# Designing for Interaction and Insight:

# Experimental Techniques

# For Visualizing Building Energy Consumption Data

By Hetian Cao

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Computational Design

December 2017

## Abstract

While more efficient use of energy is increasingly vital to the development of the modern industrialized world, emerging visualization tools and approaches of telling data stories provide an opportunity for the exploration of a wide range of topics related to energy consumption and conservation (Olsen, 2017). Telling energy stories using data visualization has generated great interest among journalists, designers and scientific researchers; over time it has been proven to be effective to provide knowledge and insights (Holmes, 2007).

This thesis proposes a new angle of tackling the challenge of designing visualization experience for building energy data, which aims to invite the users to think besides the established data narratives, augment the knowledge and insight of energy-related issues, and potentially trigger ecological responsible behaviors, by investigating and evaluating the efficacy of the existing interactive energy data visualization projects, and experimenting with user-centric interactive interface and unusual visual expressions though the development of a data visualization prototype.



**Keywords**: building energy consumption, energy efficiency, water consumption, energy data visualization, data visualization, human-data interaction

# Acknowledgements

First and foremost, I want to express my sincere gratitude to my excellent thesis advisor, Dr. Daniel Cardoso Llach, for his remarkable patience, motivation, guidance and support, and for not giving up on me in my hardest moments. I could not have done it without his help.

I would like to thank my thesis committee member, Dr. Eddy Man Kim, for his encouragement, insightful comments and advice.

My sincere thanks also go to Dr. Vivian Loftness, for pointing me to the dataset and insightful perspectives for my thesis, as well as Dr. Dave Touretzky, Dr. Chris Harrison, Prof. Joshua Bard and Dr. Daragh Byrne for offering me great opportunities in their exciting projects during my time at CMU. I would also like to thank Ardavan Bidgoli and Pedro Veloso for their help with my research and study.

My thanks also go to my dear friend Noreen Saeed for believing in me;Robert Zacharias for your kindness, your great company and encouragement. And thank you Lu Han for being so caring and supportive. My thanks also go to Amit Nambiar, Qiaozhi Wang, Hexing Ren, Chaoya Li, and the rest of my CodeLab-mates, for all your support and inspirations, the deadline-approaching sleepless nights and happy memories.

Last but not least, to my dearest parents, for always being there for me all this time. I am so lucky to have you in my life.

# Table of Contents

Introduction.....1

1 Background.....3

1.1 Motivation/Significance.....3

1.1.1 Research Area.....3

1.1.2 Obstacles and Current Approach.....4

1.1.3 The Gap.....5

1.1.4 Motivation and Relevance.....6

2 Case Studies.....8

2.1 Case #1: Unusual Visual Expressions and Data Personalization.....8

2.1.1 Dear Data: the Project.....8

2.1.2 Emotional Connection and Storytelling.....9

2.1.3 Time-based Data Representations with Unusual Visual Languages Leading to More Active Engagement.....10

2.1.4 The unexpected efficacy of designing rules and legends..12

2.1.5 Left-to-right comparison.....15

2.2 Case #2: Evaluating the Efficacy of Existing Energy Data Visualizations....16
2.2.1 A Comparison Study of Interactive Maps for Energy Data....16
2.2.2 Evaluating the Existing Energy Data Visualizations....26
3 Hypothesis....28
3.1 Research Questions....28
3.2 Hypothesis....29
3.3 Methods....29
4 WaterViz: A Prototype....30
4.1 Data Processing....31
4.1.1 Acquiring Dataset....31
4.1.2 Analysing the Main Features of the Acquired Dataset....33
4.1.3 Parsing and Filtering Data....34
4.2 Data Visualization....38
4.2.1 Constructing Data Structures for Visualization....38
4.2.2 Topographic Viewport of the US....39
4.2.3 Detailed and Integrated Viewport of State Data....44
4.2.4 Comparison Viewport of Building Data....45
4.3 Designing Data Interaction and Experience....46
4.3.1 "Free-edit" Mode....46
4.3.1 Keyboard Interactions and Toggle Dashboard....49
4.3.1 Transaction between Viewports....51
5 Conclusions and Next Steps....51
5.1 Conclusions....51
5.2 Next Steps....52
5.3 Contributions....53

Appendix A: Evaluating the Efficacy of Existing Energy Data Visualizations.....54
Appendix B: Source Code .....56
Bibliography.....126

# Table of Tables

Table 1: Comparing features of data points in each visualization.....21
Table 2: Comparing features of Interaction in each visualization.....24
Table 3: Variables of Each Data Entry from the Original Dataset.....31

## Table of Figures

Figure 1 : A variation of visualizations from the project Dear Data.....10

Figure 2: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the data visualization in the theme of “clocks” by Giorgia Lupi.....11

Figure 3: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the data visualization in the theme of “clocks” by Stefanie Posavec.....13

Figure 4: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the formatting of comparing the same data subject on the same viewport.....15

Figure 5: Screenshot of the US Map of the“Mapping How the United States Generates Its Electricity” project.....17

Figure 6: Screenshot of the “Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC” Project.....17

Figure 7: Screenshot of the Philadelphia 2017 Building Energy Benchmarking Project (Azavea Inc., 2017).....18
Figure 8: Screenshot of the US Map in the project “The United States of Energy 1.0”.....19
Figure 9: Screenshot of the US Map Divided by Region in the project “The United States of Energy 2.0”.....19
Figure 10: Screenshot of the state profile after clicking on the region in the US Map.....20
Figure 11: Screenshot of the Legend Page of the project “The United States of Energy 2.0”.....20
Figure 12: Icons that represent different energy sources from the project “Electric Generation in Spain - Latest 24 hours”.....27
Figure 13 : File Structure of the Original Dataset.....31
Figure 14: refined file structure with data sorted by year.....35
Figure 15: Extracting Individual Building Profiles.....36
Figure 16: The process of importing and aligning the topographical map of US from 50 separate models [1].....40
Figure 17: The process of importing and aligning the topographical map of US from 50 separate models [2].....41
Figure 18: The process of importing and aligning the topographical map of US from 50 separate models.....42
Figure 19: Showing water state data on the topographical map viewport.....42
Figure 20: Showing water and gas state data on the topographical map viewport.....42
Figure 21: Showing water, gas, and electricity state data on the topographical map viewport.....43
Figure 22: First Impression of the detailed viewport.....44

Figure 23: User can select buildings and arrange them freely in the comparison viewport.....45
Figure 24: “Free-edit mode” allows the users to manipulate the objects and visualizations freely in the first viewport.....46
Figure 25: “Free-edit mode” allows the users to manipulate the objects and visualizations freely in the first viewport.....46
Figure 26: “Free-edit mode” allows the users to manipulate the objects and visualizations freely in the second viewport.....48
Figure 27: Using the metaphors of “Popsicle sticks” or file handles encourage the users to engage in free editing.....48
Figure 28 and 29: the instructions for keyboard interactions on the up-left corner of both viewports.....50
Figure 30: the keyboard interaction of the left hand(WASD), AND RIGHT HAND (< and >).....50
Figure 31: the toggle dashboard in the topographic viewport.....51

# Introduction

Each and every day in countless buildings around the world, data of many aspects of energy use and consumption is monitored, collected and processed, adding to the increasingly massive and pervasive network of energy data. To fully understand the complex concept that is energy use and consumption seems like a difficult task as well as a persistent goal for researchers, engineers and policy makers, not to mention people who have limited knowledge and background in related areas.

However, in order to achieve the goal of saving energy in a greater scale, at the same time helping people with taking their ecological responsibilities, this task of breaking down and understanding energy consumption data has become the precondition and priority. In other words, in order to more effectively reduce the consumption of energy and keep the effort on course, people need to be presented with the means and resources to get the knowledge and insights of their own relationships with energy, the nature of the energy conservation

problem, and the actions to be taken to achieve the goals, on the personal level and as a community.

Although more and more people have realized the importance of saving energy and are making an effort in different aspects of their daily life, they don't always have a good enough perception of the impact they can make through their decisions and behaviors on a daily basis, therefore it eventually limits the possibilities and potentials to be fully engaged, create a long term impact, and make decisions that can reflect their environmental consciousness (Loftness, et, al., 2017). Moreover, environmental scientists and building researchers are facing the obstacle of getting the message across to actors that can directly influence the construction of infrastructures, environmental policies, and distribution of the resources across the country .

And if that is the case, how can the goal of higher level sensemaking be achieved? Could data visualization be part of the answer? Can we visualize and represent the energy data in a manner that is effective enough to move closer to the goal of understanding the patterns and valuable information hidden in the numbers? And how does one design the interaction and experience with the human in mind, so that it can be sufficiently convincing to influence decision making, behavior and policy, on the personal and institutional level on a day-to-day basis? And these are some of the questions this thesis attempts to discuss and experiment with looking for a solution from an information design perspective.

# 1 Background

## 1.1 Motivation/Significance

### 1.1.1 Research Area

Using data visualization in storytelling and interaction has generated great interest among designers, journalists and scientific researchers in the area of energy consumption and conservation, and over time it has been proven to be effective to carry huge amount of data and logic, and has great potentials of human-data interactions when explaining complex concepts and demonstrating detailed, unconventional mechanisms and processes, comparing to verbal or writing, etc.

### 1.1.2 Obstacles and Current Approach

Despite the increasing attention brought to environmental and energy problems, obstacles against the effective propagation of the knowledge and insights of energy consumption and conservation are plentiful.

The environmental coverage in the mainstream media is far from ideal to effectively inform the public. As pointed out by an Inaugural Ranking Report completed by the Project for Improved Environmental Coverage (PIEC) (Miller and Pollak, 2013): “On average, entertainment headlines get over three times more coverage than environmental stories for nationally prominent news organizations.” There is much room and potential for innovation and development of energy and environmental stories in the media, and strategies like increasing the visibility of environmental stories and focus on solutions can be helpful to some extent.

So how come it is challenging to tell environmental stories? In order to understand the bigger picture, which is often necessary when it comes to environmental changes across decades even centuries, it can be more difficult to get to the eventual point and make an argument. And the lack of established knowledge and the complexity of certain terms also contributes to the an obstacle of maintaining the current awareness and attention of the audience. If the way the information is presented fails to be intuitive or easy to grasp, the chances of reaching a bigger audience while trying to get across an idea is rather slim.

Meanwhile, the public suffers tremendously from the lack of accurate information and mixed messages in the existing news reports.that scientific consensus that lead to the conclusion of the existence of climate change and the effects of human activities are dismissed by misleading comments in the media coverage (Miller and Pollak, 2013; Huertas and Adler, 2012). While we urge the news corporations to produce fairer, science-based news coverage, it is helpful to acknowledge the challenges of telling an environmental story, as well as the necessity of looking for solutions and alternatives from the human-data interaction and information design perspective.

### 1.1.3 The Gap

For people who are not under the immediate threat of their environmental conditions, they may not have the sentiment and intention to take actions even if they understand it is a good thing to do. The gap between things people consider with higher priorities as well as better rewards in their daily life, and the hardly recognisable environmental consequences and so little reward of environmental conscious behaviors, when accumulated, could lead to a much slower progress in energy conservation.

Also, the inefficient communication among scientific researchers, the public and policymakers creates the problem of unbalanced information, that there is a gap between the perception of certain environmental problems that even if they agree there is a problem

presented, they have different opinions on solutions and consequences.

There is also the gap between the goal of data representation along with the designated narrative, and the level of understanding by the viewer, in other words, how much of the information presented can actually engage, make an impression, furthermore be internalized after the data experience. And the thesis attempts to add to the discussion of using data visualization techniques to close these gaps.

### 1.1.4 Motivation and Relevance

Researchers are constantly looking for better solutions to utilize natural resources, reduce the consumptions, including the consumptions of water, gas, electricity, without compromising the normal activities of the building occupants.

At the same time, building investors and corporations are gradually becoming more conscious and sensitive about information concerning energy consumption and building efficiency, that investing in a more energy-efficient building and infrastructure can be profitable, sustainable and beneficial in the long term, from both the standpoints of business development and cultural influence. According to a 2012 study, by retrofitting buildings for energy efficiency, the energy saved over 10 years could total more than $1 trillion; more than 600 million metric tons of CO2 per year could be mitigated, more than 3.3 million new direct and indirect cumulative job years can be created (excluding induced) in the United States

economy (Fulton et al., 2012). One can argue that the investment of improving the energy efficient features can lead to not only more sustainable, robust infrastructure, but also a more trusting and stable relationship with clients and occupants.

In the meantime, the increasingly alarming consequences of climate change and global warming around the world has brought the conversation of energy consumption and conservation into the limelight of political debates, journalism and social studies. In order to have more fair, well-informed and deeper discussions in these context, it is essential for more and more people to learn and understand the cause and effect of climate change and global warming as well as human behaviors as a society, in order to stay engaged, factual and vigilant. When people are better informed with scientific facts and research, they are given a more solid ground and potentially open community when advocating for their own values and beliefs, therefore their values regarding energy consumption and conservation can be better represented by the state and the government.

Having a better way to turn energy data from the scientific world into visual languages that can allow people of different backgrounds to understand has never been more important. Effective energy data visualization could be an efficient tool for encouraging ecological behaviors in the short and long term.

# 2 Case Studies

## 2.1 Case #1: Unusual Visual Expressions and Data Personalization

### 2.1.1 Dear Data: the Project

This case study intends to look into the collaborative project: Dear Data (Posavec and Lupi, 2016a), by designer Stefanie Posavec and information designer Giorgia Lupi. Dear Data is an one year-long, analog data drawing project. Each week during the year in progress, a particular type of data that reflects a certain aspect of the designers' lives, would be chosen, collected and measured from the real life events of the week. Then the collected data are turned into a piece of data visualization that is hand drawn on a postcard, which is then sent to the other collabrater through mail.

The journey of the postcards that carries the data visualization drawing of the week is a long and relatively old-fashioned one, with the traces of time and transportation, which is described as "a type of 'slow data' transmission"(Posavec and Lupi, 2016b). And the creative

approaches in which the data are represented and introduced in the project are inspirational on many levels, for augmenting the degree of engagement and interaction in designing energy data visualizations.

### 2.1.2 Emotional Connection and Storytelling

As introduced in the project homepage of Dear data:

> ***"We've always conceived Dear Data as a "personal documentary" rather than a quantified-self project which is a subtle – but important – distinction. Instead of using data just to become more efficient, we argue we can use data to become more humane and to connect with ourselves and others at a deeper level."***

The project documents the process of two designers getting to know each other through the unique way of sharing their lives through data, and each postcard is independently a piece of art but together they present a comprehensive picture of different aspects of the designers' lives. Just like reading a diary, readers can get to know them when going through the visualizations. It's a transformation from getting the information and visual appreciation of one aspect but to put them in place and form a deeper understanding of a concept of a higher level and complexity.

### 2.1.3 Time-based Data Representations with Unusual Visual Languages Leading to More Active Engagement

The nature of personal data in a specific time frame of a week makes small multiples an obvious choice when visualizing data. The project is inspirational for how data and unusual visual expressions can be correlated and experimented to create data stories and narratives.

Figure 1 : A variation of visualizations from the project Dear Data

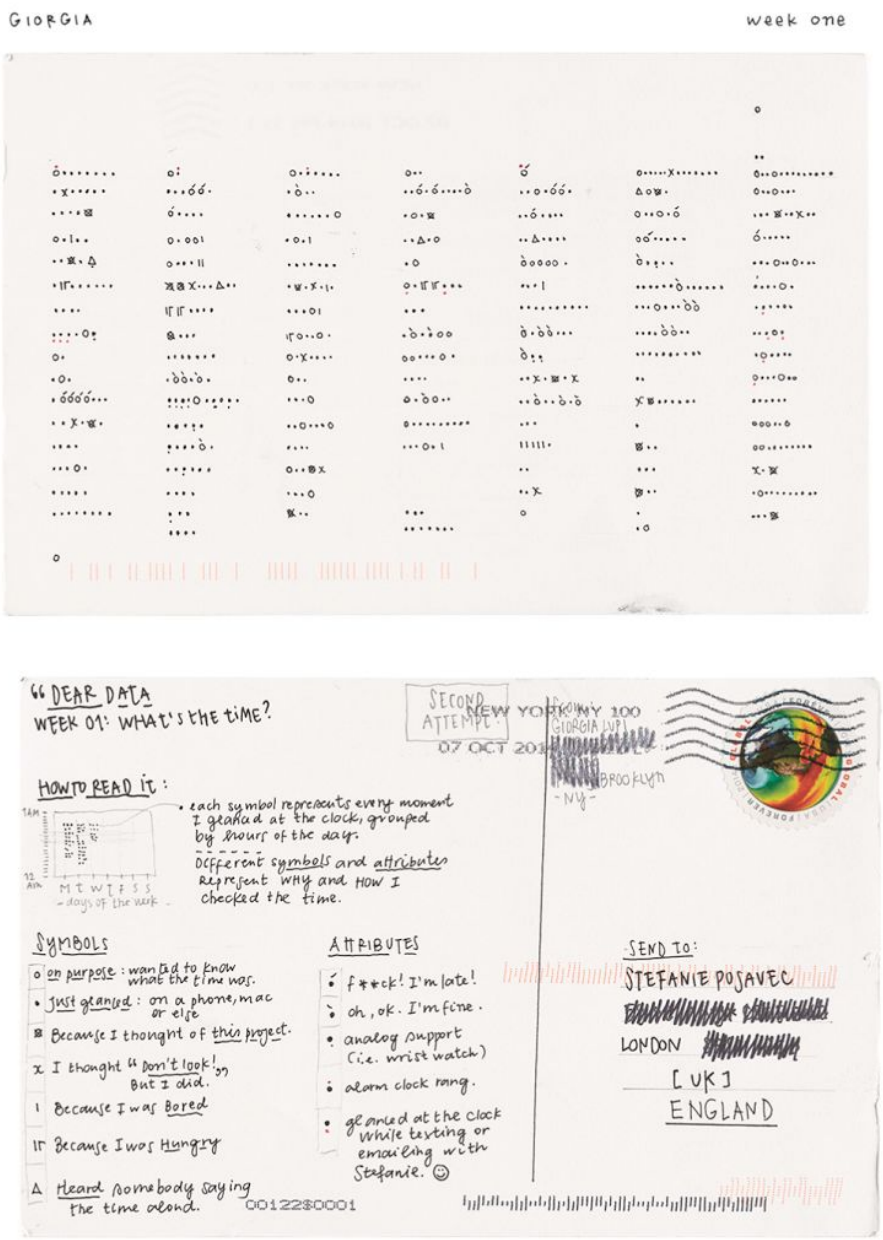


Figure 2: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the data visualization in the theme of "clocks" by Giorgia Lupi.

