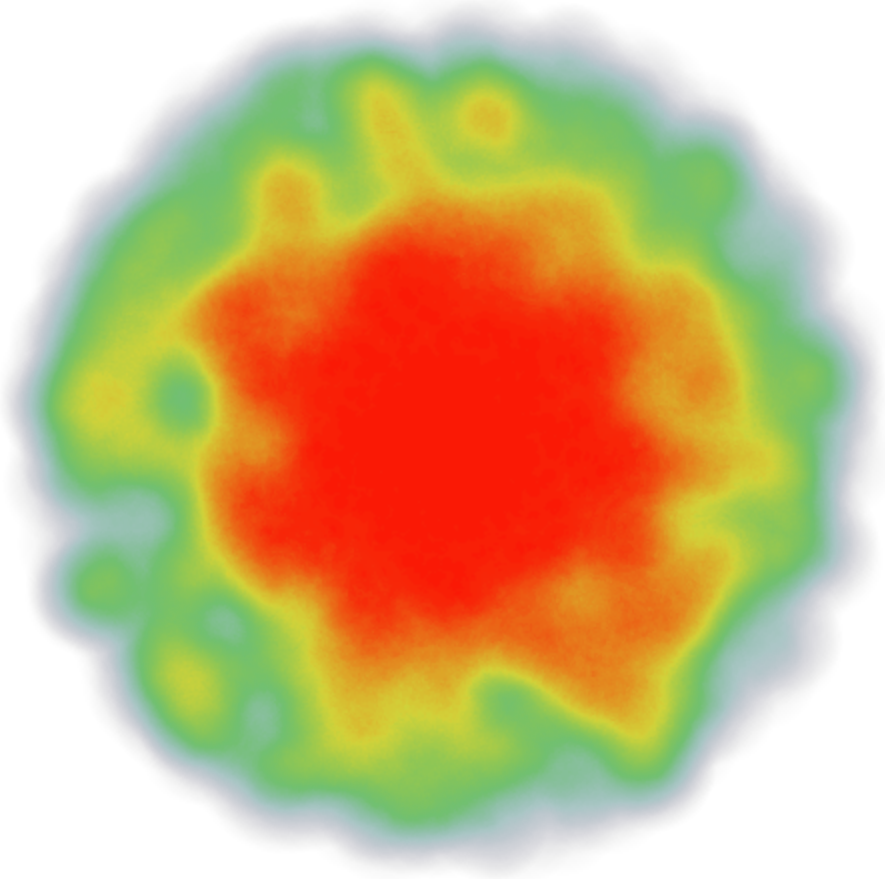

figure 5. Heat map generated by web technologies. [27]

Other software also uses a color scale to visualize intensity. For example, Google Maps, another mapping application, uses color to represent the traffic intensity of streets. The color of a street can either be red (lots of traffic), orange (medium traffic), or green (little or no traffic).