### 2.1.4 The Unexpected Efficacy of Designing Rules and Legends

As introduced in the project homepage of Dear data:

> ***"On the front of the postcard there would be a unique representation of our weekly data, and, on the other side (in addition to the necessary postage and address), we would squeeze in detailed keys to our drawings: the code to enable the recipient to decipher the picture, and to fantasize about what had happened to her new friend the week before."***

One of the most intriguing experience of exploring the postcards are deciphering the rules. By visualizing data with unusual shapes that and rules in the legends, readers are prone to learn and adapt in order to make sense of the visualization, giving them more opportunity and incentives to get to know how the data is presented, comparing to conventional charts and graphs, which people often assume they know how to read therefore can misunderstand when not reading carefully, the data has personalities, therefore, it's own signature.


This week Giorgia and Stefanie tried gathering data in small notebooks (tedious), but soon switched to making notes on their phones (much easier). Stefanie's favourite clock to capture: a bell tolling the time in a town in Devon.

Figure 3: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the data visualization in the theme of "clocks" by Stefanie Posavec.

However, even if the rules are often simple, they sometimes still requires careful reading and investigation. But once the first several

minutes pass, the reader would having a much better time and immersive experience understanding the data.

Creating the right rules can not only enhance the visual identity of the visualization, but better engage with viewer by encouraging them to make effort. But the precondition would be that the rules are clear and well-explained, or else it would be counterproductive.

When the viewer first comes across the visualization that doesn't resemble their expectations, they are likely to be curious about the data and how it's represented. Then the next thing they would look for is the legends, in other words, the rules to decipher the true meaning behind the new visual expression.

The process of "deciphering" would take several minutes at most, while it seems to take longer to understand comparing to conventional visualization methods, therefore the system provides the viewer with a more steep learning curve, which can result in a better, more deep understanding and memory in the information visualized.

After making an effort to learn, even memorize the rules while reading back and forth between the visualization and the legend, in order to understand the details and the whole picture, the viewer would more likely pay more attention and spend more time on the visualization, allowing the information to digest and have a better chance turning into knowledge and insight.

### 2.1.5 Left-to-right Comparison

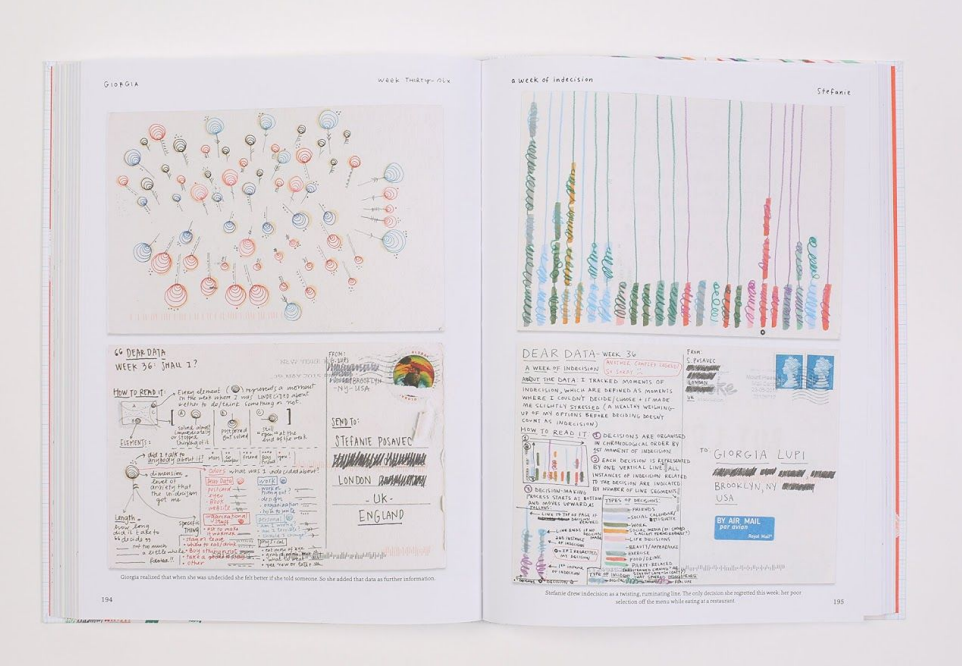


Figure 4: A page from the book Dear Data (Posavec and Lupi, 2016a), showing the formatting of comparing the same data subject on the same viewport.

For readers, it's a more interesting experience to see how two different artists visualize and process data entirely differently. And the left and right page composition makes it easier to get a better picture of the whole exchange of the conversation. At the same time it invites the reader to participate and record their own data in their own ways using simply a postcard and pencils while providing a comprehensive example of how data visualization can help one understand oneself and others in unexpected and surprising ways.

## 2.2 Case #2: Evaluating the Efficacy of Existing Energy Data Visualizations

This case study is intended to look into the patterns of the choices designers made in the existing energy data visualizations. To understand how data visualization techniques and methods are applied in the narration of energy and environmental stories, the visualizations selected from a larger pool of projects and have been evaluated from the perspectives of visualization techniques, human-data interaction, and narration of energy stories. The full report can be found in appendix A.

### 2.2.1 A Comparison Study of Interactive Maps for Energy Data

The following projects are selected from a larger pool of projects of energy data visualization. These selected projects all consist of a territorial map in the scale of city to country, in the narratives of the visualization, visualizing energy data with cursor based interaction features on a webpage. Chapter 2.2.1 is a paralleled comparison study for the existing methods of visualizing geolocation energy data.

#### 2.2.1.1 Selected Projects

Project #1 Mapping how the United States generates its electricity (Muyskens, Keating and Granados, 2017)

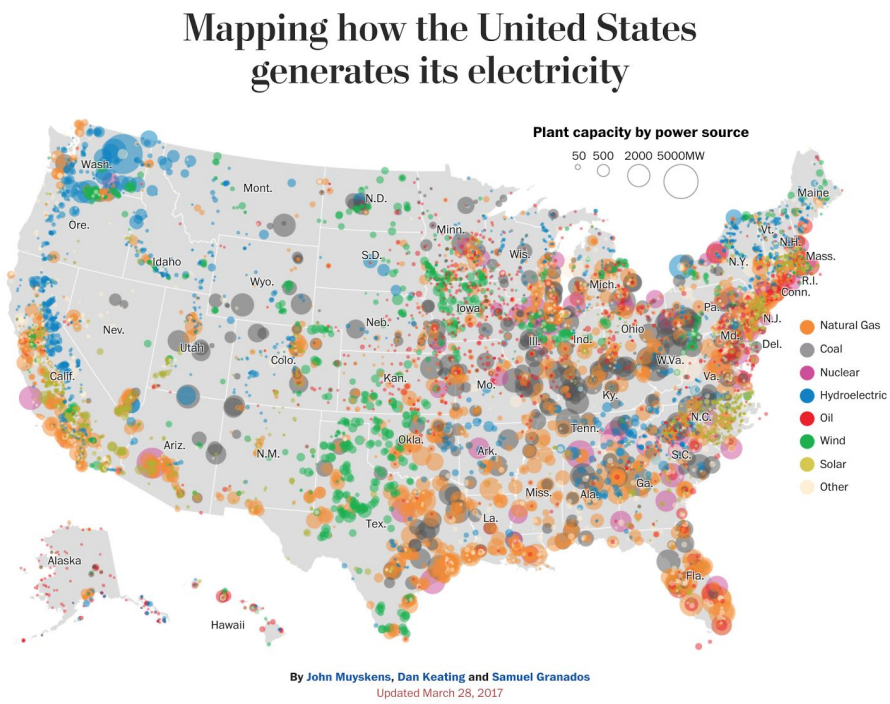


Figure 5: Screenshot of the US Map of the"Mapping How the United States Generates Its Electricity" project

## Project #2 Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC (Sherpa et al., n.d.)

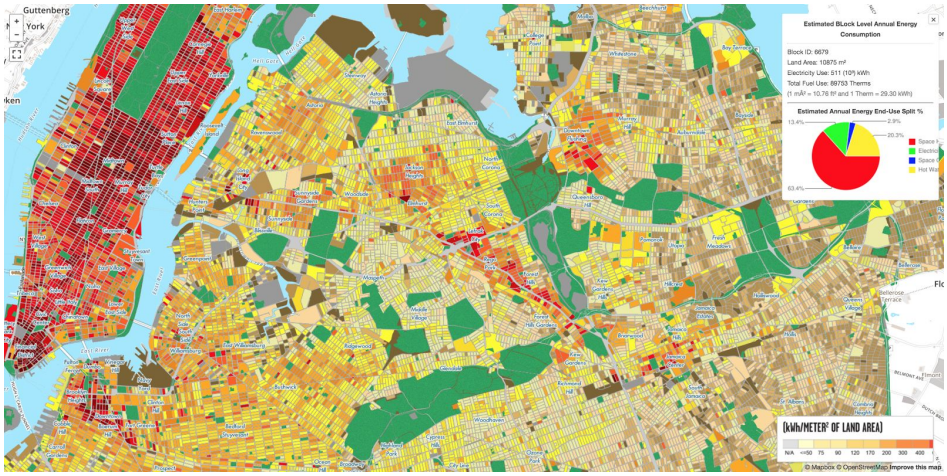


Figure 6: Screenshot of the "Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC" Project

Project #3  Philadelphia 2017 Building Energy Benchmarking (Azavea Inc., 2017)

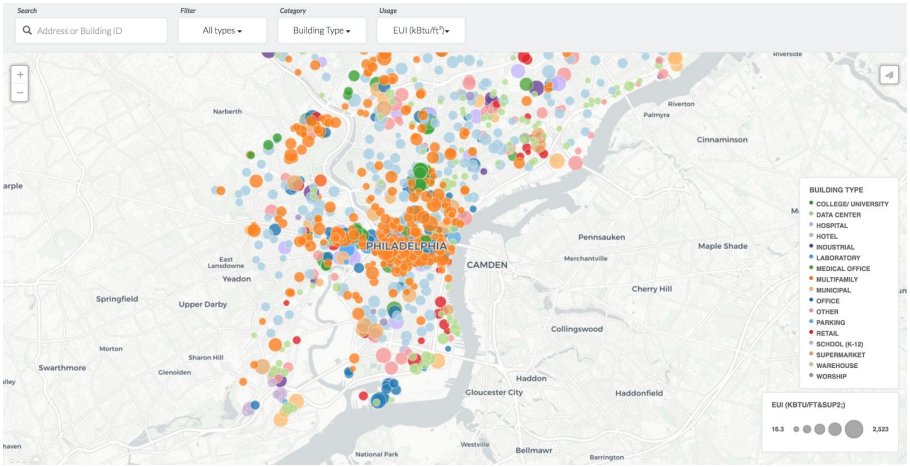


Figure 7: Screenshot of the Philadelphia 2017 Building Energy Benchmarking project(Azavea Inc., 2017)

Project #4 The United States of Energy (Saxum, 2017)

Project #4a  The United States of Energy 1.0 (Saxum, n.d)

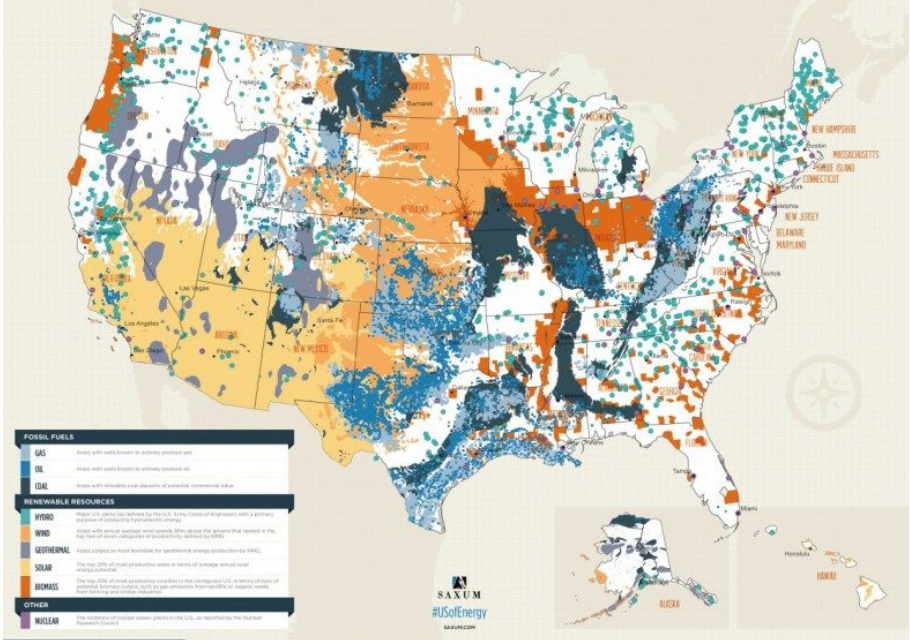


Figure 8: Screenshot of the US Map in the project "The United States of Energy 1.0"

Project #4b  The United States of Energy 2.0 (Saxum, 2017)

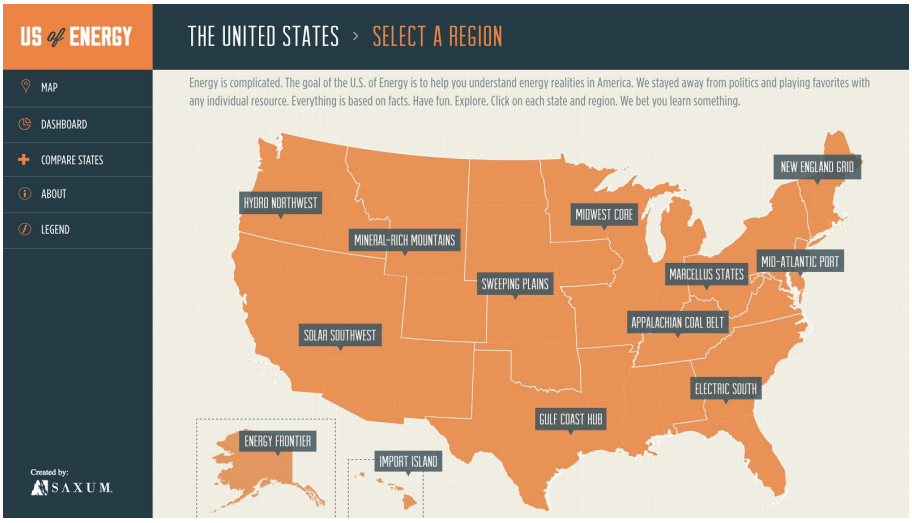


Figure 9: Screenshot of the US Map Divided by Region in the project "The United States of Energy 2.0"

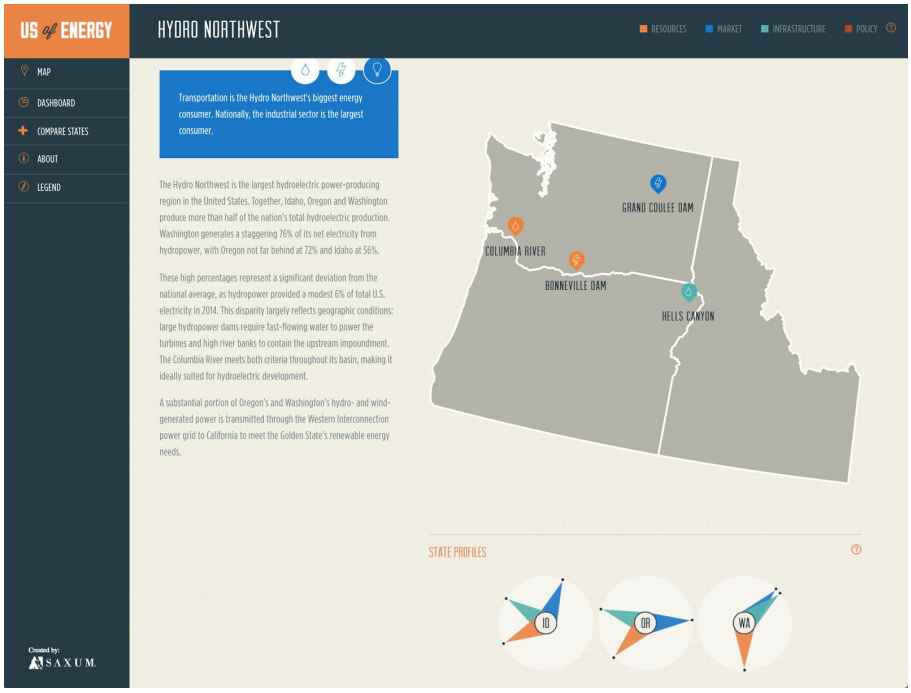


Figure 10: Screenshot of the state profile after clicking on the region in the US Map

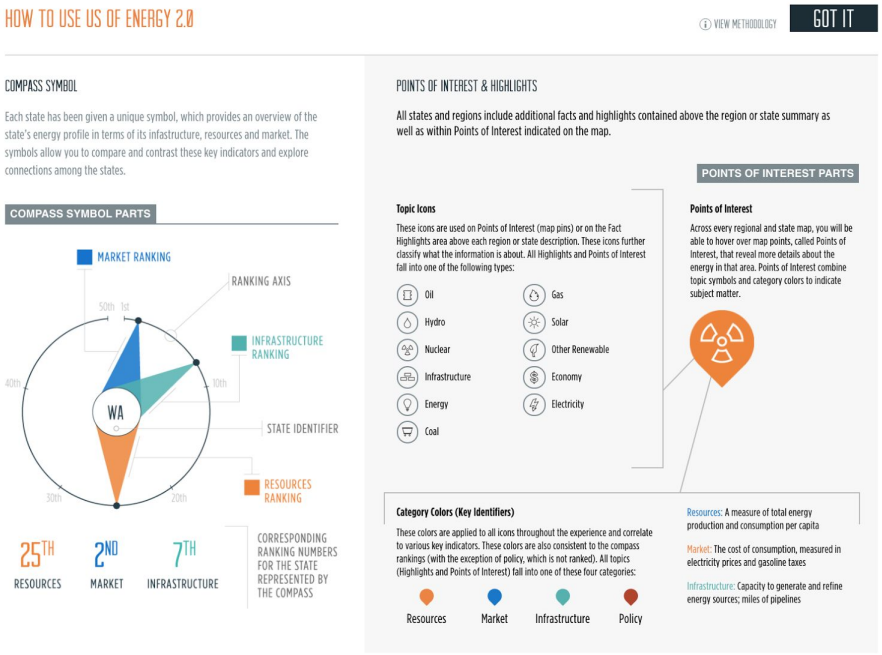


Figure 11: Screenshot of the Legend Page of the project "The United States of Energy 2.0"

#### 2.2.1.2 Comparing features of data points in each visualization

| Data Visualization Project | Representation of Data Points | Categorization Method |
|---|---|---|
| Mapping how the United States generates its electricity | Position: center of circle<br>Subject: Power plant data<br>Data value: Area of circle represents the plant capacity to generate electricity | Distinguished colors for each energy type |
| Philadelphia 2017 Building Energy Benchmarking | Position: center of circle<br>Subject: Building data<br>Data value: Area of circle represents plant capacity by power source | Distinguished colors for each building type |
| Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC | Subject: separated by outline of block or lot | Intensity of colors (heatmap) for electricity usage rate for each block and lot |
| The United States of Energy 1.0 | Subject: position or area of energy produced | Distinguished textures and colors for each energy type |
| The United States of Energy 2.0 | regions in the US | marked with the name of the region next to the data point |

Table 1: Comparing features of data points in each visualization

Project 1 and 3 are using circles to represent data points. The advantage is that it's really intuitive for users to connect the size of the circle to the amount of energy generated or consumed, as well as the radiation or significance of what the data point represents from its center, which pinpoints the location of the data point.