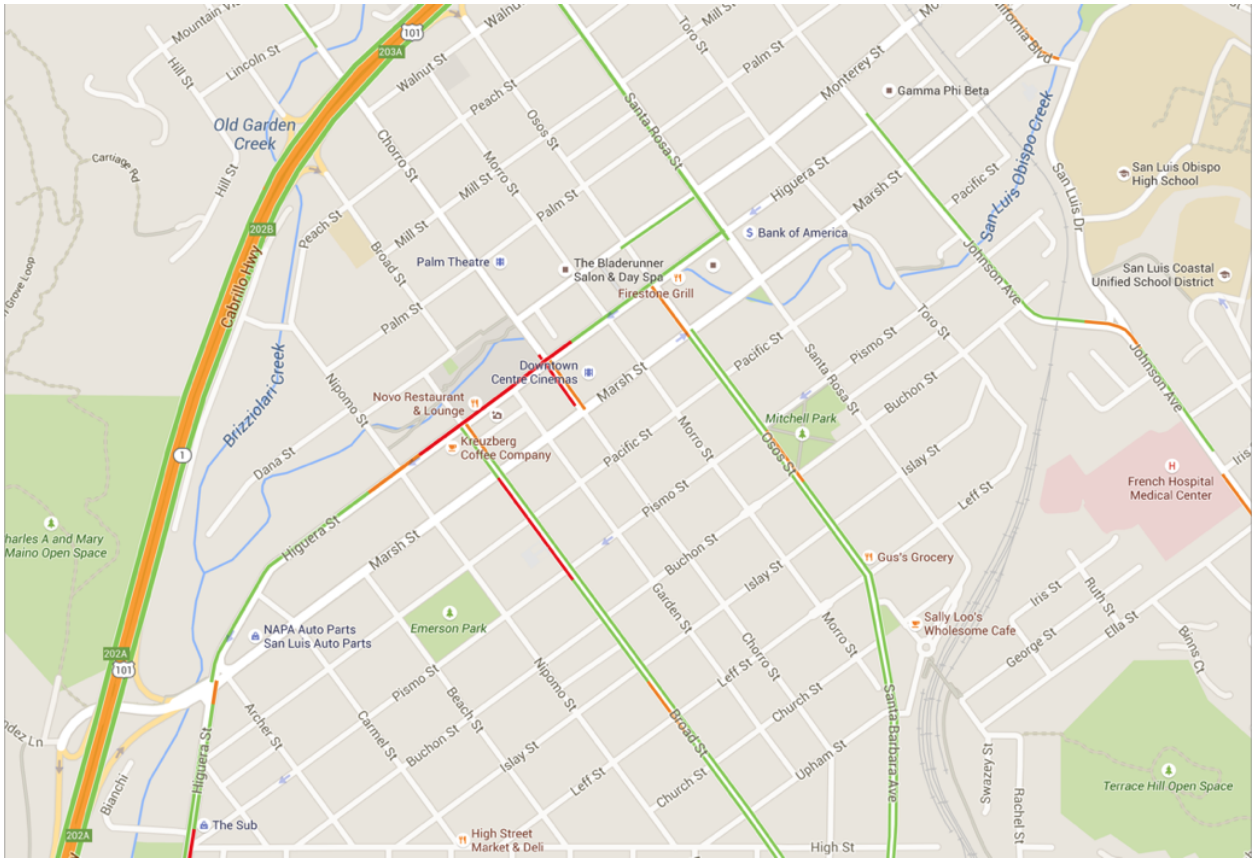

figure 6. Traffic intensity visualized in Google Maps.

To test a multi-color design, I configured my product so that each polygon would be a certain color depending on its crowdedness.