At the same time, the data used in project 1 and 3 is ordered and all data points have the same subject. In project 1, the data point represents the amount of electricity generated for each plant, differentiated by the capacity, location and the type of resource used; project 3 has the subject of energy consumed in each building, differentiated by the type of building from which the data is collected. And it can be another reason that the entity of a circle (which consist of a center that represents its position, a redian that indicates the size, and a color that can be used to categorize) can be effective enough to represent an energy data point with a single subject in the scale of the country, directing users to observe the visualization to the relevant variables represented by these entities, while construction a general image of the distributions and tendencies of these data points across the map.

On the other hand, similar to project 3, project 2 is on the city scale, visualizing electricity consumption in blocks and lots in New York City area. However, it represents the data points with the area of the outline of the buildings instead of a circle like project 3. A reason could be that the data acquired in project 2 is much more complete and precise, covering almost every block of the entire area, and by outlining them equally precise, each building is defined with a unique shape and profile, providing users with not only the location, but other information that indicates the properties of the block and its surroundings.

#### 2.2.1.3 Comparing features of Interaction in each visualization

Since the target projects are in the form of web application designed for web browsers, the main focus of the interaction is using mouse gestures. Therefore in projects whose maps were created using the google map or similar plugins, the interactions share the following rules, fully or partially, as follows (Google, 2017a; Google, 2017b):

- Zoom control: to change the zoom level of the map.
    - click "+" and "-" buttons in the corner of the map;
    - double click using the mouse or trackpad;
    - scroll using the mouse or pinch using the trackpad;
- Pan/drag control: the action of dragging a map while keeping it at the same scale.
    - click and hold the mouse or trackpad, then drag the maps by moving the mouse or finger. Release to stop panning.
    - Click arrow keys: up for north, right for east, down for south and left for west.
- Fullscreen control: click the fullscreen button in the corner of the map to open the map in fullscreen mode.

And in order to simplify, in the following table comparing the interaction features, the map interactions mentioned above are collectively referred as “general map controls”.

| Data Visualization Project | Hovering | Clicking | Dragging |
|---|---|---|---|
| Mapping how the United States generates its electricity | None | None | None |
| Philadelphia 2017 Building Energy Benchmarking | None | Click on: Data point<br>Action: Popup Window showing detail of datapoint | General map controls |
| Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC | None | Click on: Data point<br>Action: Popup Window showing detail of datapoint | General map controls |
| The United States of Energy | (On legend) display one type of energy | None | None |
| The United States of Energy 2.0 | Change color | Enter detailpage of clicked Region | None |

Table 2: Comparing features of Interaction in each visualization

Aside from the scale of the map as mentioned in the last chapter, another similarity between project 2 and 3 is that they both use the interaction of opening a pop up window upon clicking on the individual data points. The reason could be that both datasets consists of multiple dimensions, and both visualizations made the decision of visualizing the most important dimension on the default map, and by arranging the secondary dimensions in a pop up window,

which only displays when users want to get into more details about a single data point by clicking on it. By doing so it extends the flow of the narrative into a more detailed level, such as showing numbers that are too long to remember, reinforcing the understanding of the data point from the map view.

Although these interaction features are helpful to understand energy data, there can be limitations. For example, it might be helpful for users to compare specific data points to experiment with their own narratives, reinforcing the perception from their personal experience, from which they would learn about certain aspect of the dataset, specific knowledge and insights. However, with the interaction features such as clicking back and forth between data points, moving around on the map, etc., it creates a barrier for users to focus on the questions they want to explore and find patterns and clues from the data points. Therefore the exploration aspect of the data experience could be undermined.

### 2.2.2 Evaluating the Existing Energy Data Visualizations

There are two major similarities among the visualizations for location based data with geolocation maps. The project all used color code to differentiate categories for better comparison (e.g. kind of energy source, kind of building, etc.), and the major Interactions are clicking on data points and popup windows when hovering/clicking displaying details of certain data points.

In terms of visualization techniques and energy data characteristics:

- There is a strong relation between geolocation-based energy data and certain visualization techniques like representing data using geographical map to show relative locations, heatmap to show energy consumption rate, etc.
- For projects that visualize a bigger scale of energy data, the categories of energy are more likely to be comprehensive, which is related to the fact that the goal of these visualizations are likely to present a broader picture, or the history (timewise), distribution (location-wise) of energy data in certain areas.
- For projects that visualize a smaller scale of energy data, the categories of energy included mainly depend on the goal of the project. In these projects, geolocation-based data tend to have the goal of community engagement and tend to visualize the area that has personal relations with the intended users. For energy data closer to the household or personal scale, visualized data are more likely to be real-time and with a more concentrated goal of improving the environmental consciousness of a small group of people.

In terms of human-data interactions:

- For most of the projects that are web applications, cursor-based interaction are the primary method to interact. The cursor-based interactions include dragging to zoom / move, click to highlight, hover to highlight, and other conventional cursor gestures on geographical maps.
- For projects that have physical entities like the Power Award Cord, the interaction can be entirely different. Physical interactions can not only be functional and serve as real-time notifications and dashboard, but they can also in some cases contribute to the understanding and reinforce the perception of data through sensory experience.

In terms of visual expressions:

- It is common among the project to use color or color gradients as indications of quantities of data from the same category; and it's also common to use different color to differentiate energy sources;
- Some projects use icons that serve as the metaphor of certain type of energy, combined with color coding to distinguish an energy source from another.


Figure 12: Icons that represent different energy sources from the project "Electric Generation in Spain - Latest 24 hours"

# 3 Hypothesis

## 3.1 Research Questions

From the case studies there are several questions that I find important to ask:

How to cultivate a personal relationship between human and energy data using data visualization techniques?

What can be done to raise the ecological awareness from a design perspective?

And how to create a better human-data interaction for a more immersive and exploratory experience?

By asking the questions above I formed my research question as the following:

***How to use data visualization and information design techniques and technology to design experience for human-energy data interaction, and tell environmental stories to trigger ecological responsible behaviors?***

### 3.2 Hypothesis

Using data visualization techniques, especially using user centric interactive interface and unusual visual expressions can be effective in designing better experience for human-energy data interaction to trigger ecological responsible behaviors and insights.

### 3.3 Methods

This thesis attempt to test the validity of the hypothesis through the following methods:

- Case studies for current energy data visualization projects across different scales, investigating and evaluating the efficacy of the existing interactive energy data visualization projects;
- experimenting with user-centric interactive interface and unusual visual expressions though the development of a data visualization prototype.
- Designing an interactive experience for building energy data with the focus of time-based and geolocation-based water consumption data;
- Evaluating and analysing the design process and thinking methods used in the prototype.

# 4 WaterViz: A Prototype

Loosely based on the major data visualization process from the book: Visualizing Data (Fry, 2008), the process of developing the prototype contains the following steps: acquiring, parsing, filtering, mining, representing, refining and interacting.

## 4.1 Data Processing

### 4.1.1 Acquiring Dataset

The dataset is acquired from the EUAS application, a web based system which is used for tracking energy details for various energy sources (e.g. electricity, natural gas, oil, chilled water, steam and renewable energy, etc.) (General Services Administration, 2015). It serves Energy Center of Expertise, under the Office of Facilitates Management and Service Programs. The dataset containing 30 variables for the research of energy consumption has been collected from more than 1400 service centers across the US during the time of January 2003 to July 2016. The buildings are initially divided into 11

regions, later expanded into 13 regions, according to the geolocation information, and each region includes buildings are from the same or adjacent states.

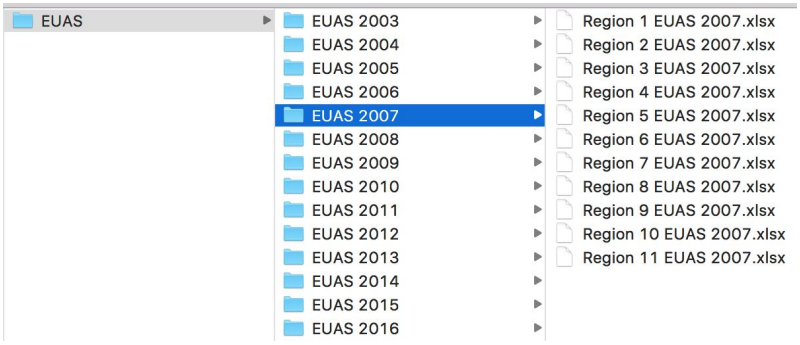


Figure 13 : File Structure of the Original Dataset

The dataset includes 14 folders, each contains all data from all months of each year from January 2003 to July 2016. In each folder, all data from the same year are sorted into several (between 11 and 13) xlsx files, according to the regions the buildings are located, as shown in figure 13. In each xlsx file, each entry of the building data are listed and sorted according to fiscal month. Each entry includes the following variables:

| | |
|---|---|
| Region No. | the region of the service building is put |
| State | which State the service center is located |
| Service Center | Descriptions of the service center |
| Fiscal Month | the month in which the data entry was recorded |
| Fiscal Year | the year in which the data entry was recorded |
| Building Number | a unique ID for the service center |

| | |
|---|---|
| Area Field Office | the field office that the service center is responding to according to its location |
| Cat | categories of the service center |
| Building Designation | building designations of the service center |
| Gross Sq.Ft | the area of the service center |
| Electricity (KWH) | electricity consumed to maintain the activities of the service center during the month recorded |
| Electricity (Cost) | electricity cost |
| Demand (KW) | electricity demanded to maintain the activities of the service center during the month recorded |
| Demand (Cost) | Demanded electricity cost |
| Steam (Thou. lbs): | Steam consumed to maintain the activities of the service center during the month recorded |
| Steam (Cost) | Steam cost |
| Gas (Cubic Ft) | Gas consumed to maintain the activities of the service center during the month recorded |
| Gas (Cost)* | Gas cost |
| Oil (Gallon) | Oil consumed to maintain the activities of the service center during the month recorded |
| Oil (Cost) | Oil cost |
| Chilled Water (Ton Hr) | Chilled water consumed to maintain the activities of the service center during the month recorded |
| Chilled Water (Cost) | Chilled water cost |
| Renewable Electricity (KWH) | Renewable electricity consumed to maintain the activities of the service center during the month recorded |
| Renewable Electricity (Cost) | Renewable electricity cost |
| Renewable Gas (Cubic Ft) | Renewable gas consumed to maintain the activities of the service center during the month recorded |
| Renewable Gas (Cost) | Renewable gas cost |

| | |
|---|---|
| Other (mmBTU) | Other energy consumed to maintain the activities of the service center during the month recorded |
| Other (Cost) | Other energy cost |
| Water (Gallon) | water consumed to maintain the activities of the service center during the month recorded |
| Water (Cost) | Water cost |

Table 3: Variables of Each Data Entry from the Original Dataset

The original dataset described above provided a comprehensive picture of the multiple forms of energy consumption data for each building. Although the data acquired are very comprehensive and orderly, several variables are absent from the start to the end of the time frame, therefore further filtering is necessary to extract consistent variables for visualization.

### 4.1.2 Analysing the Main Features of the Acquired Dataset

To effectively visualize the acquired building data, we must start by identifying the characteristics in order to find the compatible data visualization methods. The following aspects of energy data are distinguishable and can be seen as the starting point of getting into the data visualization process:

- The data is time-based, location-based and multi-dimensional

  Data collected in this project is based on concentrated and multi-dimensional data structure across a relatively long period of time.

- The data structures are orderly yet with inevitable inconsistency

  Since the data collection process requires great amount of time and resource, the categories, frequency and other aspects of the data collected are planned before the data was collected, as well as the features of the subject building. At the same time, since the data collection process has a relatively longer timeframe, sometimes it can be challenging to coordinate and integrate data collected from different sources, departments and methods. Therefore some part of the data can get missing and mixed up, resulting in inevitable inconsistency and incompletion in the dataset.

- The data have clear patterns related to several factors:

  One of the important goal of the data collected is for quantitative analysis and simulation, data collected in these projects could have a better possibility of having distinctive relations to the intended research questions. The dataset has a clear narrative that is essential for the proving of certain arguments, which provides great opportunities for creating the narratives of the visualization.

#### 4.1.3 Parsing and Filtering Data

##### 4.1.3.1 Merging into Simpler File Structure

Before filtering the unusable and incomplete data entries and variables, the xlsx files need to be broken down into manageable

sizes. At the same time, since the files are sorted by region, which is an artificial concept designed for easier organization of data, while considering the task of visualizing building data in the scale of the country, it would be more intuitive and straightforward to sort the data according to states, since it's more likely to be the established conventional knowledge of more viewers. Therefore, by merging all xlsx files sorted by region in the same year folder, the file structure can be simplified as shown in Figure 14.

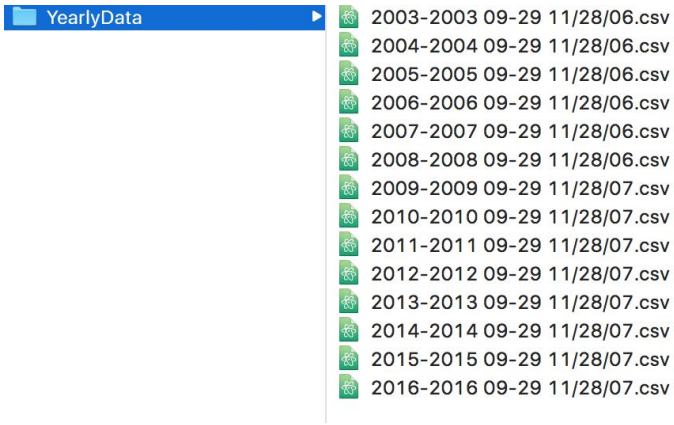


Figure 14: refined file structure with data sorted by year

#### 4.1.3.2 Extracting Individual Building Profiles

In order to prepare the visualization with multiple viewports, creating a combination of overview and detail, it would be helpful to rearrange the dataset from yearly data to individual building data. At the same time, variables that are absent and unusable would be eliminated when creating new files for each building. After this step there will be a folder of 1700 tsv files containing usable data from each building. Each file is named according to the unique building id

and the gross floor area, which are constant values for each building and make it easier to extract these information from file names and use in the visualizations. The file structure of the result of this step is shown in Figure 15.

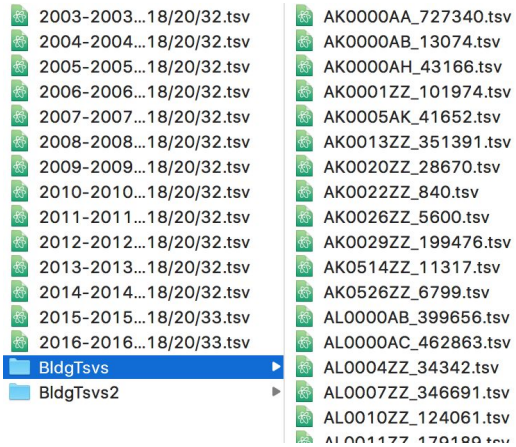


Figure 15: Extracting Individual Building Profiles

#### 4.1.3.3 Filtering Usable Building Data

When observing some samples in the dataset of individual buildings, it was found that each building data has its different level of completeness. Therefore a further filtering process is necessary to make sure the data material that is used for visualization is not incomplete and has fewer imperfections. By looping through the entire building profile folder, 4 buckets are created to contain 4 Tiers of completeness of building data:

Tier 1: Data entry with water, gas and electricity data all zero exists in the building profile

Tier 2: Data entry with two zeros in the data of water, gas and electricity exists in the building profile

Tier 3: Data entry with one zero in the data of water, gas and electricity exists in the building profile

Tier 4: Data entry with no zero in water, gas and electricity data exists in the building profile

And after sorting and grouping the four tiers of buildings, here is the results:

Tier 1 (3 zeros): 99 buildings;
Tier 2 (2 zeros): 410 buildings;
Tier 3 (1 zero): 632 buildings;
Tier 4 (0 zero): 4 buildings.

Building data from each tier can be used for different purposes in the future visualization from detailed comparison to general visualization across the country.

## 4.2 Data Visualization

### 4.2.1 Constructing Data Structures for Visualization

After data is rearranged, prepared and cleaned, creating the bridge between data points and graphic tools for visualization is the next step, in other words, data need to be prepared in certain format that is suitable and efficient to the visualization tool, so that the data processing process can be simplified when visualizing. Due to the multi-dimensional characteristics of the dataset, the format of JSON would be a fair choice. After this step, in each of the json file that contains data from each building in all years:


***{***
***"bldgData":[***
***{"water":2.7448,***
***"index":0,***
***"elec":0.0,***
***"gas":0.0,***
***"time":200301},***
***…… *****
***{"water":0.0,***
***"index":167,***
***"elec":0.0,***
***"gas":0.0,***
***"Time":201612}***
***],***
***"sqft":351391.0,***
***"bldgId":"AK0013ZZ",***

***"bldgIndex":5,***

***"state": "AK",***

***"stateIndex": 0;***

***}***

In the JSON file, "bldgData" contains all data entries of water consumption, gas consumption, electricity consumption, index of the entry sorted by time, as well as entry time. There are 158 entries in each segment of "bldgData", corresponding to each fiscal month from January 2003 to July 2016; "Sqft" is the gross floor area of the building; "bldgId" is the unique building ID of the building; "bldgIndex": is the index of the building among all profiles; "state" is the two-character abbreviation of the corresponding state; and "stateIndex": is the index of the state out of 50 state abbreviations (for future reference).

### 4.2.2 Topographic Viewport of the US

#### 4.2.2.1 Importing and Aligning Geographical Map

In the visualization I used a set of open-sourced stl models of the geographical map of each of the states in the US (NIorang, 2016). It is easy to import and fast to load. Also the topographical details can add to the depth of the dimensions of the visualization.

Since the source files of the map has 50 parts, corresponding to 50 States. So the coordinates of each state need to be rearranged into

the correct relative location. In order to get to the most accurate position possible, by using the dragging function of mouse in three.js to freely manipulate all imported models. At the same time recording the eventual coordinates of each model after each mouse-dragging event update. Then extract the arrays of the x, y coordinate of the final position and import the coordinates as constants into the scene to create the correct position. A button in the dashboard is created to create two csv files containing temporarily saved coordinates. User can also click the button to extract the information of their own arrangement of the map and import the coordinates later. The process of aligning the 50 states is shown in the following figures.