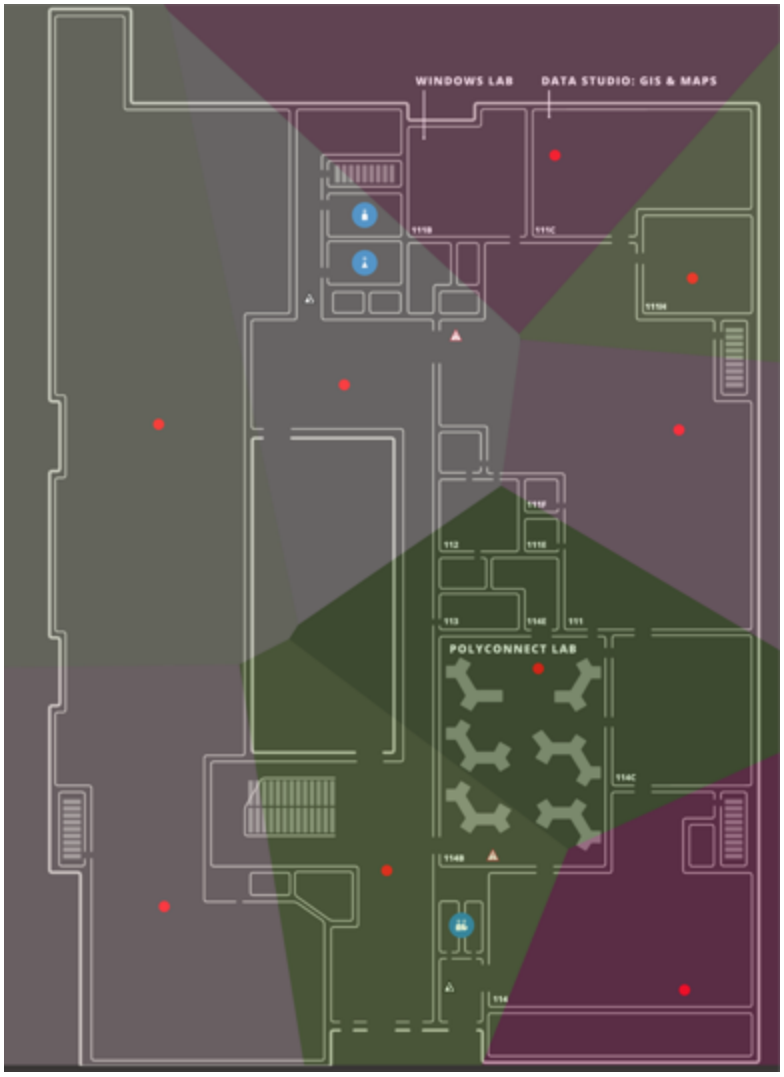

figure 7. Colored voronoi polygons generated by router location.

The color-coded polygons, although being concisely defined with the help of a map key, did not meet my expectations for a proper way to design crowdedness. Emphasizing to the user the hard-lined shapes of the crowdedness polygons, while layering it on top of the map, which is also built from lines, makes the map lose its focus and readability. It's too many different lines for the user to make sense of and, as I saw it, exposes the mathematics of the voronoi diagram generation in an unflattering way.

Another detriment to the hard-lined polygons is the inherent inaccuracy of wifi routers. Sometimes a wifi router signal may not reach the edge of its voronoi polygon, causing a user close to its edge to connect to another signal. If the crowdedness visualizations are shown as clear-cut lines of separation, it expresses a notion of accuracy that is unmet by the infrastructure for measurement.

I was concerned about the usage of color to display crowdedness and dropped the idea for my final product. In Google Maps, color is used to express the intensity of traffic which, to all drivers, is bad. I felt like, for example, painting a polygon red if it is crowded may express negative connotations about that space to a user. Because the library is transitioning into a multicommon, it is inherent that certain spaces would be crowded at times. The library even has an appointed exhibits and campus arts curator, Catherine Trujillo, who works to organize events and audiences to the building. I wanted the user experience to feel neutral for finding both empty and crowded spaces in the library.

Color also restricts my audience by making it incomprehensible for colorblind users. According to the California State University Website Guidelines [28], and the law [29], websites need to have “standards for accessibility and [...] electronic and information technology [...] to be accessible to people with disabilities, including employees and members of the public." Software should not rely on color as the only means to express data.

I came up with a final list of goals for my visualizations based off the initial colored map:

1. Obscure the underlying voronoi diagram so as to keep the visualizations simple and adhere to unavoidable inaccuracies with wifi routers.
2. Not use color to visualize crowdedness so as to express neutrality with different densities of population and adhere to design regulations for colorblind users.

This led me to a final design that is based off showing crowdedness as a layer of fog over the map, with dense fog representing a more crowded area, and light fog representing an empty area.