Figure 16: The process of importing and aligning the topographical map of US from 50 separate models [1]

Figure 17: The process of importing and aligning the topographical map of US from 50 separate models [2]


Figure 18: The process of importing and aligning the topographical map of US from 50 seperate models

Also by importing the temperature data and mapping as color gradient for each state, users can explore the relationship with energy consumption and temperature on the state scale. And by designing this feature as a tuggle, it's easy to hide this feature and stop it from distractive when focusing on other aspects.

### 4.2.3.2 Displaying Building Data onto the Map

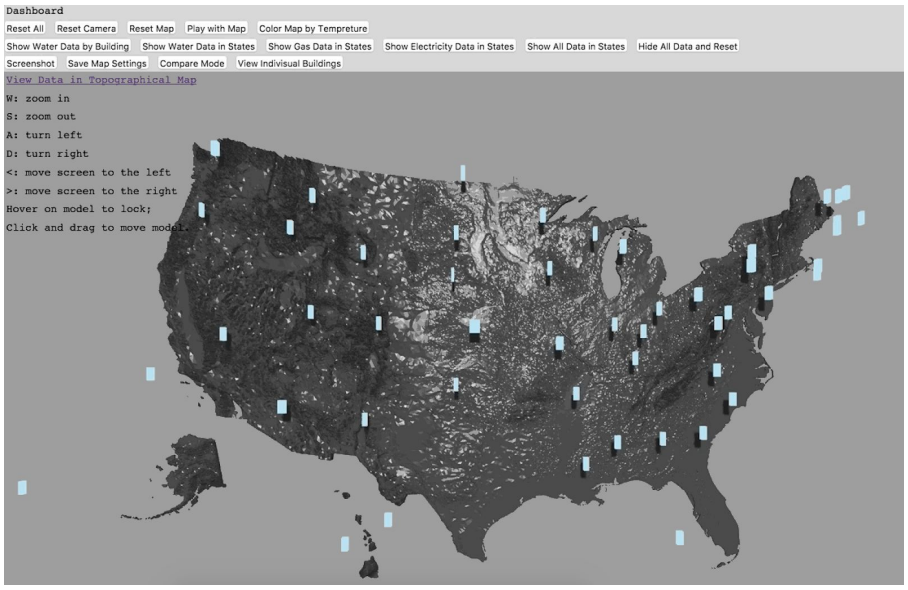


Figure 19: Showing water state data on the topographical map viewport

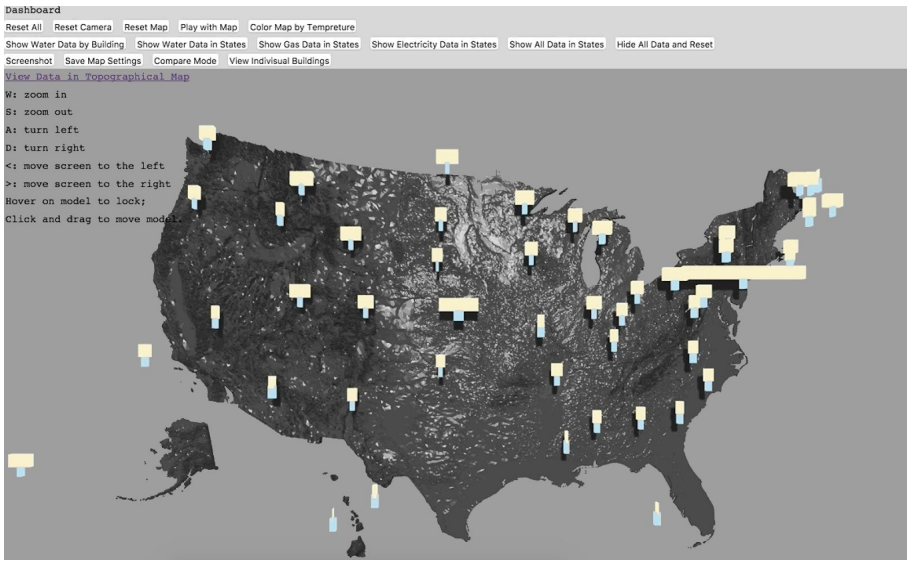


Figure 20: Showing water and gas state data on the topographical map viewport

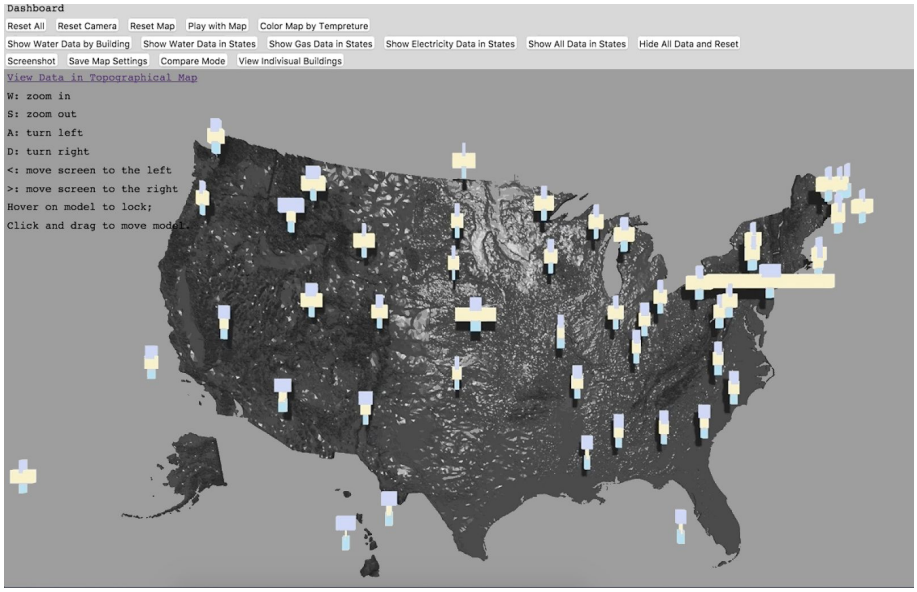


Figure 21: Showing water, gas, and electricity state data on the topographical map viewport

When the user is in the topographic viewport, it is important to present the relationship between the visualized building data and the relative location on the map. However the limitation of space on the map makes it difficult to differentiate each building. By displaying the general information of the buildings in each state, as well as variables that is useful to display in the topographical viewport, the whole picture of the distribution and tendencies of building energy consumption across the country are going to be clearer.

As analysed in the case study about Unusual Visual Expressions and Data Personalization in chapter 3.4, using the unusual visual language of describing each building can not only create the comparison feature when presented with other buildings while mapped according to state and location, it can also invite the user to explore the most important piece of information about the buildings, forming a memory and a focal point to move forward onto other viewports to

discover new aspects of the visualized data. (Because there's no specific location, the relative location are estimated.)

### 4.2.3 Detailed and Integrated Viewport of State Data

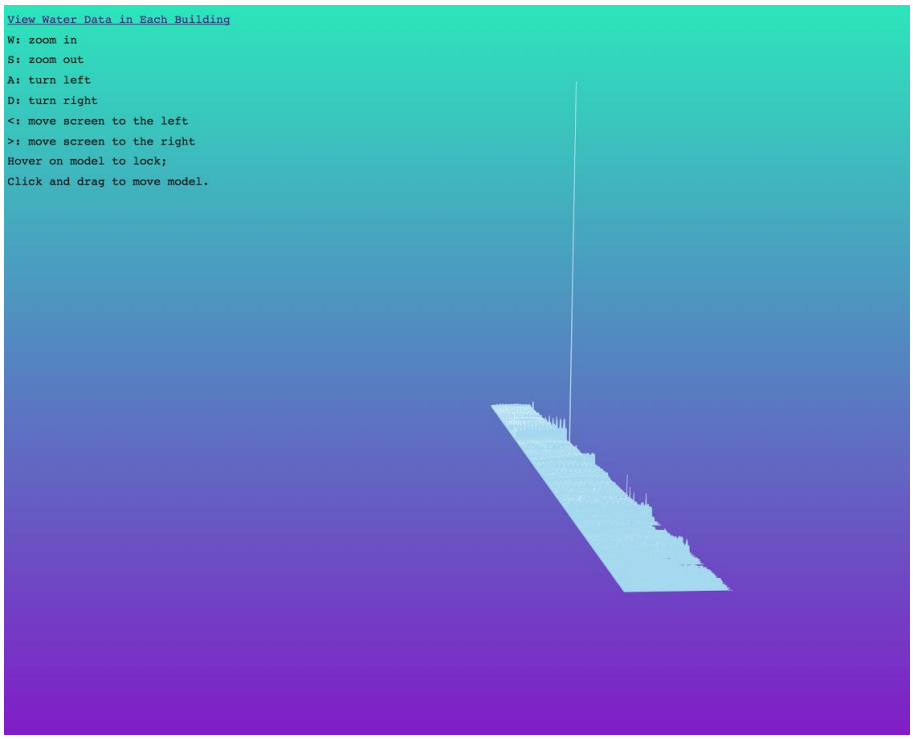


Figure 22: First Impression of the detailed viewport

Combining the “Free Edit” Mode that can be invoked in all viewports with the zooming feature, as well as the feature of temperature heatmap, users can choose multiple states by selecting the area of the topographical map in the first viewport. After locking down the selected states, more detailed and organized information that is integrated from all the building data in the selected states, are displayed in a paralleled manner, allowing users to compare data in

the scale of state from different aspects. And users can arrange the information in the manner they prefer, and save the satisfied view when ready.

### 4.2.4 Comparison Viewport of Building Data

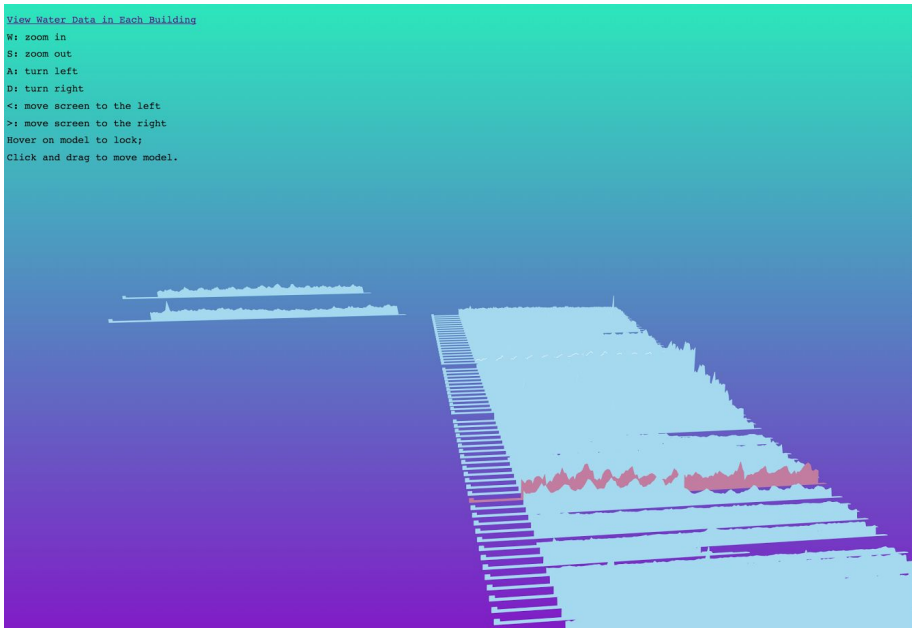


Figure 23: User can select buildings and arrange them freely in the comparison viewport

From the viewport of the state data, users can select individual building data from the selected state. At the same time, there is a paralleled coordinates of all building data in the entire visualization. In order to have a more direct link among all the small multiples, when the mouse is hovered above a point on any line chart, the point of same time stamp will be highlighted.

## 4.3 Designing Data Interaction and Experience

### 4.3.1 “Free Edit” Mode

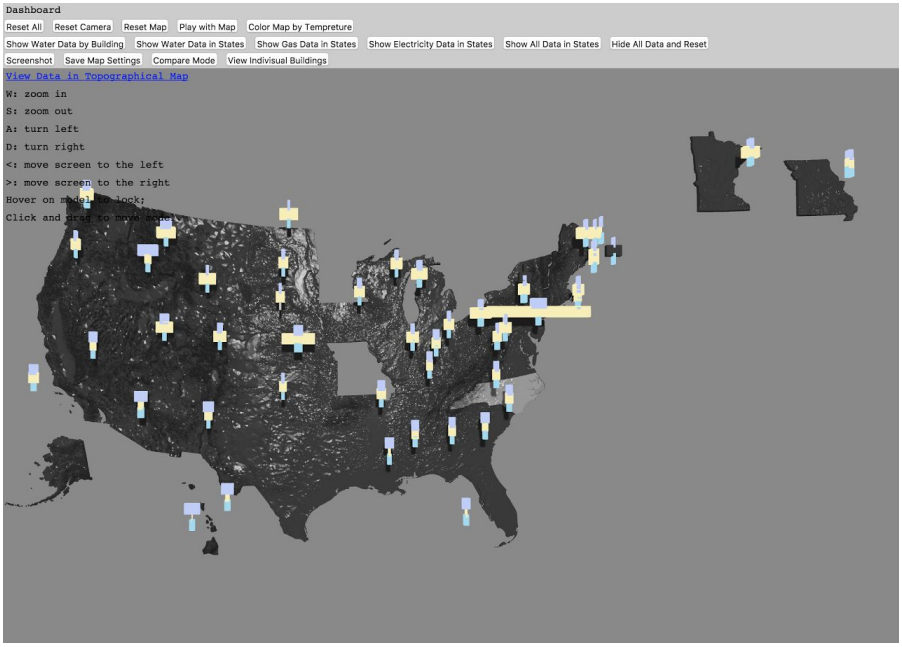


Figure 24: “Free-edit mode” allows users to manipulate the objects and visualizations freely in the first viewport

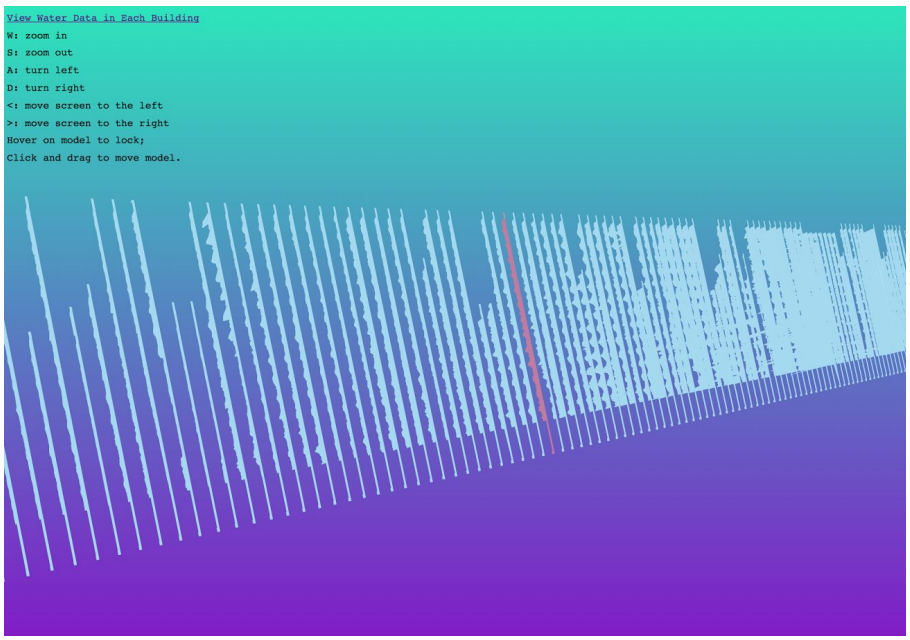


Figure 25: “Free-edit mode” allows users to manipulate the objects and visualizations freely in the first viewport

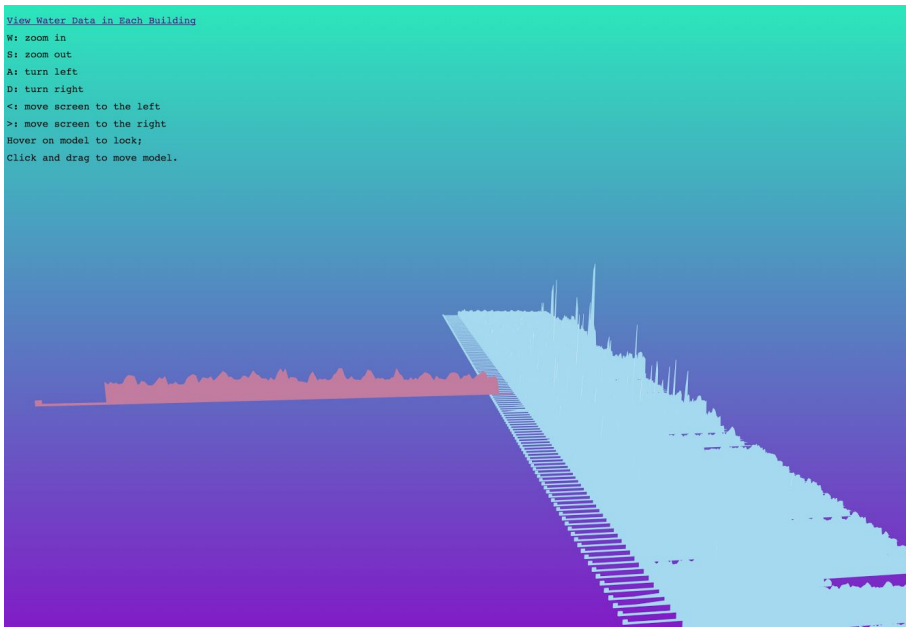


Figure 26: “Free-edit mode” allows users to manipulate the objects and visualizations freely in the second viewport

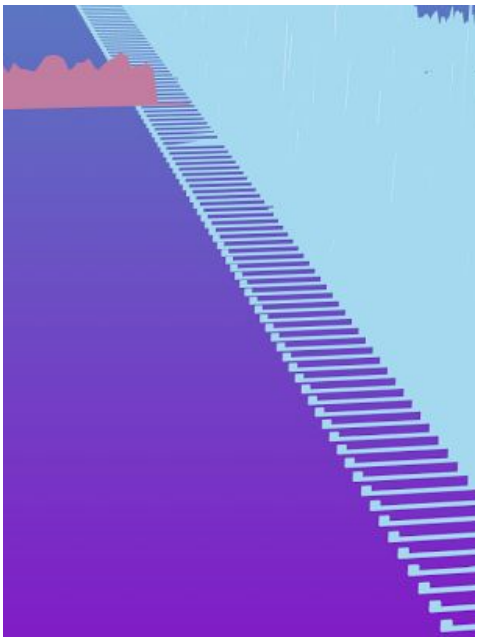

Figure 27: Using the metaphors of “Popsicle sticks” or file handles encourage users to engage in free editing

One of the main features of the prototype is the free edit mode implemented in both the topographic viewport and the detailed and integrated viewport. Not only can users move around the entire scene in the viewport to adjust angles, positions and distance of the graphs, they can also select certain graphs, which are highlighted when being selected, drag and drop them to any position.