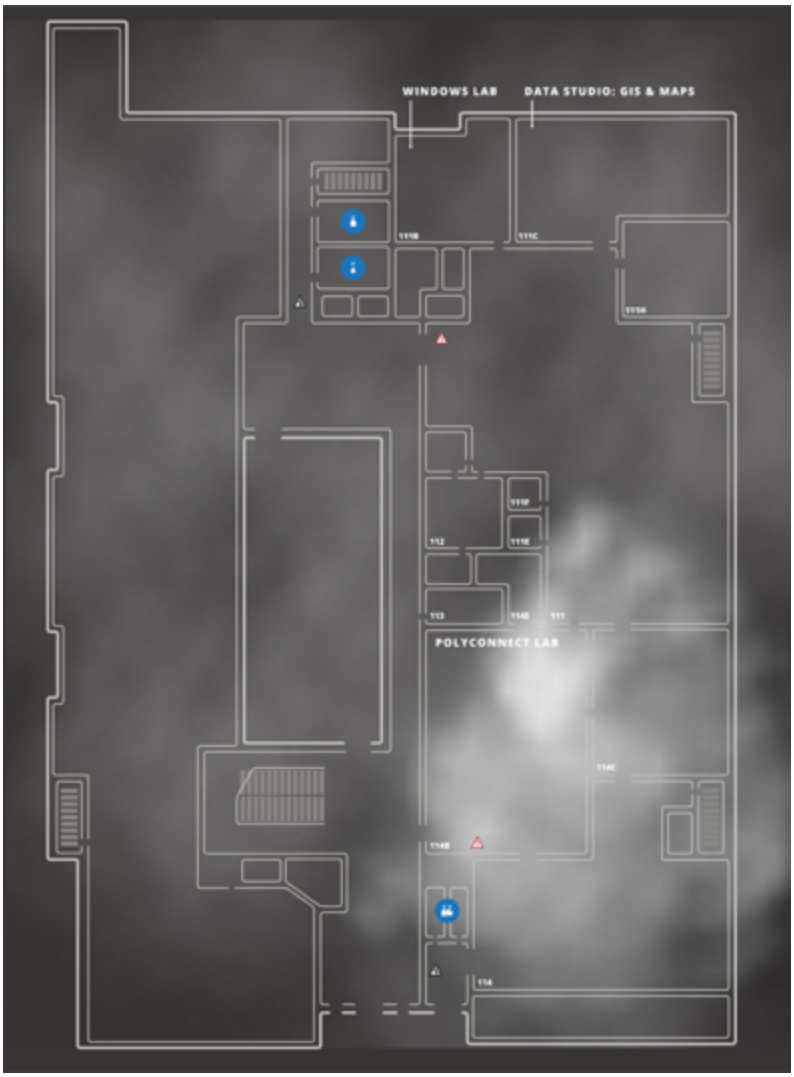

figure 8. Fog visualizations over the first floor of the library.

This design doesn't trail too far from visual elements that users expect to see on maps. For example, The Weather Channel's website has an interactive map that shows the cloud

density over America [30] . This map is shown every day on its television channel, teaching its audience how to perceive “cloudiness” as an element of user interface.

To achieve my first goal (obscure the underlying voronoi diagram so as to keep the visualizations simple and adhere to unavoidable inaccuracies with wifi routers), I faded the fog over the lines of the voronoi polygons so each edge was obscured by a smooth transition of density. Using seemingly formless, natural clouds atop the lined, structured map helps to accentuate both visual elements while not clashing them together.

4.4. GENERATING FOG USING WEB TECHNOLOGIES

The fog for my web application is created within an HTML “Canvas” element. The W3C definition of a Canvas element is “The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.” [31] Much like an artboard canvas, an HTML canvas can be drawn on with little rules and regulations for where and what to draw. This is different than other HTML elements which appear on a webpage based on their position in an HTML document. Normal HTML elements are “stacked” on one another from the top of a webpage to the bottom, with some exceptions such as elements that have fixed or inline positions. A canvas exists as a blocked HTML element, but what is rendered inside that canvas is completely up to the web developer to implement.

The visualization is created by generating thousands of copies of one small image of fog all over the map. When a user turns on the feature, a first wave of fog particles are expelled out as a circle from each router point. After that, a second wave is expelled to evenly fill out the plane. For every frame that the fog animates, my program loops through each particle, checking and affecting its internal data. Each particle contains the following information:

1. top - the distance from the top of the canvas
2. left - the distance from the left of the canvas
3. start - the time the particle was created
4. life - the time (in milliseconds) the particle should animate for
5. startOpacity - the opacity it should adhere to based on its birthing wifi router crowdedness data
6. newTop - the top position it will move to in the next animation frame
7. newLeft - the left position it will move to in the next animation frame
8. size - the size of the particle
9. startPoly - the voronoi polygon the particle was birthed in
10. degrees - the angle the particle will shoot from the router
11. distance - the distance the particle will travel

Based on all of these assets, the particles float onto the canvas. After every loop, I implemented an algorithm created by W. Randolph Franklin of Rensselaer Polytechnic Institute called PNPOLY (Point Inclusion in Polygon Test) [32], and converted to Javascript by James Halliday [33] to test if the particle has moved out of the original polygon it was created in. If it is, its opacity is changed to abide to the crowdedness data associated with the new polygon it's now over.