In the detailed viewport, in order to encourage users to engage in free editing, the graphs of the building data are sorted and aligned into a staircase-like formation, while adding the metaphors of "Popsicle sticks" or file handles on the left of each graph.

Similar to the detail viewport, in the topographic viewport, users can hover to highlight, click and drag to move the model of each state to anywhere in the blank canvas. When the user moves the state model, an average value of the water, electricity and gas data of the selected states move with the model, when user update the display of the data by pressing the button on the second row of the dashboard.

The goal of this feature is to encourage users to take initiation when having an interactive experience with data. Instead of accepting the well organized data narrative, users can follow their own instincts and find information provided in the existing narrative, and apply them in the context that is more intuitive and relevant to themselves.

With the export functionality shown in the dashboard, after users finish creating their own narrative on the blank scene, which can be seen as the "canvas", they can save the entire viewport into a jpeg file, and share their own story that is built with the data material from

the existing narrative, to others, who might also have their own version of the story. The effect can still exist and be shared after the initial interaction of the user with the visualization itself, therefore potentially accumulating the interest and improving the experience of the data story, augmenting the knowledge and insight of energy-related issues, and potentially trigger ecological responsible behaviors in the long term.

### 4.3.2 Keyboard Interactions and Toggle Dashboard

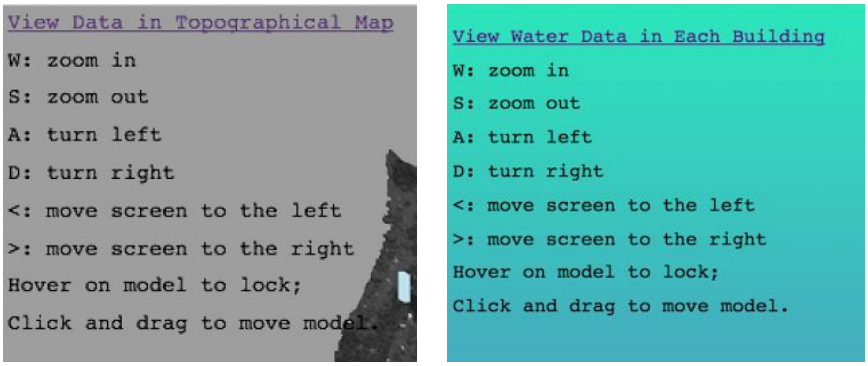


Figure 28 and 29: the instructions for keyboard interactions on the up-left corner of both viewports

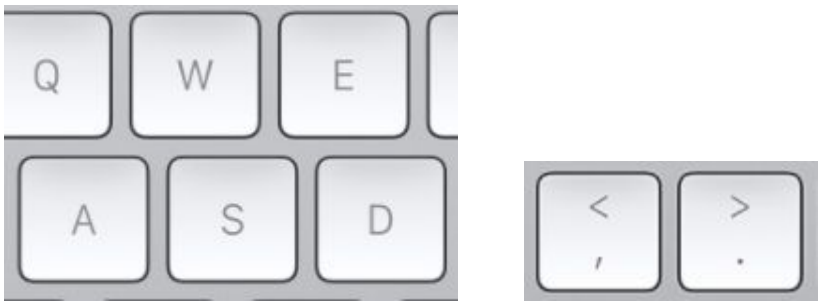


Figure 30: the keyboard interaction of the left hand(WASD), AND RIGHT HAND (< and >)

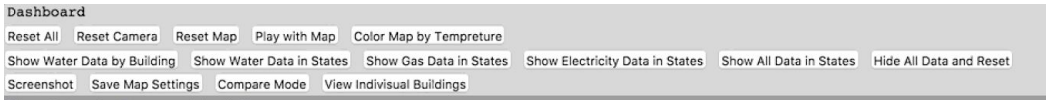


Figure 31: the toggle dashboard in the topographic viewport

The first row of the dashboard allows users to reset the entire scene, reset the camera to set the model in the center of the scene, reset map to return all state models into the original positions, enter "free-edit" mode, and color the map by temperature.

The second row of the dashboard allows users to show different data, including the average value of water, gas and electricity of each state, reflecting a general but relatively comparable representation of the data in each state.

The third row of the dashboard allows users to save and export coordinates of the state models as well as screenshots of the entire scene, and enter the detailed viewport after they are ready to explore the dataset in a more comprehensive level.

### 4.3.3 Transitions Between ViewPorts

This interactive function is meant to connect users perceptions of different depth and angles from multiple viewports. By transitioning of both viewports through arranged layers and according to the level of detail of the data presentation, users can gradually form a vivid, comprehensive structure of the entire dataset.

# 5 Conclusions and Next Steps

## 5.1 Conclusions

One major goal of the prototype is to explore possibilities for a more proactive interaction environment, through which users can gain a comprehensive understanding of a relatively complex concept. Therefore using different kinds of visual focus to express the layer's most sensible information can be useful. The interfaces are designed to be more flexible comparing to the conventional forms of data visualization by allowing users to freely manipulate the objects that are part of the visualization, and gives users endless possibilities of observation, comparison and storytelling according to their preference and attention flow.

Free Edit mode turns a passive receiving experience into a liberated one, allowing users to create their own version of the visualization using the basic modules and components created with target data, turning the viewing experience into a playing one. With “Free Edit” Mode, users can create their own narrative, arranging data elements that are relevant and interesting to them on a personal level, so that they can engage with the narrative even more than simply interacting with animation and pre-arranged way of data presentation.

Meanwhile, without using digital tools specialized for data visualization and plotting, the visual language and expressions can be freed that it gives designers more opportunities to think out of the box, developing personal connections between users and data, creating an environment that allow users to fully experience the possibilities of their explorations based on the simple rules set by the designer.

### 5.2 Next Steps

Although the prototype allows users to freely manipulate modules and components in different viewports, it’s still limited in a web browser at the moment. A future step of taking the interaction further is to explore the possibilities of breaking through the limitations of interaction in a computer screen, but using more intuitive and immersive ways, such as gesture and sound, to interact with data visualization. Furthermore, the effectiveness of these design methods used in the prototype and the efficacy of visualizing

energy data needs to be compared and investigated in the future research as well.

At the same time, the question that whether an interactive data visualization such as what the prototype demonstrates can help people better perceive energy data and be more ecological responsible. Therefore a user testing process is very much needed in order to get feedbacks from real users, provide guidelines to further discussions of the efficacy and effectiveness of the method, and further improve the experience and getting closer to the goal of triggering users' ecological responsible behaviors. There are some technical issues with the current prototype that needed to be solved and some difficulties and limitations when it comes to the interaction activities such as the struggle of aligning data more accurately in the 3d perspective.

### 5.3 Contributions

Investigated the fundamentals of data visualization in the context of energy data through case studies for current energy data visualization projects across different scales;

Designed an interactive experience for building energy data with the focus of time-based water use and consumption;

Developed an approach towards the application of data visualization based on visualization and interaction techniques and methods to elevate the experience of human-data interaction.

## Bibliography


Anonymous (n.d.). Datavisualization.ch. http://selection.datavisualization.ch/.

Apple (2017). Magic Keyboard with Numeric Keypad. https://www.apple.com/shop/product/MQ052LL/A/magic-keyboard-with-numeric-keypad-us-english.

Azavea Inc. (2017). Philadelphia 2017 Building Energy Benchmarking. http://visualization.phillybuildingbenchmarking.com/#!/.

Bornstein, D. (2015). Investing in Energy Efficiency Pays Off. https://opinionator.blogs.nytimes.com/2015/02/06/investing-in-energy-efficiency-pays-off/.

Canadian Energy Systems Analysis Research (2013). Sankey diagrams associated with fuel and electricity production and use in Canada. http://www.cesarnet.ca/visualization/sankey-diagrams-canadas-energy-systems.

Detroit Edison Company (1935). 3D Visualizations of Power Consumption. Willard Cope Brinton (1939) Graphic Presentation pp 364-365. Available at: http://dataphys.org/list/electricity-power-demand/.

Dropcountr (n.d.). Dropcountr. https://dropcountr.wpengine.com/.

Fry, B. (2008). Visualizing Data: Exploring and Explaining Data with the Processing Environment. 1st Edition. O'Reilly Media, Inc. 384 p.

Fulton, M., Baker, J., Brandenburg, M., Herbst, R., Cleveland, J., Rogers, J. and Onyeagoro, C. (2012). United States Building Energy Efficiency Retrofits Market Sizing and Financing Models. Available at: http://web.mit.edu/cron/project/EESP-Cambridge/Articles/Finance/Rockefeller%20and%20DB%20-%20March%202012%20-%20Energy%20Efficiency%20Market%20Size%20and%20Finance%20Models.pdf.

Gasbuddy (2017). USA National Gas Price Heat Map. http://www.gasbuddy.com/GasPriceMap?z=4.

Google, (2017a). Google Map APIs Documentation: Controls. https://developers.google.com/maps/documentation/javascript/controls.

Google, (2017b). Google Map APIs Documentation: interaction. https://developers.google.com/maps/documentation/javascript/interaction.

General Services Administration (2015).Energy Usage Analysis System (EUAS). Available at: https://catalog.data.gov/dataset/energy-usage-analysis-system.

Holmes, T. (2006-2009). 7000 oaks and counting. https://tholme.myportfolio.com/current-ecoart/7000-oaks-and-counting/.

Holmes, T. (2007). Eco-visualization: Combining Art and Technology to Reduce Energy Consumption. In Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition (C&C '07). ACM, New York, NY, USA, 153-162. DOI: https://doi.org/10.1145/1254960.1254982.

Huertas, A. and Adler, D. (2012), Is News Corp. Failing Science? Available at: https://www.ucsusa.org/sites/default/files/legacy/assets/documents/global_warming/Is-News-Corp-Failing-Science.pdf.

International Energy Agency (2017). International energy technology perspectives: ETP 2017 Data Visualization. Emissions Reductions and Energy Flows. http://www.iea.org/etp/explore/.

Jansen, Y. (2011). Can We Keep Up: Sponges Show Domestic Water Usage. http://dataphys.org/list/domestic-water-usage-visualized-with-sponges/.

Jansen, Y., Dragicevic, P. and Fekete, J. (2013). Evaluating the efficiency of physical visualizations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). ACM, New York, NY, USA, 2593-2602. DOI: https://doi.org/10.1145/2470654.2481359.

Kekeritz, T. (2007). The virtual water project. http://virtualwater.eu/.

Khan, M., & Khan, S. S. (2011). Data and information visualization methods, and interactive mechanisms: A survey. International Journal of Computer Applications, 34(1), 1-14.

Klemm, M. (2017). CO2 Emissions Shown with Balloons. http://dataphys.org/list/co2-emissions-shown-with-balloons/.

Kontokosta, C. E., Kontokosta, C., Marulli, D., Tull, C., & Pingerra, R. (2015). Web-Based Visualization and Prediction of Urban Energy Use from Building Benchmarking Data Web-Based Visualization and Prediction of Urban Energy Use from Building Benchmarking Data, (October).

Lindeman, T., Mellnik, T. and Englund, W. (2015). As appetite for electricity soars, the world keeps turning to coal. https://www.washingtonpost.com/apps/g/page/world/as-appetite-for-electricity-soars-the-world-keeps-turning-to-coal/1842/.

Miller, T. and Pollak, T. (2013). Environmental Coverage in the Mainstream News: We Need More. Available at: https://climateaccess.org/system/files/PIEC_Environmental%20Coverage.pdf.

Mortier, R., Haddadi, H., Henderson, T., Mcauley, D., Crowcroft, J., and Crabtree, A. (n.d.). “41. Human-Data Interaction”. The Encyclopedia of Human-Computer Interaction, 2nd Ed. Available at: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-data-interaction.

Muyskens, J., Keating, D. and Granados, S. (2017). Mapping how the United States generates its electricity. https://www.washingtonpost.com/graphics/national/power-plants/?utm_term=.bb48f65020d4.

Nlorang (2016). 3D Topographic Maps of US States. https://www.thingiverse.com/thing:1524543.

Olsen, E. (2017). A Sense for Energy - Representing Energy in Buildings. (Ed. Willis, D., Braham, W. W., Muramoto, K. and Barber, D. A.). Energy Accounts: Architectural Representations of Energy, Climate, and the Future. Routledge, 2017.

Posavec, S. and Lupi, G. (2016a). Dear Data. 1st ed. Princeton Architectural Press. 288p.

Posavec, S. and Lupi, G. (2016b). Dear Data: The Project. http://www.dear-data.com/theproject.

Saxum (n.d.). The United States of Energy. http://archive.usofenergy.com/.

Saxum (2017). The United States of Energy 2.0 (Saxum, 2017) http://usofenergy.com/overview/.

Sherpa, S., Howard, B., Parshall,L., Thompson,J., Hammer, S., Dickinson, J. and Modi, V. (n.d.) Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC. http://qsel.columbia.edu/nycenergy/

Vicente, V. S. (n.d.). Electric Generation in Spain - Latest 24 hours. http://energia.ningunaparte.net/en/.

Loftness, V., Aziz, A., Lasternas, B., and Peters, S. (2017). Ducks, Dollars, or kWh? (Ed. Willis, D., Braham, W. W., Muramoto, K. and Barber, D. A.). Energy Accounts: Architectural Representations of Energy, Climate, and the Future. Routledge, 2017.

The Pac (2005). The Power-aware Cord. http://www.poweraware.com/en/.

ThingsBoard authors (n.d.).thingsboard.io. https://thingsboard.io/smart-energy/.

Tobin, M. (2013). The United States of Energy. http://ecowest.org/2013/07/16/the-united-states-of-energy/

Wilson, M (2002). Six Views of Embodied Cognition. Psychonomic Bulletin & Review 9, 625–636.

World Bank Group (n.d.). CO2 emissions (metric tons per capita). https://data.worldbank.org/indicator/EN.ATM.CO2E.PC.

Yanko Design (2008). SAVERCLIP. http://www.yankodesign.com/2008/01/14/saver-clip-shames-your-electricity-usage/.

Zoss, A. (2017). Data Visualization: Visualization Types. https://guides.library.duke.edu/datavis/vis_types.

# Appendix A: Evaluating the Efficacy of Existing Energy Data Visualizations

| *YEAR* | *Name* | *format* | *Scale of Energy Data* | *Categories of Energy Sources* | *Charcateristics of Energy Data* | *Visualization Techniques* | *Human-Data Interaction* | *Visual Expressions* | *Narrative* |
|---|---|---|---|---|---|---|---|---|---|
| 1935 | 3D Visualizations of Power Consumption (Detroit Edison Company, 1935) | physical data visualisation | community | electricity | Time-based<br>Single type of energy | Area Chart<br>Time Line<br>data physicalization | n/a | Material | User Driven<br>Comparison |
| 2005 | The Power-aware Cord (The Pac, 2005) | product design, sensor | household individual | electricity | Real-time<br>Single type of energy | Dashboard | Real-time Update | color coding | User Driven<br>Comparison |
| 2006 - 2009 | 7000 oaks and counting (Holmes, 2006-2009) | web application | community | Carbon footprint | Geolocation-based<br>multiple types of energy<br>Time-based | Dashboard | Real-time Update | metaphore<br>multi-media<br>interaction<br>real-time data visualization | User Driven<br>Comparison |
| 2007 | The virtual water project (Kekeritz, 2007) | Info graphics in print | | water | Single type of energy | Small Muptiples | n/a | Icons | User Driven<br>Comparison |
| 2008 | SAVERCLIP (Yanko Design, 2008) | product design, sensor | household individual | electricity | Real-time<br>Single type of energy | Dashboard | Real-time Update | Animation<br>Color Coding | User Driven<br>Comparison |
| 2010 | Greenhouse Gas Emission Per Capita (World Bank Group, n.d.) | web application | Continent | greenhouse gas emission | Geolocation-based<br>multiple types of energy<br>Time-based | Time line<br>Bubble Chart<br>Line Chart<br>Semantic Network | Cursor-based<br>Interaction (hover, click)<br>Dashboard for filtering<br>Overview+Detail | Icons<br>Color Coding<br>Color Gradient | User Driven<br>Comparison |
| 2011 | Can We Keep Up: Sponges Show Domestic Water Usage (Jansen, 2011) | physical data visualisation | worldwide | water | Single type of energy<br>Geolocation-based | Geographic Map<br>Bar Chart<br>data physicalization | n/a | Metaphor<br>Color Coding<br>Material | User Driven<br>Comparison |
| 2013 | Sankey diagrams associated with fuel and electricity production and use in Canada (Canadian Energy Systems Analysis Research, 2013) | web application | country | fuel<br>electricity<br>production and use | Geolocation-based<br>multiple types of energy<br>Time-based | Data Flow Diagrams | Cursor-based<br>Interaction (hover, click)<br>Scroll down<br>Dashboard for filtering<br>Overview+Detail | Color Coding | User Driven<br>Comparison |
| 2015 | As appetite for electricity soars, the world keeps turning to coal (Lindeman, Mellnik and Englund, 2015) | web application | worldwide | coal<br>electricity<br>other(multiple) | Geolocation-based<br>multiple types of energy<br>Time-based | Bubble Chart<br>Pie Chart<br>Geographic Map<br>Multi Series Area Chart | Scroll down | Icons<br>Color Coding | Author Driven<br>Information Slideshow<br>literary narration<br>comparison |
| 2017 | Mapping how the United States generates its electricity (Muyskens, Keating and Granados, 2017) | web application | country | multiple | Geolocation-based<br>multiple types of energy<br>Time-based | Bubble Chart<br>Geographic Map<br>Bar Chart<br>Line Chart | Cursor-based<br>Interaction (hover, click)<br>Scroll down | Small Multiples<br>Icons<br>Color Coding | Author Driven<br>Classification<br>Comparison<br>Literary narrition |

| *YEAR* | *Name* | *format* | *Scale of Energy Data* | *Categories of Energy Sources* | *Charcateristics of Energy Data* | *Visualization Techniques* | *Human-Data Interaction* | *Visual Expressions* | *Narrative* |
|---|---|---|---|---|---|---|---|---|---|
| 2017 | International energy technology perspectives: ETP 2017 Data Visualization. Emissions Reductions and Energy Flows (International Energy Agency, 2017) | web application | country | Emission reduction energy flow | Geolocation-based<br>multiple types of energy<br>Time-based | Data Flow Diagrams<br>Time Line<br>Line Chart<br>Area Chart | Cursor-based<br>Interaction (hover, click)<br>Scroll down<br>Dashboard for filtering<br>Overview+Detail | Animation<br>Color Coding | User Driven<br>Comparison |
| 2017 | CO2 Emissions Shown with Balloons (Klemm, 2017) | physical data visualisation | worldwide | CO2 emission | Single type of energy<br>Geolocation-based | Area Chart<br>Geographic Map<br>data physicalization | n/a | Metaphor<br>Material | User Driven<br>Comparison |
| n/a | Electric Generation in Spain - Latest 24 hours (Vicente, V. S., n.d.) | web application | country | multiple | Geolocation-based<br>multiple types of energy<br>Time-based | Time Line<br>Area Chart | Cursor-based<br>Interaction (hover, click)<br>Scroll down<br>Real-time Update | Icons<br>Color Coding | User Driven<br>classification<br>comparison |
| n/a | The United States of Energy (Saxum, n.d.) | web application | country | multiple | Geolocation-based<br>multiple types of energy<br>Time-based | Bubble Chart<br>Geographic Map<br>Bar Chart<br>Line Chart<br>Small Multiples<br>Pie Chart<br>Area Chart | Cursor-based<br>Interaction (hover, click)<br>Scroll down<br>Zoom in, zoom out | Animation<br>Color Coding | User Driven<br>Comparison<br>Literary narrition |
| n/a | Estimated Total Annual Building Energy Consumption at the Block and Lot Level for NYC (Sherpa, S. Et, al., n.d.) | web application | city | multiple | Geolocation-based<br>multiple types of energy<br>Time-based | Pie Chart<br>Geographic Map<br>Heatmap<br>Bar Chart | Cursor-based<br>Interaction (click, drag)<br>Scroll down<br>Zoom in, zoom out<br>Dashboard for Detail Display | Color Coding | User Driven<br>Comparison<br>Estimation |
| n/a | Dropcountr (Dropcountr, n.d.) | mobile application<br>desktop application<br>web application | community | multiple | Geolocation-based<br>multiple types of energy<br>Real-time | Bubble Chart<br>Geographic Map<br>Bar Chart<br>Dashboard | Touch-based<br>Interaction (click, drag)<br>Scroll down<br>Zoom in, zoom out<br>Overview+Detail | Color Coding | User Driven<br>Comparison |
| n/a | thingsboard.io (ThingsBoard authors, n.d.) | web application<br>LoT | household individual | multiple | Geolocation-based<br>multiple types of energy<br>Real-time | Dashboard<br>Geographic Map<br>Bar Chart<br>Pie Chart<br>Table<br>Line Chart<br>Pie Chart | Touch-based<br>Interaction (hover, click)<br>Overview+Detail<br>Real-time Update | real-time data visualization dashboard | User Driven<br>Comparison |