Implementing this feature is the reason the fog's edges are transitioned so smoothly. I first attempted to address this goal by creating multiple canvas elements over the map that were related to the number of wifi routers on the floor. I would mask every canvas to its relative voronoi polygon using the SVG mask attribute. SVG masks "will be painted onto the background through the mask, thus completely or partially masking out parts of the graphical object." [34] For my application, this means that the canvas will only be visible inside the voronoi polygon associated with it, isolating all different fog animations within each independent canvas.

My use of masks still doesn't solve how the fog will smoothly transition from one polygon to the next. From what I initially believed to be possible, I would give each mask element a "stroke" property, which acts as a border around an SVG object. The masks would have a uniform "stroke-width" property as well, which determines a width of each border. If all masks have the same border width, then each adjacent borders would overlap with one another's, exposing a small area in between the two masks where both of their fog effects would be shown. The strokes would have a gradient of opacity going from inside the particle to the outside edge of the border, smoothly transitioning the overlapping fog visualizations by fading in and out their opaqueness.

I initially thought this was possible with SVGs from doing tests in Adobe Illustrator. [35] I would create a polygon with a stroke value, then I would give that stroke a gradient classified as a "gradient across stroke." This adheres to what I was attempting to do with the shape smoothly becoming less opaque over the width of its stroke. As I learned, this feature for vector images isn't an organized standard yet, but has been marked by the W3C as a feature to implement in a future SVG standards iteration entitled "SVG2." [36] Currently, no modern web browser has all SVG2 features implemented to use, but Google Chrome, Mozilla Firefox, and Apple Safari are all actively developing them [37]. Looking at the SVG code that is generated by the "gradient across stroke" feature in Adobe Illustrator reveals that the feature is being encoded as a PNG image within the SVG file itself, because there is no standard way to define its shape yet.

5. Design / implementation timeline

(January - March 2015) Research similar technologies that measure crowdedness.

WHILE

(March 2015) Research ways that population density visually represented.

THEN

(May 2015) Program a crowdedness measurement system for the library maps.

THEN

(May 2015) Design a visual representation for population density in the library.

THEN

(June 2015) User testing on my designs.

6. Analysis and verification

There are two main aspects of verification to my final project.

1. Web application is universally compatible with all current browsers.

The web application and population density view correctly loads on all current modern web browsers: Chrome, Safari, Firefox, Internet Explorer, Opera, Chrome Mobile, Safari Mobile.

Gradations of success:

4. Application successfully runs on all web browsers, including mobile browsers.

3. Application successfully runs on all web browsers, excluding mobile browsers.

2. Application successfully runs on Chrome, Safari, and Firefox.

1. Application successfully runs on Chrome, the browser it is mainly developed in.

0. Application fails to successfully run on any web browser to usable standards.

Final grade:

The application loads the map SVG and gathers data about the wifi router locations accurately in all browsers. However, the fog generation and animation worked in all browsers except Firefox. I had expected that it would be easiest for my software to work in all desktop browsers first, leaving me to do final code iterations to work on smartphones and tablets. This wasn't the case and my gradations of success doesn't strictly determine what to conclude if my product works in all web browsers, including mobile browsers, except Firefox.

I'm appointing myself with a score of 2.5. I believe it is still a significant success of mine to program a performant enough product that perfectly works on mobile phones and tablets, which generally have much less processing power than desktop computers. Though, I recognize that not getting it to work on Firefox is a large hindrance in the goals of an open, universal web platform.

2. Creating visualizations over the Maps application that intuitively show how crowded certain areas in the library are.

My designs will be user tested with the approval of the Cal Poly Human Subjects Committee Institutional Review Board, with the goal of verifying that the way I visualize population density is useful and intuitive.

Gradations of success:

5. (1) Users understand how to switch on the population filter from the main maps application view. (2) Users are able to comprehend that the filter is a layover of the existing maps application. (3) Users are able to point out which areas in the library are busy, based on the visualizations presented to them. (4) Users find the tool useful and intuitive. (5) Users find the fog effects attractive.

4. Users are not able to complete or agree with 1 of the 5 goals.

3. Users are not able to complete or agree with 2 of the 5 goals.

2. Users are not able to complete or agree with 3 of the 5 goals.

1. Users are not able to complete or agree with 4 of the 5 goals.

0. Users are not able to complete or agree with 5 of the 5 goals.

Final grade:

I collected eight subjects from the Spring 2015 Liberal Arts and Engineering Studies senior project class (LAES 461/462) to conduct research with. The users were given a task script to complete, then filled out a survey expressing their opinions. Each of the five points were addressed in the survey and were graded on a scale of one to five. I divided each answer by five to make each topic worth one point, and I added the points together for a total score with the max possibility of five.

	user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8	
Analysis of success									
Users understand how to switch on the population filter from the main maps application view.	5	5	5	5	5	5	5	5	
Users are able to comprehend that the filter is a layover of the existing maps application.	5	5	5	4	5	5	5	5	
Users are able to point out which areas in the library are busy, based on the visualizations presented to them.	5	5	5	3	5	5	5	5	
Users find the tool useful and intuitive.	4	3	5	3	4	5	5	4	
Users find the crowdedness fog effect attractive.	5	5	5	4	5	5	4	4	
TOTAL	24	23	25	19	24	25	24	23	average
-- divided by 5	**4.8**	**4.6**	**5**	**3.8**	**4.8**	**5**	**4.8**	**4.6**	**4.675**
Future improvements									
Users would use this feature for another building in the future	3	5	5	3	4	5	5	4	average
-- divided by 5	**0.6**	**1**	**1**	**0.6**	**0.8**	**1**	**1**	**0.8**	**0.85**

My total score was a 4.675 out of five, which is a percentage grade of 93.5%.

7. Societal impacts

The population density tool will increase the usability of the library space, and increase the productivity of student research. When visitors to the library are trying to determine where to study, they can be able to see what areas in the library are the most crowded and avoid going to those places. This detracts from time wasted trying to find a place to study, and increases the student's comfort in the the space they have chosen.

The population density tool can also be expanded to other departments at Cal Poly. For example, the crowdedness of campus entities such as restaurants can be determined with my algorithms, and people can see how busy certain spaces will be before visiting. Keeping track of the MAC addresses logging into the wifi routers can determine the average time people stay within a wifi router's Voronoi space, concluding an "average wait time" for food or other services.

8. Future work / next steps

I would like to continue exploring the algorithm I use to generate the fog particles. Right now they are all expelled in a circular direction from every router, though I think that I will get a more uniform amount of fog on the canvas if they are simply gridded onto the canvas in an ordered fashion. If there is a completely even amount of fog on the canvas, then my other algorithm that manipulates the fog particles' opacities will be the only tool that determines how dense the fog will be over the map. This will lead to more accurate readings for crowdedness.

After that, I would build the tool into the original map application using the existing application framework AngularJS. This would require me to create four more layers in the map SVG file that contain the locations of the routers on the second, third, fourth, and fifth floors. I would also have to set up a variable in my existing program that observes what floor the user is currently viewing, and affect my code to generate fog based off that floor's router data.

The next steps would be to base the data in my application off the actual data of the the library building. I need to reposition, add, or remove routers to adhere to the actual router structure in the building, and I need to work on algorithms for determining crowdedness based off the number of people connected to a wifi router.

The complete goal of my feature is to express crowdedness. This goal can be split into two actions: determining crowdedness and designing crowdedness. I only tackled the later goal of designing crowdedness in my project and simply hard-coded static data about the crowdedness of each location on a scale of one-to-ten.