## Appendix B: Source Code

```
Part I: Data Processing


"""
STEP 1:
Combines all data entries from the same year
   into a single csv file for future data
processing.
"""

import os, xlrd, csv
from time import gmtime, strftime

# converts xlsx file into csv file and rename
def excel_to_tsv(sheet_path, sheet_name,
path_tsv):
    workbook = xlrd.open_workbook(sheet_path +
sheet_name)
    # sh = wb.sheet_by_name(sheet_name)
    sheet = workbook.sheet_by_index(0)

    tsv_file = open(path_tsv + sheet_name +
'.tsv', 'wb')
    writer = csv.writer(tsv_file,
quoting=csv.QUOTE_ALL)

    for rownum in xrange(sheet.nrows):
        writer.writerow(sheet.row_values(rownum))

    tsv_file.close()
```

```
"""
adapted from
https://stackoverflow.com/questions/20105118/
convert-xlsx-to-csv-correctly-using-python
"""

# combines csv files containing data from teh same
year into one file and rename
def combineTsvs(path, end_path, from_year,
to_year):
   print "Combining..."
   os.chdir(path)       #change directory to the
csv files' folder
   tsv_name = str(from_year)+"-"+str(to_year)+"
"+strftime("%m-%d %H:%M:%S", gmtime())
   output_file = open(end_path + tsv_name +
'.tsv', 'a')

   for fn in os.listdir('.'):
        if (os.path.isfile(fn) & (fn !=
".DS_Store")
             & ((("EUAS " + str(from_year)) in fn)
or (("EUAS " + str(to_year))) in fn)):
             print (fn)
             for line in open(fn):
                  output_file.write(line)
   output_file.close()

#puts all csv files of certain years into the same
folder
def integrate(from_year, to_year, in_path,
out_path):
   print "integrating..."
   for year in xrange(from_year, to_year+1):
        tail_path = "EUAS " + str(year) + "/"
# "EUAS 2003/"
        path = in_path + tail_path
        os.chdir(path)       # change directory to
certain year
        for fn in os.listdir('.'):
             if os.path.isfile(fn) & (fn !=
".DS_Store"):
                  print (fn)
                  excel_to_tsv(path, fn,
out_path)
```

```
in_path =
"/Users/darcy/Desktop/THESIS/dataProcessing/EUAS/"
out_path =
"/Users/darcy/Desktop/THESIS/dataProcessing/EUASTs
v/"
final_path = in_path + "YearlyData/"

# testing
# path_sheet_data_folder =
"/Users/darcy/Desktop/WaterViz/EUAS/EUAS 2003/"
# path_project = "/Users/darcy/Desktop/WaterViz/"
# test_sheet_name = "Region 1 EUAS 2003"

# excel_to_csv(path_sheet_data_folder,
test_sheet_name, path_csv_data_folder)


# shortcut of processing multiple years
def integrateDatabase(from_year, to_year, in_path,
out_path, final_path):
   integrate(from_year, to_year, in_path,
out_path)
   combineTsvs(out_path, final_path, from_year,
to_year)


# integrateDatabase(2003, 2003, in_path, out_path,
final_path)


def main():
   #integrate(2003, 2016, in_path, out_path)
   for year in xrange(2003, 2016+1):
        print "Combining data from year " +
str(year)
        combineTsvs(out_path, final_path, year,
year)


main()



# def path_leaf(path):
#     head, tail = ntpath.split(path)
#     return tail or ntpath.basename(head)
# https://stackoverflow.com/questions/8384737/
```

```
#
extract-file-name-from-path-no-matter-what-the-os-
path-format
```

---

```
'''
STEP 2
Creates data for a single building (or a certain
building)
   from the yearly data csv files got from step 1.
Rename file based on building id, State located
and other bldg properties.
Discards redundant columns leaving only the
consumption and cost of
   gas, electricity and water for future
referance.


Data input columns:
   "Region No.",
   "State",[1]
   "Service Center",
   "Fiscal Month",[3]
   "Fiscal Year",[4]
   "Building Number",[5]
   "Area Field Office",[6]
   "Cat",[7]
   "Building Designation",[8]
   "Gross Sq.Ft",[9]
   "Electricity (KWH)",[10]
   "Electricity (Cost)",[11]
   "Demand (KW)",
   "Demand (Cost)",
   "Steam (Thou. lbs)",
   "Steam (Cost)",
   "Gas (Cubic Ft)",[16]
   "Gas (Cost)",[17]
   "Oil (Gallon)",
   "Oil (Cost)",
   "Chilled Water (Ton Hr)",
   "Chilled Water (Cost)",
   "Renewable Electricity (KWH)",
   "Renewable Electricity (Cost)",
   "Renewable Gas (Cubic Ft)",
   "Renewable Gas (Cost)",
```

```
   "Other (mmBTU)",
   "Other (Cost)",
   "Water (Gallon)",[28]
   "Water (Cost)"[29]



Data output columns:
   in titie:
        ("State",)
        "Building Number",
        "Gross Sq.Ft",
   in file:
        "Fiscal Month",
        "Fiscal Year",
        "Electricity (KWH)",
        "Electricity (Cost)",
        "Gas (Cubic Ft)",
        "Gas (Cost)",
        "Water (Gallon)",
        "Water (Cost)"

'''

import os, csv

# creates empty files of unique bldg id in new
folder
def createNewBldgTsvs(set_of_ids):
   os.chdir('./BldgTsvs2/')
   # os.chdir('./BldgTsvs/')
   temp_file_names = set_of_ids.copy()
   while len(temp_file_names) != 0:
        temp_file_name = temp_file_names.pop()
        # found new building. Create new file
        if temp_file_name not in os.listdir('.'):
             tsv_file_temp =
open(temp_file_name.strip('\"') + ".tsv",
'a').close()
   return


#   "State",[1]
#  "Fiscal Month",[3]
#  "Fiscal Year",[4]
#  "Building Number",[5]
#  "Gross Sq.Ft",[9]
#  "Electricity (KWH)",[10]
```

```
#  "Electricity (Cost)",[11]
#  "Gas (Cubic Ft)",[16]
#  "Gas (Cost)",[17]
#  "Water (Gallon)",[28]
#  "Water (Cost)"[29]


def assembleBldgTsv(entry_dict):
   print "Getting Bldg data..."
   #for i in xrange(10):

   for i in xrange(1, len(entry_dict)):
        if
(checkDataIsComplete(entry_dict[i][28])):
             temp_month = entry_dict[i][3]
             temp_year = entry_dict[i][4]
             temp_id =
entry_dict[i][5].strip('\"')
             temp_location =
entry_dict[i][5].strip('\"')
             temp_category =
entry_dict[i][5].strip('\"')
             temp_Sqft = entry_dict[i][9]
             temp_ElecKWH = entry_dict[i][10]
             temp_ElecCost = entry_dict[i][11]
             temp_GasCubic = entry_dict[i][16]
             temp_GasCost = entry_dict[i][17]
             temp_WaterGallon = entry_dict[i][28]
             temp_WaterCost = entry_dict[i][29]
             #print type(temp_id), temp_id
             #print temp_id + ".tsv"
             if (temp_id+".tsv") in
os.listdir('.'):
                  tsv_file = open(temp_id +
".tsv", 'a')
                  writer = csv.writer(tsv_file,
delimiter='\t', lineterminator='\n')
                  temp_row =
[temp_month.strip('\"'), temp_year.strip('\"'),
temp_Sqft.strip('\"'),
                       temp_ElecKWH.strip('\"'),
temp_ElecCost.strip('\"'),
temp_GasCubic.strip('\"'),
                       temp_GasCost.strip('\"'),
temp_WaterGallon.strip('\"'),
temp_WaterCost.strip('\"')]
                  writer.writerow(temp_row)
                  #print temp_row
```

```
                    #entry_dict.pop(i) ??????why
after delete this line it appends??

                    tsv_file.close()
        else:
              temp_month = entry_dict[i][3]
              temp_year = entry_dict[i][4]
              temp_id =
entry_dict[i][5].strip('\"')
              temp_location =
entry_dict[i][5].strip('\"')
              temp_category =
entry_dict[i][5].strip('\"')
              temp_Sqft = entry_dict[i][9]
              temp_ElecKWH = entry_dict[i][10]
              temp_ElecCost = entry_dict[i][11]
              temp_GasCubic = entry_dict[i][16]
              temp_GasCost = entry_dict[i][17]
              temp_WaterGallon = entry_dict[i][28]
              temp_WaterCost = entry_dict[i][29]
              #print type(temp_id), temp_id
              #print temp_id + ".tsv"
              if (temp_id+".tsv") in
os.listdir('.'):
                    tsv_file = open(temp_id +
".tsv", 'a')
                    writer = csv.writer(tsv_file,
delimiter='\t', lineterminator='\n')
                    temp_row =
[temp_month.strip('\"'), temp_year.strip('\"'),
temp_Sqft.strip('\"'),
                          temp_ElecKWH.strip('\"'),
temp_ElecCost.strip('\"'),
temp_GasCubic.strip('\"'),
                          temp_GasCost.strip('\"'),
temp_WaterGallon.strip('\"'),
temp_WaterCost.strip('\"')]
                    writer.writerow(temp_row)
                    #print temp_row
                    #entry_dict.pop(i) ??????why
after delete this line it appends??

                    tsv_file.close()
   # search for bldg in all yearly files
   # rename and save

   return
```

```
def yearlyDataToBldgData(year, path):
   os.chdir(path)
   #look for yearly data files
   for fn in os.listdir('.'):
        if (os.path.isfile(fn) & (fn !=
".DS_Store")
            & (str(year) in fn)):
            file_name = fn
            break
   print file_name + " found!"

   #open yearly data file and get each column
   with open(path + file_name, 'rb') as csvfile:
       reader = csv.reader(csvfile, delimiter=';',
quotechar='|')
       entry_dict = dict()
       index = 0
       #need to clean out
       for line in reader:
        #print line
        #split with "," instead of , to get rid of
entries like "G, A, X"
        entry_dict[index] = line[0].split('\",\"')
        #print entry_dict[index][8]
        index += 1

   print len(entry_dict)


   set_of_ids = set()
   for i in xrange(len(entry_dict)):
        bldg_id = entry_dict[i][5]
        #print bldg_id
        if (len(bldg_id) == 8 and bldg_id not in
set_of_ids):
            set_of_ids.add(bldg_id)
            #print "new building " + bldg_id + "
added to the set."
   print
"-----------------------------"+str(len(set_of_id
s)) + " bldg IDs found."


   createNewBldgTsvs(set_of_ids)
   print "All new files created."
   assembleBldgTsv(entry_dict)
```

```
   return


path =
"/Users/darcy/Desktop/THESIS/dataProcessing/EUAS/Y
earlyData/"


# calls getBldgCsv() function while keeping track
of all building ids
def main():
   for year in xrange(2003, 2016+1):
        yearlyDataToBldgData(year, path)
        print
"-----------------------------------------year
", year, " is entered into bldg data."

   Return

main()
#yearlyDataToBldg(2013, path)
```

---

```
'''
STEP 3
Preparing building data for visualization
'''

import csv, os, json
import collections


def changeNameOfFiles(path):
   os.chdir(path)
   for fn in os.listdir('.'):
        if (os.path.isfile(fn) & (fn !=
".DS_Store")):
             #next(reader)print fn
             file_name1 = fn
             with open(path +file_name1, 'rb') as
csvfile:
                  reader = csv.reader(csvfile,
delimiter='\t', quotechar='|')
                  file_name2 = next(reader)[2]
                  file_name3 = next(reader)[2]
                  #print file_name3, file_name2
```

```
                    if (file_name2 == file_name3):
#check if it's sqft
                        newName =
fn.split(".tsv")[0] + "_" +
file_name2.split(".0")[0] + ".tsv"
                        print newName
                        os.rename(fn, newName)
   print "changed name format into id + sqft."
   return


"""
This function extracts the relevant data and sort
according to key value.
Output a tsv file with index, Year + month,
gallon/sqft, sorted according to timeline.
"""
def dataPrep(path, name, path2, count):
   hostDict = dict()
   sqft = float(name.split("_")[1].split(".")[0])
   bldgId = name.split("_")[0].split(".")[0]
   hostDict["sqft"] = sqft
   hostDict['bldgId'] = bldgId
   hostDict['bldgIndex'] = count
   print sqft, bldgId
   if (sqft != 0):
        with open(path + name, 'rb') as csvfile:
            reader = csv.reader(csvfile,
delimiter=';', quotechar='|')
            entry_dict = dict()
            index = 0
            for line in reader:
              #print line
              temp = line[0].split('\t')
              key = int(float(temp[0]) +
float(temp[1])*100)
              #print key
              entry_dict[key] =
[round(float((temp[-2]))/sqft, 4),
              round(float((temp[-4]))/sqft, 4),
round(float((temp[-6]))/sqft, 4)]
              #print temp
              #print entry_dict[key]

            #print entry_dict
            od =
collections.OrderedDict(sorted(entry_dict.items())
)
```

```
            #print od

            json_out = open(path2 +
name.split(".tsv")[0] + "_viz" + ".txt", "wb")

            hostDict["bldgData"] = []
            i = 0
            for key in od:
             tempDict = dict()
             tempDict["index"] = i
             tempDict["time"] = key
             tempDict["elec"] = od.get(key)[0]
             tempDict["gas"] = od.get(key)[1]
             tempDict["water"] = od.get(key)[2]
             hostDict["bldgData"].append(tempDict)
             i+=1

            print hostDict["bldgData"]
            with json_out as outfile:
             json.dump(hostDict, outfile)

            # tsv_out = open(path2 +
name.split(".tsv")[0] + "_viz" + ".tsv", "wb")
            # writer = csv.writer(tsv_out,
delimiter='\t', lineterminator='\n')
            # writer.writerow(["index", "day",
"hour", "value"])
            # #temp_row = [key, entry_dict[key]]
            # i = 0
            # for key in od:
            #      #print key%100, key/100
            #      writer.writerow([i, key/100,
key%100, od.get(key)])
            #      i+=1



path =
"/Users/darcy/Desktop/THESIS/dataProcessing/EUAS/Y
earlyData/BldgTsvs/"
#[1]
#changeNameOfFiles(path)

path2 =
"/Users/darcy/Desktop/THESIS/WaterViz/VIZZ/bldg_da
ta_viz2/"
def preps(path, path2):
   os.chdir(path)
```

```
   #look for yearly data files
   count = 0
   name_arr = []
   for fn in os.listdir('.'):
        if (os.path.isfile(fn) & (fn !=
".DS_Store")):
             file_name = fn


             temp = "bldg_data_viz2/"
+file_name.split(".tsv")[0] + "_viz" + ".csv"
             sqft =
file_name.split(".tsv")[0].split('_')[1]
             if (sqft != 0):
                  name_arr.append(temp)
                  print temp
             #dataPrep(path, file_name, path2,
count)
             count += 1



   return name_arr

#dataPrep(path, 'AK0000AA_727340.tsv', path2, 1)

print preps(path, path2)

# path0 = "/Users/darcy/Desktop/WaterViz/"
# name = "ID0025ZZ_274412.csv"

#dataPrep(path, name)
```

---

```
# -*- coding: utf-8 -*-
import csv, os, json
from pprint import pprint
from shutil import copyfile


'''
```

```
Cleaning data and put all building viz data into 4
buckets
according to the rate of completeness of water,
gas and electricity data
3_zeros: there are entries from this building with
all three data absent in certain year
2_zeros: there are entries from this building with
at most two data absent in certain year
1_zero: there are entries from this building with
at most one data absent in certain year
0_zeros: here are no entries from this building
with any data absent
'''

path0 =
"/Users/darcy/Desktop/THESIS/3dWaterViz/bldg_data_
viz2/"
path3 =
"/Users/darcy/Desktop/THESIS/3dWaterViz/3_zeros/"
path2 =
"/Users/darcy/Desktop/THESIS/3dWaterViz/2_zeros/"
path1 =
"/Users/darcy/Desktop/THESIS/3dWaterViz/1_zero/"
path4 =
"/Users/darcy/Desktop/THESIS/3dWaterViz/0_zero/"



#'bldgData', 'sqft', 'bldgId', 'bldgIndex'
#'water','index','elec','gas','time'
def readJson(filename):
   file = open(path0 + filename, 'r')
   #print file.readlines()
   j = json.load(file)
   #print j
   if (len(j['bldgData']) < 12 * 8):
        os.remove(path0+filename)
        return ("short", filename)

   #while (len())
   for i in range(len(j['bldgData'])):
        entry = j['bldgData'][i]
        if (entry['time'] < 201607):
             #if (entry['water'] == 0)

             if (entry['water'] == entry['gas'] ==
entry['elec'] == 0):
```

```
                    copyfile(path0+filename,
path3+filename)
                    os.remove(path0+filename)
                    return ("3", filename)

               elif (entry['water'] == entry['gas']
== 0
                    or entry['elec'] ==
entry['gas'] == 0
                    or entry['water'] ==
entry['elec'] == 0):
                    copyfile(path0+filename,
path2+filename)
                    os.remove(path0+filename)
                    return ("2", filename)

               elif (entry['water']== 0
                    or entry['elec'] == 0
                    or entry['gas'] == 0):
                    copyfile(path0+filename,
path1+filename)
                    os.remove(path0+filename)
                    return ("1", filename)

               else:
                    copyfile(path0+filename,
path4+filename)
                    os.remove(path0+filename)
                    return ("0", filename)



   return




def readAll():
   nameList = os.listdir(path0);


   while (nameList != []):
        if (nameList[0] == ".DS_Store"):
             nameList.remove(nameList[0])

        if (nameList[0] != ".DS_Store"):
             message, filename =
readJson(nameList[0])
```

```
            nameList.remove(filename)
            print filename, 'removed'


   return

   #print l

   # for i in range(len(nameList)):
   #     name = nameList[i]
   #     if (name != ".DS_Store"):
   #           readJson(name)
   #           print i, 'is done!'

   return



readAll()
#readJson('/AK0000AA_727340_viz.txt', path0,
path3, path2, path1, path4)
```