Once I come up with a reasonable algorithm to determine crowdedness, I would need to conduct more user testing to see if the results I had configured reflect human perceptions of crowdedness. Possible user testing scenarios would be to take people to different locations of the library and have them rate the crowdedness of the library, from their perception, on a scale of

one-to-ten. If the values I programmatically had determined were within a range of accuracy to the human perceptions, then I would determine my program successful and combine the already completed visualizations of crowdedness with the automatic determination of crowdedness.

9. Conclusion

I am very excited to present my work to the library, both my ideological reasoning for its existence and the technologies used to build it. No matter the purpose visitors of the library have when attending, crowdedness is a universal concern to all occupants. Addressing different features of the map, such as a faster way to find books, would restrict my audience to those who are only at the library to research its archives. My work has the possibility to affect everyone inside the building, while keeping a neutral viewpoint if their goal is to find a more crowded area or a more empty area.

My map technologies are split into two independent structures: the distributed spaces within the library that measure their own level of crowdedness, and the overlying canvas layer that combines all the data together into a visualization for a user. This makes it so that the library engineering team can use one or both of my structures modularly from each other. For instance, if a new technology comes along that measures crowdedness more accurately than wifi routers, the library can distribute the map into new partitions not made up by a voronoi diagram of wifi routers and assign a crowdedness level to each space, and my fog generation and animation technologies would seamlessly conform to the new structure.

From another perspective, the library can also keep my voronoi distribution of the floors based off the wifi routers and implement a new design to visualize crowdedness, if they find in further user testing that my designs were less effective than other possibilities such as colored heat maps.

The vital pillar of a multicommon is the existence of inhibitors within it, as a multicommon is both constructed *by* those within it, and *for* those within it. The utility of a library remains existent even while vacant, but its cultural environment is only activated by those occupying and using the utility's space. Gold's presentation described in section 1.1 shows

that an increase in library occupancy is, by the definition of a multicommon, a reality. More people go into the building each day, and the building retains each visitor for a longer period of time. This makes a proper visualization of population in the building vital for its utilities available to use, and vital for visitors' end goals of what environment they would like to experience in the space.

10. References

1. Gold, A. (2013, September). Open Culture at the Heart of the University: Libraries as Multicommons. OKCon: Open Knowledge Conference 2013. Geneva, Switzerland. Retreived June 11, 2015, from http://works.bepress.com/agold01/13

2. Hardy, S. & Kukla, R. (1999). A Paramount Narrative: Exploring Space on the Starship Enterprise. The Journal of Aesthetics and Art Criticism 57 2.

3. Beebe, G. (Ed.). (2014, February 25). Hunt, Carl W. Retrieved June 11, 2015, from http://lib.calpoly.edu/people/cwhunt/

4. Robert E Kennedy Library. (2015). Kennedy Library Maps [Computer software].

5. Google. (2015). Google Maps [Computer software].

6. Trujillo, C. (Ed.). (2015, February 25). Trujillo, Catherine. Retrieved June 11, 2015, from http://lib.calpoly.edu/people/ctrujill/

7. Liegl, C. (2014, March 8). Hacking for the library. Retrieved June 11, 2015, from http://lib.calpoly.edu/outloud/2014/03/hacking-for-the-library/

8. Aimetis. (2015). Aimetis People Counter [Computer software].

9. AXIS. (2015). AXIS Camera Application Platform [Computer software].

10. Aimetis People Counter Quick Tour. Retrieved June 11, 2015, from http://www.aimetis.com/Embedded/APCTour/APC.htm

11. Placemeter. (2015). Placemeter [Computer software].

12. About Us. Retrieved June 11, 2015, from https://www.placemeter.com/about

13. Placemeter at the WebSummit. (2013, November 14). Retrieved June 11, 2015, from https://vimeo.com/79246854

14. Jaffe, E. (2014, August 4). The View From Your Window Is Worth Cash to This Company. Retrieved June 11, 2015, from http://www.citylab.com/tech/2014/08/the-view-from-your-window-is-worth-cash-to-this-company/375471/

15 Placemeter at the WebSummit. (2013, November 14). Retrieved June 11, 2015, from https://vimeo.com/79246854

16. Density Inc. (2015). Density - The Heartbeat of a City [Computer software].

17 Launch Festival 2014: Keynote - Paul Graham, Y Combinator + Session 4 - Day 1. (2014, February 24). Retrieved June 11, 2015, from https://www.youtube.com/watch?v=0rVpAKziQJA

18 Higginbotham, S. (2015, May 22). Meet Density, a startup that lets you see if your favorite coffee shop is full. Retrieved June 11, 2015, from http://fortune.com/2015/05/22/meet-density-a-startup-that-lets-you-see-if-your-favorite-coffee-shop-is-full/

19. O'Connor, & Robertson. (n.d.). Georgy Fedoseevich Voronoy. Retrieved June 11, 2015, from http://www-history.mcs.st-and.ac.uk/Biographies/Voronoy.html

20. Aurenhammer, F., & Klein, R. (1996). Voronoi Diagrams. Handbook of Computational Geometry, 201-290. Retrieved June 11, 2015, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.7777&rep=rep1&type=pdf

21. 1854 Broad Street cholera outbreak. (n.d.). Retrieved June 11, 2015, from http://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak

22. DiMarco, M. (n.d.). UI Algorithms. Retrieved June 11, 2015, from http://markmarkoh.com/jsconf2014/assets/player/KeynoteDHTMLPlayer.html#43

23. Buchanan, L., Wallace, T., & Watkins, D. (2014, February 13). Long-Lived Greatness. Retrieved June 11, 2015, from http://www.nytimes.com/interactive/2014/02/13/sports/baseball/jeter-long-lived-greatness.html

24. Hill, R. (n.d.). Javascript implementation of Steven J. Fortune's algorithm to compute Voronoi diagrams. Retrieved June 11, 2015, from http://raymondhill.net/voronoi/rhill-voronoi.html

25. Beutel, A. (n.d.).Interactive Voronoi Diagram Generator with WebGL. Retrieved June 11, 2015, from http://alexbeutel.com/webgl/voronoi.html

26. Friend, S. (2015, March). Visualizing Relationships between Related Variables: Improving Physics Education through D3.js Network Visualizations. Retrieved June 11, 2015, from http://digitalcommons.calpoly.edu/laessp/43/

27. Halliday, J. (2013, April 10). Heatmap. Retrieved June 11, 2015, from https://github.com/substack/node-heatmap

28. (2011, March) CSU Website Guidelines, from http://www.calstate.edu/brand/resources/documents/toc_web_style_guide.docx

29. Section 508 of the Rehabilitation Act (29 U.S.C. 794d), as amended by the Workforce Investment Act of 1998 (P.L. 105-220), August 7, 1998

30. Commuter Forecast. (n.d.). Retrieved June 11, 2015, from http://www.weather.com/weather/map/interactive/l/l

31. 4.8.11 The canvas element. (n.d.). Retrieved June 11, 2015, from http://www.w3.org/TR/2011/WD-html5-20110525/the-canvas-element.html

32. PNPOLY - Point Inclusion in Polygon Test. (2014, January 21). Retrieved June 11, 2015, from http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html

33. Halliday, J. (2012, January 1). point-in-polygon. Retrieved June 11, 2015, from https://github.com/substack/point-in-polygon

34. 14 Clipping, Masking and Compositing. (n.d.). Retrieved June 11, 2015, from http://www.w3.org/TR/SVG/masking.html#Masking

35. Adobe. (2013). Adobe Illustrator CS6 [Computer software].

36. SVG2 Requirements Input. (2014, January 31). Retrieved June 11, 2015, from http://www.w3.org/Graphics/SVG/WG/wiki/SVG2_Requirements_Input#Gradient_along.2Across_stroke

37. SVG2. (n.d.). Retrieved June 11, 2015, from https://www.chromestatus.com/feature/5760616295825408