---

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-
import csv, os, json
from pprint import pprint
from shutil import copyfile

# Make it work for Python 2+3 and with Unicode
import io
try:
    to_unicode = unicode
except NameError:
    to_unicode = str

'''
STEP 5
Using the data from the best two buckets and
format them so that
they can be loaded into the visualization

o_zero:4
```

```
1_zero:632
Make the connection between data and states /
geolocation
'''


pathin =
"/Users/darcy/Desktop/THESIS/3dWaterViz/GeoPairing
/"
pathout =
"/Users/darcy/Desktop/THESIS/3dWaterViz/GeoPairedc
opy/"
path = "/Users/darcy/Desktop/THESIS/3dWaterViz/"

states =
["AK","AL","AZ","AR","CA","CO","CT","DE","FL","GA"
,"HI","IL",
"ID","IN","IA","KS","KY","LA","ME","MD","MA","MI",
"MN","MS","MO","MT","NE",
"NV","NH","NJ","NM","NY","NC","ND","OH","OK","OR",
"PA","RI","SC","SD","TN",
"TX","UT","VT","VA","WA","WV","WI","WY"];


allBldgData = [];
dataByState = dict()

waterDataByState = dict()
gasDataByState = dict()
elecDataByState = dict()

#'bldgData', 'sqft', 'bldgId', 'bldgIndex'
#'water','index','elec','gas','time'

def geoPairing():
   nameList = os.listdir(pathin);

   for i in range(len(nameList)):
         if (nameList[i] == ".DS_Store" or
nameList[i] == 'GeoPaired'):
               continue
         else:
               json_filein = pathin + nameList[i]
               # #print file.readlines()
               # j = json.load(file)
               # print j
               #print json_filein
               with open(json_filein) as json_file:
```

```
                    json_decoded =
json.load(json_file)
                    #print json_decoded
                    state =
json_decoded['bldgId'][:2]
                    data = json_decoded['bldgData']

                    #print data[0]
                    if (state == 'DC'):
                         state = 'MD'
                         json_decoded['DC'] = True

                    stateIndex =
states.index(state)

                    if stateIndex not in
dataByState.keys():
                         print 'a'

                         dataByState[stateIndex] =
[]

dataByState[stateIndex].append(data)

                    else:

dataByState[stateIndex].append(data)



                    json_decoded['state'] = state
                    json_decoded['stateIndex'] =
stateIndex
                    #print state, stateIndex
                    allBldgData.append((data,
stateIndex));

                    #save updated json file into
the new folder Geopaired
                    json_fileout = pathout +
nameList[i]
                    with io.open(json_fileout, 'w',
encoding='utf8') as outfile:
                        str_ =
json.dumps(json_decoded,
                                           indent=4,
sort_keys=True,
```

```
separators=(',', ': '), ensure_ascii=False)

outfile.write(to_unicode(str_))

                  # Read JSON file
                  with open(json_fileout) as
data_file:
                      data_loaded =
json.load(data_file)

                  #print(json_decoded ==
data_loaded)


   return



def getFileNames():
   tempList = []
   for fn in os.listdir(pathout):
        if (fn != ".DS_Store"):
             file_name = fn
             print file_name+","
             tempList.append(file_name+",")
   with open('path + "viz_file_names.csv', 'wb')
as csvfile:
        writer = csv.writer(csvfile,
delimiter=',',
                        quotechar='|',
quoting=csv.QUOTE_MINIMAL)
        writer.writerow(tempList)
        csvfile.close()


   return


def saveAllWaterData():
   for key in dataByState:
        waterSum = 0
        #print key, dataByState[key][0][0]
        for i in range(len(dataByState[key])):
             for j in
range(len(dataByState[key][i])):
```

```
                    waterSum +=
dataByState[key][i][j]["elec"]
                    print
dataByState[key][i][j]["water"]
        waterDataByState[key] = waterSum
/len(dataByState[key])
        print waterSum
   return



geoPairing()


#dataByState: dict of stateindex as keys and 2d
list as values

print dataByState[0], len(dataByState)
#getFileNames()

#saveAllBldgData()

saveAllWaterData()
print waterDataByState, len(waterDataByState)
```

Part 2: Data Visualization

```
<!DOCTYPE html>
<html lang="en">
   <head>
        <title>Map view</title>
        <meta charset="utf-8">
        <meta name="viewport"
content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
        <style>
```

```
            body {
                font-family: Monospace;
                background-color: #cccccc;
                margin: 0px;
                overflow: hidden;
            }
/*
            a {
                font-size: 50px
            }
*/
            a, button {
                display: inline-block;
                position: relative;
                z-index: 1;
                padding: .2em;
                margin: .1em;

            }
        </style>
    </head>
    <body>
        <a>Dashboard</a>
        <br>
        <button
onclick="location.href='3dWaterVizfly.html'"
type="button">Reset All</button>
        <button id="camerareset">Reset
Camera</button>
        <button id="resetmap">Reset Map</button>
        <button id="editmap" >Play with
Map</button>
        <button id="colorbytemp">Color Map by
Tempreture</button>
        <br>
        <button id="showWater">Show Water Data by
Building</button>
        <button id="showwdbs">Show Water Data in
States</button>


        <button id="showgdbs">Show Gas Data in
States</button>
        <button id="showedbs">Show Electricity
Data in States</button>
        <button id="showAlldbs">Show All Data in
States</button>
```

```
        <button
onclick="location.href='3dWaterVizfly.html'"
type="button">Hide All Data and Reset</button>

        <br>
        <button id="shot">Screenshot</button>
        <button id="mapcoordinates">Save Map
Settings</button>

        <button id="comparestate">Compare
Mode</button>
        <button
onclick="location.href='3dWaterVizfly2.html'"
type="button">View Indivisual Buildings</button>

<!--
control
https://github.com/mrdoob/three.js/blob/master/exa
mples/webgl_geometry_spline_editor.html-->
<!--        new set of x, y value to show the
breaking down effect-->

<!--        <a href="3dWaterViz2.html"
id="editmap">Edit map off</a>-->

        <script
src="three.js-master/build/three.js"></script>
        <script
src="https://d3js.org/d3.v4.min.js"></script>
        <script
src="js/controls/FlyControls.js"></script>

<!--
        <script type="text/javascript"
src="three.js-master/build/three.min.js"></script>
        <script type="text/javascript"
src="three.js-master/build/three-vr-viewer.js"></s
cript>

-->

        <script
src="js/controls/DragControls.js"></script>
        <script
src="js/controls/TrackballControls.js"></script>
        <script src="js/Detector.js"></script>
```

```
        <script
src="three.js-master/examples/js/loaders/STLLoader
.js"></script>
        <script
src="js/libs/stats.min.js"></script>
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.2.1/jquery.min.js"></script>

        <script>
            //TO DO
            //change tex when click, build
dashboard

//document.getElementById("peep").innerHTML =
"Welcome " + name;
            //move camera according to cursor
            //find tempreture data and map to map
            //change background and flow between
modes
            //comparing mode and small multiple
mode
            //design building signature
            //only drag in 2d

            const states =
["AK","AL","AZ","AR","CA","CO","CT","DE","FL","GA"
,"HI","IL","ID","IN","IA","KS","KY","LA","ME","MD"
,"MA","MI","MN","MS","MO","MT","NE","NV","NH","NJ"
,"NM","NY","NC","ND","OH","OK","OR","PA","RI","SC"
,"SD","TN","TX","UT","VT","VA","WA","WV","WI","WY"
];

            const stateIndex =
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43,44,45,46,47,48,49,50];



            const xpstn =
[-771.5195426637397,192.9120623696258,-380.5345212
538828,62.55575967266884,-578.3734923627312,-234.9
815409434305,455.2701253106838,425.2814168558325,2
18.36344644986104,253.7395028474221,-285.674405220
41346,120.43517222396176,-368.1286023540158,187.45
4144117512,22.62435981878511,-90.34250818635275,16
3.75919090056604,77.3213723218617,468.618133109434
```

```
27,352.5637913514511,455.2592937680254,133.0673679
9868054,12.411404639514473,124.83274671877716,37.6
8743829619295,-334.7916130692514,-123.645291875500
59,-468.97549792092764,457.3270990773462,426.34027
44226916,-255.9633981260098,327.05607636604043,274
.4917707300043,-108.04654217758784,246.18830760862
39,-118.58072822073825,-501.49653486182626,325.986
6073420785,491.3626630036835,298.20059332728704,-1
18.08152677724465,151.53222017545247,-220.66269959
637145,-337.2844614009403,440.4581975225201,276.47
65650848281,-481.25521056718935,291.19910626454407
,90.9059015422269,-258.190187286931];

            const ypstn =
[-244.73866881936206,-170.95915978327184,-122.8856
3450598433,-102.99262542548735,-73.47697016716467,
2.7706313421242825,145.7822451956747,77.5778746760
653,-320.163066823004,-162.14050818154948,-329.702
3565104875,1.3612283689822107,147.3054252059451,24
.92761366302522,85.66417768067507,-1.5394908753911
665,-9.189052794202816,-208.65012259352386,199.301
13656936916,48.430856172151834,154.488127852715,11
8.44670734139257,165.42389528007405,-176.329537268
3515,-27.281742794884494,195.22022354102592,75.852
3620509458,-13.290833751604424,194.28437725247764,
89.12685540976432,-141.97420200078307,111.08089832
317293,-67.75510784306641,229.2486167364129,46.130
26605921847,-89.50964071151921,173.4100614319632,9
0.03061465340473,160.9310187192081,-111.2944394801
1468,139.21921078942634,-50.04576678050063,-293.02
84193100634,19.84758308148158,193.96006796192128,2
.9471042530664704,266.18333415989605,18.9396464383
3037,137.71526067747104,109.38877671240826];

            var mapEditMode = false;

              var container, stats;
              var camera, controls, scene,
raycaster, renderer;
              var objects = [];
            var statesStl = [];
            var wdbs = [];
            var gdbs = [];
            var edbs = [];
            var map = [];
            var buildingCount = 20;
              init();
              animate();
```

```
            const material = new
THREE.MeshPhongMaterial( { color: 0x585858,
specular: 0x111111, shininess: 200 } );


            const materialx = new
THREE.MeshPhongMaterial( { color: 0xe7e7e7,
specular: 0x111111, shininess: 200 } );

            var lineMaterial = new
THREE.LineBasicMaterial({ color: 0xffffff });


            var waterDataNames = [];
            var waterDataNamesBS;
            var gasDataNamesBS;
            var elecDataNamesBS;
            var bldgIndex, sqft, bldgId, bldgData,
bldgState, bldgStateIndex;

            const material0 = new
THREE.MeshBasicMaterial( { color: 0xA2D9F0} );

            const material1 = new
THREE.MeshBasicMaterial( { color: 0xF8EFB6} );

            const material2 = new
THREE.MeshBasicMaterial( { color: 0xBFCCF9} );
            const dist0 = 0;
            const dist1 = 10;
            const dist2 = 20;


            var mouse;
            var INTERSECTED;
             var radius = 100, theta = 0;


            function drawWaterData(dataPoints) {
                //loop through all buildings in
the name list
                for (var i = 0; i <
waterDataNames.length; i++) {
                    d3.json(waterDataNames[i],
function(error, dataPoints) {
                        //console.log(dataPoints)
```

```
                        bldgIndex =
+dataPoints.bldgIndex;
                        sqft = +dataPoints.sqft;
                        bldgId =
dataPoints.bldgId;
                        bldgData =
dataPoints.bldgData;
                        bldgState =
dataPoints.state;
                        bldgStateIndex =
dataPoints.stateIndex;


                        var xb, yb, zb;
                        zb = 3;
                        xb =
stateXs[bldgStateIndex];
                        yb =
stateYs[bldgStateIndex];

                        var geometry = new
THREE.BoxGeometry( 10, 10, 0 );
                           for ( var i = 0; i <
buildingCount; i ++ ) {
                            const material0 = new
THREE.MeshBasicMaterial( { color: 0xffffff} );
                            var object = new
THREE.Mesh( geometry, material0);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
                            object.position.x =
xb+100;// + 50*Math.random();//Math.random() *
1000 - 500;
                            object.position.y =
yb+100;// + 50*Math.random();//Math.random() * 600
- 300;
                            object.position.z =
zb;//boxz;//0;//Math.random() * 800 - 400;
                            object.rotation.x =
0;//Math.random() * 2 * Math.PI;
                            object.rotation.y =
0;//Math.random() * 2 * Math.PI;
                            object.rotation.z =
0;//Math.random() * 2 * Math.PI;
                            object.scale.x =
1;//Math.random() * 2 + 1;
                            object.scale.y =
1;//Math.random() * 2 + 1;
```

```
                            object.scale.z =
1;//Math.random() * 2 + 1;
                            object.castShadow =
true;
                            object.receiveShadow =
false;
                            scene.add( object );
                            objects.push( object
);
                        }



                        var x, y, z;
                        z = 3;
                        x =
stateXs[bldgStateIndex];
                        y =
stateYs[bldgStateIndex];
                        var lineGeometry = new
THREE.Geometry();

//lineGeometry.vertices.push(new THREE.Vector3(x,
y, z));


                        //loop through all points
for each building
                        for (var j = 0; j <
bldgData.length; j++) {
                            x =
stateXs[bldgStateIndex] + bldgData[j].index;
                            y =
stateYs[bldgStateIndex] + bldgData[j].water*20;

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
                        }
                        y =
stateYs[bldgStateIndex];

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
                        x =
stateXs[bldgStateIndex];
```

```
lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));


                        var lines = new
THREE.Line(lineGeometry, lineMaterial);
                        scene.add(lines);

                        console.log(lines);

                        var lineMesh = new
THREE.Mesh( lineGeometry, material);
                        scene.add(lineMesh);
                        console.log(lineMesh);

                    });
                }
            }

            function showWaterDataByState() {

console.log("drawWaterDataEachState");
                d3.json("./wdbs.txt",
function(error, root) {
                    //console.log("error:", error)
                    console.log("load water by
state...");
                    console.log(root.length);
                    waterDataNamesBS = root;
                    //getStateDataPosition(root,
dist0, material0);
                    for (var j = 0; j <
wdbs.length; j++) {
                        scene.remove(wdbs[j]);
                    }
                    wdbs = [];

                    for (var i = 0; i < 50; i++) {
                    //console.log(root[i]);

                        var geometry = new
THREE.BoxGeometry( root[i]/20, 20, 10 );

                        var object = new
THREE.Mesh( geometry, material0);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
```

```
                        object.position.x =
statesStl[i].position.x+80 ;
                        object.position.y =
statesStl[i].position.y+80+ dist0;
                        object.position.z = 10;
                        object.rotation.x =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.y =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.z =
0;//Math.random() * 2 * Math.PI;
                        object.scale.x =
1;//Math.random() * 2 + 1;
                        object.scale.y =
1;//Math.random() * 2 + 1;
                        object.scale.z =
1;//Math.random() * 2 + 1;
                        object.castShadow = true;
                        object.receiveShadow =
false;
                        scene.add( object );
                        wdbs.push( object );
                        objects.push( object );

                    }
                });
            }


            function showGasDataByState() {
                d3.json("./gdbs.txt",
function(error, root) {
                    //console.log("error:", error)
                    console.log("load gas...");
                    console.log(root);
                    gasDataNamesBS = root;
                    //getStateDataPosition(root,
dist1, material1);
                    for (var j = 0; j <
gdbs.length; j++) {
                        scene.remove(gdbs[j]);
                    }
                    gdbs = [];

                    for (var i = 0; i < 50; i++) {
                    //console.log(root[i]);
```

```
                        var geometry = new
THREE.BoxGeometry( root[i]/20, 20, 10 );

                        var object = new
THREE.Mesh( geometry, material1);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
                        object.position.x =
statesStl[i].position.x+80 ;
                        object.position.y =
statesStl[i].position.y+80+ dist1*1.5;
                        object.position.z = 10;
                        object.rotation.x =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.y =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.z =
0;//Math.random() * 2 * Math.PI;
                        object.scale.x =
1;//Math.random() * 2 + 1;
                        object.scale.y =
1;//Math.random() * 2 + 1;
                        object.scale.z =
1;//Math.random() * 2 + 1;
                        object.castShadow = true;
                        object.receiveShadow =
false;
                        scene.add( object );
                        gdbs.push( object );
                        objects.push( object );

                    }
                });
            }


            function showElecDataByState() {
                d3.json("./edbs.txt",
function(error, root) {
                    //console.log("error:", error)
                    console.log("load elec...");
                    console.log(root);
                    elecDataNamesBS = root;
                    //getStateDataPosition(root,
dist2, material2);
                    for (var j = 0; j <
edbs.length; j++) {
                        scene.remove(edbs[j]);
```

```
                    }
                    edbs = [];

                    for (var i = 0; i < 50; i++) {
                    //console.log(root[i]);

                        var geometry = new
THREE.BoxGeometry( root[i]/10, 20, 10 );

                        var object = new
THREE.Mesh( geometry, material2);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
                        object.position.x =
statesStl[i].position.x+80 ;
                        object.position.y =
statesStl[i].position.y+80+ dist2*1.5;
                        object.position.z = 10;
                        object.rotation.x =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.y =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.z =
0;//Math.random() * 2 * Math.PI;
                        object.scale.x =
1;//Math.random() * 2 + 1;
                        object.scale.y =
1;//Math.random() * 2 + 1;
                        object.scale.z =
1;//Math.random() * 2 + 1;
                        object.castShadow = true;
                        object.receiveShadow =
false;
                        scene.add( object );
                        edbs.push( object );
                        objects.push( object );

                    }
                });
            }


            function showAllDataByState() {
                showWaterDataByState();
                showGasDataByState();
                showElecDataByState();
            }
```

```
            function showWaterData() {
                d3.csv("./viz_file_names.csv",
function(error, dataPoints) {
                  //
console.log(dataPoints.columns[0]);
                    waterDataNames =
dataPoints.columns;
                    //console.log(waterDataNames);
                    drawWaterData(dataPoints);
                })
            }



            function makeBoxes() {
                const boxz = 0;

                var geometry = new
THREE.BoxGeometry( 10, 10, 0 );
                   for ( var i = 0; i <
buildingCount; i ++ ) {
                    const material0 = new
THREE.MeshBasicMaterial( { color: 0xffffff} );
                        var object = new
THREE.Mesh( geometry, material0);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
                        object.position.x =
Math.random() * 1000 - 500;
                        object.position.y =
Math.random() * 600 - 300;
                        object.position.z =
boxz;//0;//Math.random() * 800 - 400;
                        object.rotation.x =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.y =
0;//Math.random() * 2 * Math.PI;
                        object.rotation.z =
0;//Math.random() * 2 * Math.PI;
                        object.scale.x =
1;//Math.random() * 2 + 1;
```

```
                        object.scale.y =
1;//Math.random() * 2 + 1;
                        object.scale.z =
1;//Math.random() * 2 + 1;
                        object.castShadow = true;
                        object.receiveShadow =
false;
                        scene.add( object );
                        objects.push( object );

//                      var lineMaterial = new
THREE.LineBasicMaterial({ color: 0xffffff });
//                      var lineGeometry = new
THREE.Geometry();
//                      var x = object.position.x;
//                      var y = z = 5000;
//
lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
//                      var line = new
THREE.Line(lineGeometry, lineMaterial);
//                      scene.add( line );
//                      lines.push(line);
                    }
                }
```

```
            function makeMap() {
                var loader = new
THREE.STLLoader();
                //stl 3d model from
https://www.thingiverse.com/thing:1524543
                var counter = 0;
                for ( var i = 0; i <
states.length; i ++ ) {
                    loader.load(
'usmap/'+states[i]+'.stl', function ( geometry ) {
                        var mesh = new THREE.Mesh(
geometry, material );
                        mesh.position.set(
stateXs[counter], stateYs[counter], -50 );

//x:-1200+250*(counter%10)
y:200+180*Math.floor(counter/10)
```

```
                        //console.log(i,
mesh.position.y);
                        mesh.rotation.set(
Math.PI/2, 0, Math.PI*2);
                        mesh.scale.set(.6, .6,.6);
                        mesh.castShadow = true;
                        mesh.receiveShadow = true;
                        scene.add(mesh);
                        statesStl.push(mesh);

                        counter ++;
                    } );
                }
            }



            function setClickListeners() {
//
document.getElementById("resetall").addEventListen
er('click', init);

document.getElementById("shot").addEventListener('
click', takeScreenshot);
document.getElementById("camerareset").addEventLis
tener('click', cameraReset);

document.getElementById("mapcoordinates").addEvent
Listener('click', mapCoordinates);

document.getElementById("resetmap").addEventListen
er('click', resetMap);

document.getElementById("editmap").addEventListene
r('click', editMap);

document.getElementById("colorbytemp").addEventLis
tener('click', colorByTemp);

document.getElementById("showWater").addEventListe
ner('click', showWaterData);

document.getElementById("showwdbs").addEventListen
er('click', showWaterDataByState);
```

```
document.getElementById("showgdbs").addEventListen
er('click', showGasDataByState);

document.getElementById("showedbs").addEventListen
er('click', showElecDataByState);


document.getElementById("showAlldbs").addEventList
ener('click', showAllDataByState);


document.addEventListener('keydown', onkeydown,
false );
//
document.addEventListener('mousedown',
updatemappo, false );
                document.addEventListener(
'mousemove', onDocumentMouseMove, false );


            }



              function init() {
                 stateXs = xpstn;
                 stateYs = ypstn;

                    container =
document.createElement( 'div' );
                    document.body.appendChild(
container );
                    camera = new
THREE.PerspectiveCamera( 50, window.innerWidth /
window.innerHeight, 1, 10000 );
                    camera.position.z = 1000;
                 //console.log(camera);
                 //var oriCamera =
JSON.parse(JSON.stringify( camera ));



                    controls = new
THREE.TrackballControls( camera );
                    controls.rotateSpeed = 1.0;
                    controls.zoomSpeed = 1.2;
                    controls.panSpeed = 0.8;
```

```
                    controls.noZoom = false;
                    controls.noPan = false;
                    controls.staticMoving = false;
                    controls.dynamicDampingFactor =
0.3;

                    scene = new THREE.Scene();
                    scene.background = new
THREE.Color( 0x898989 );
                    scene.add( new
THREE.AmbientLight( 0x505050 ) );
                    var light = new
THREE.SpotLight( 0xB8B8B8, 0.5 );
                    light.position.set( 0, 500,
2000 );
                    light.castShadow = true;
                    light.shadow = new
THREE.LightShadow( new THREE.PerspectiveCamera(
50, 1, 200, 10000 ) );
                    light.shadow.bias = - 0.00022;
                    light.shadow.mapSize.width =
2048;
                    light.shadow.mapSize.height =
2048;
                    scene.add( light );



                ////////////////////make
boxes/////////////////

//                  makeBoxes();


                mouse = new THREE.Vector2();

                raycaster = new THREE.Raycaster();


                    renderer = new
THREE.WebGLRenderer( { antialias: true } );
                    renderer.setPixelRatio(
window.devicePixelRatio );
                    renderer.setSize(
window.innerWidth, window.innerHeight );
                    renderer.shadowMap.enabled =
true;
```

```
                renderer.shadowMap.type =
THREE.PCFShadowMap;
                container.appendChild(
renderer.domElement );


            setClickListeners();

                var dragControls = new
THREE.DragControls( objects, camera,
renderer.domElement );
                dragControls.addEventListener(
'dragstart', function ( event ) { controls.enabled
= false; } );
                dragControls.addEventListener(
'dragend', function ( event ) { controls.enabled =
true; } );




                var info =
document.createElement( 'div' );
                info.style.position =
'absolute';
                info.style.top = '10px';
                info.style.width = '100%';
                info.style.textAlign = 'left';
            info.innerHTML =
'<br><br><br><br><br><a
href="3dWaterVizfly2.html">View Data in
Topographical Map</a><br><a>W: zoom
in</a><br><a>S: zoom out</a><br><a>A: turn
left</a><br><a>D: turn right</a><br><a><: move
screen to the left</a><br><a>>: move screen to the
right</a><br><a>Hover on model to
lock;</a><br><a>Click and drag to move
model.</a>';
            container.appendChild( info );
                //stats = new Stats();
                //container.appendChild(
stats.dom );
                //
                window.addEventListener(
'resize', onWindowResize, false );
```

```
                ////////////map
settings///////////////

                makeMap();


            }

            function onWindowResize() {
                camera.aspect =
window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();
                renderer.setSize(
window.innerWidth, window.innerHeight );
            }

           function onDocumentMouseMove( event )
{
                //event.preventDefault();
                mouse.x = ( event.clientX /
window.innerWidth ) * 2 - 1;
                mouse.y = - ( event.clientY /
window.innerHeight ) * 2 + 1;
            }


            function animate() {
                requestAnimationFrame( animate
);
                render();
            }

            function render() {
                controls.update();
                renderer.render( scene, camera
);

               raycaster.setFromCamera( mouse,
camera );
                var intersects =
raycaster.intersectObjects( scene.children);

                if ( intersects.length > 0 ) {
```

```
                        if ( INTERSECTED !=
intersects[ 0 ].object ) {
                                if ( INTERSECTED )
INTERSECTED.material = material;
                                INTERSECTED =
intersects[ 0 ].object;
                                INTERSECTED.material
= materialx;

//INTERSECTED.material.emissive.setHex( 0xff0000
);

///console.log("intersected 1");

//console.log(INTERSECTED);

                        }
                    }

                else {
                    if ( INTERSECTED )
INTERSECTED.material = material;

                     INTERSECTED = null;
                     //console.log("intersected
2");

                    }
                    renderer.render( scene, camera
);



             }

            function takeScreenshot() {
                console.log("taking
screeshot...");
                // open in new window like this
                var w = window.open('', '');
                w.document.title = "Screenshot";

//w.document.body.style.backgroundColor = "red";
                var img = new Image();
                // Without 'preserveDrawingBuffer'
set to true, we must render now
                renderer.render(scene, camera);
```

```
                img.src =
renderer.domElement.toDataURL();
                w.document.body.appendChild(img);
                // download file
                var a =
document.createElement('a');
                // Without 'preserveDrawingBuffer'
set to true, we must render now
                renderer.render(scene, camera);
                a.href =
renderer.domElement.toDataURL().replace("image/png
", "image/octet-stream");
                a.download = 'canvas.png';
                a.click();
            }

            function cameraReset() {
                console.log("Reseting camera...");
                //console.log(camera);
                camera.position.x = 0;
                camera.position.y = 0;
                camera.position.z = 1000;
                camera.rotation.x = 0;
                camera.rotation.y = 0;
                camera.rotation.z = 0;
                camera.fov = 50;
                camera.filmOffset = 0;
                camera.updateProjectionMatrix();
                controls.reset();
                //window.innerWidth /
window.innerHeight, 1, 10000
            }


            function mapCoordinates() {
                //statesStl = [];
                var xs = [];
                var ys = [];
                //console.log(statesStl.length,
buildingCount);
                //locate all state stl files
                for (var i = 0; i <
statesStl.length; i++) {
                    //statesStl.push(objects[i]);

xs.push(statesStl[i].position.x);

ys.push(statesStl[i].position.y);
```

```
                    console.log(statesStl.length);

                }
                console.log(statesStl.length);

                const linkx =
document.createElement( 'a' );
                linkx.style.display = 'none';
                document.body.appendChild( linkx
);

                const blobx = new Blob( [ xs ], {
type: 'text/plain' } );
                const objectURLx =
URL.createObjectURL( blobx );

                linkx.href = objectURLx;
                linkx.href = URL.createObjectURL(
blobx );
                linkx.download =
'x_map_setting.json';

                linkx.click();



                const linky =
document.createElement( 'a' );
                linky.style.display = 'none';
                document.body.appendChild( linky
);

                const bloby = new Blob( [ ys ], {
type: 'text/plain' } );
                const objectURLy =
URL.createObjectURL( bloby );

                linky.href = objectURLy;
                linky.href = URL.createObjectURL(
bloby );
                linky.download =
'y_map_setting.json';

                linky.click();
```

```
//https://discourse.threejs.org/t/how-to-create-a-new-file-and-save-it-with-arraybuffer-content/628/3

            }

            const zzz = -100;

            function resetMap() {
                var counter = 0;
                for (var i = 0; i < statesStl.length; i++) {
                    statesStl[i].position.x = xpstn[counter];
                    statesStl[i].position.y = ypstn[counter];
                    statesStl[i].position.z = zzz;//-100;
                    statesStl[i].material = material;

                    counter++;
                }

            }

            function colorByTemp() {
                var material2 = new THREE.MeshPhongMaterial( { color: 0xffffff, specular: 0x111111, shininess: 20 } );
                var counter = 0;
                for (var i = 0; i < statesStl.length; i++) {
                    statesStl[i].material = material2;
                    counter++;
                }
            }


            function editMap() {
                if (mapEditMode == false) {
                    mapEditMode = true;
                    var dragControlsMap = new THREE.DragControls( statesStl, camera, renderer.domElement );
```

```
dragControlsMap.addEventListener( 'dragstart',
function ( event ) { controls.enabled = false; }
);

dragControlsMap.addEventListener( 'dragend',
function ( event ) { controls.enabled = true; } );
                }
                else if (mapEditMode == true){
                    mapEditMode = false;

                    //controls.enabled = true;
                   // dragControlsMap.abort;

                    //TODO
                }

                console.log("edit mode:",
mapEditMode);

            }


            var mode;
            var MODE = { TRACKBALL: 0, FLY: 1 };

            function onkeydown() {
               var keyCode = event.which;
                console.log("aaa");

                if (keyCode == 87) {
                    console.log("up");
                    //camera.position.y += 10;
                    //camera.zoom += 2;
                    camera.fov -= 3;
                } else if (keyCode == 83) {
                    console.log("down");
                    camera.fov += 3;
                } else if (keyCode == 65) {
                    camera.position.x -= 10;
                } else if (keyCode == 68) {
                    camera.position.x += 10;
                } else if (keyCode == 32) {
                    cameraReset();
             } else if (keyCode == 188) {
                   camera.filmOffset -= 1;
                } else if (keyCode == 190) {
                   camera.filmOffset += 1;
```

```
                }
                camera.updateProjectionMatrix();
            }


            //https://jsfiddle.net/2pha/art388yv/
            //chrome://flags/#enable-webvr
                        //VRViewer({THREE});
//python -m SimpleHTTPServer
            //localhost:8000
        </script>

   </body>
</html>
```

---

```
<!DOCTYPE html>
<html lang="en">
   <head>
        <title>Map view</title>
        <meta charset="utf-8">
        <meta name="viewport"
content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
        <style>
             body {
                  font-family: Monospace;
                  margin: 0px;
                  overflow: hidden;
             }
/*
            a {
                font-size: 50px
            }
*/
            a, button {
                display: inline-block;
                position: relative;
                z-index: 1;
                padding: .2em;
                margin: .1em;
```

```
            }
            #info {
   position: absolute;
   top: 10px;
   width: 100%;
   text-align: center;
   z-index: 100;
   display:block;
}
            body {
              width: 100%;
              height: 100%;

              background: #11e8bb; /* Old browsers
*/
              background:
-moz-linear-gradient(top,  #11e8bb 0%, #8200c9
100%); /* FF3.6-15 */
              background:
-webkit-linear-gradient(top,  #11e8bb 0%,#8200c9
100%); /* Chrome10-25,Safari5.1-6 */
              background: linear-gradient(to
bottom,  #11e8bb 0%,#8200c9 100%); /* W3C, IE10+,
FF16+, Chrome26+, Opera12+, Safari7+ */
              filter:
progid:DXImageTransform.Microsoft.gradient(
startColorstr='#11e8bb',
endColorstr='#8200c9',GradientType=0 ); /* IE6-9
*/
            }
        </style>
   </head>
   <body>
<!--        <a>Dashboard</a>-->
<!--        <br>-->
<!--
        <button
onclick="location.href='3dWaterVizfly.html'"
type="button">Reset All</button>
        <button id="camerareset">Reset
Camera</button>
        <button id="resetmap">Reset Map</button>
        <button id="editmap" >Play with
Map</button>
        <button id="colorbytemp">Color Map by
Tempreture</button>
        <br>
```

```
        <button id="showWater">Show Water Data on
Map</button>
        <button id="showWaterbldg">Show Water Data
by Building</button>

        <button
onclick="location.href='3dWaterVizfly.html'"
type="button">Hide All Data and Reset</button>

        <br>
        <button id="shot">Screenshot</button>
        <button id="mapcoordinates">Save Map
Settings</button>

        <button id="comparestate">Compare
Mode</button>

-->

<!--        <div id="info">Visualize Water Data By
Building</div>-->

<!--
control
https://github.com/mrdoob/three.js/blob/master/exa
mples/webgl_geometry_spline_editor.html-->
<!--        new set of x, y value to show the
breaking down effect-->

<!--        <a href="3dWaterViz2.html"
id="editmap">Edit map off</a>-->

        <script
src="three.js-master/build/three.js"></script>
        <script
src="https://d3js.org/d3.v4.min.js"></script>
        <script
src="js/controls/FlyControls.js"></script>
        <script
src="js/renderers/Projector.js"></script>
        <script
src="js/renderers/CanvasRenderer.js"></script>

<!--
        <script type="text/javascript"
src="three.js-master/build/three.min.js"></script>
```

```
        <script type="text/javascript"
src="three.js-master/build/three-vr-viewer.js"></s
cript>

-->

        <script
src="js/controls/DragControls.js"></script>
        <script
src="js/controls/TrackballControls.js"></script>
        <script src="js/Detector.js"></script>
        <script
src="three.js-master/examples/js/loaders/STLLoader
.js"></script>
        <script
src="js/libs/stats.min.js"></script>
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.2.1/jquery.min.js"></script>

        <script>
            //TO DO
            //change tex when click, build
dashboard

//document.getElementById("peep").innerHTML =
"Welcome " + name;
            //move camera according to cursor
            //find tempreture data and map to map
            //change background and flow between
modes
            //comparing mode and small multiple
mode
            //design building signature
            //only drag in 2d

            const states =
["AK","AL","AZ","AR","CA","CO","CT","DE","FL","GA"
,"HI","IL","ID","IN","IA","KS","KY","LA","ME","MD"
,"MA","MI","MN","MS","MO","MT","NE","NV","NH","NJ"
,"NM","NY","NC","ND","OH","OK","OR","PA","RI","SC"
,"SD","TN","TX","UT","VT","VA","WA","WV","WI","WY"
];

            const stateIndex =
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,3
6,37,38,39,40,41,42,43,44,45,46,47,48,49,50];
```

```
            const xpstn =
[-771.5195426637397,192.9120623696258,-380.5345212
538828,62.55575967266884,-578.3734923627312,-234.9
815409434305,455.2701253106838,425.2814168558325,2
18.36344644986104,253.7395028474221,-285.674405220
41346,120.43517222396176,-368.1286023540158,187.45
4144117512,22.62435981878511,-90.34250818635275,16
3.75919090056604,77.3213723218617,468.618133109434
27,352.5637913514511,455.2592937680254,133.0673679
9868054,12.411404639514473,124.83274671877716,37.6
8743829619295,-334.7916130692514,-123.645291875500
59,-468.97549792092764,457.3270990773462,426.34027
44226916,-255.9633981260098,327.05607636604043,274
.4917707300043,-108.04654217758784,246.18830760862
39,-118.58072822073825,-501.49653486182626,325.986
6073420785,491.3626630036835,298.20059332728704,-1
18.08152677724465,151.53222017545247,-220.66269959
637145,-337.2844614009403,440.4581975225201,276.47
65650848281,-481.25521056718935,291.19910626454407
,90.9059015422269,-258.190187286931];

            const ypstn =
[-244.73866881936206,-170.95915978327184,-122.8856
3450598433,-102.99262542548735,-73.47697016716467,
2.7706313421242825,145.7822451956747,77.5778746760
653,-320.163066823004,-162.14050818154948,-329.702
3565104875,1.3612283689822107,147.3054252059451,24
.92761366302522,85.66417768067507,-1.5394908753911
665,-9.189052794202816,-208.65012259352386,199.301
13656936916,48.430856172151834,154.488127852715,11
8.44670734139257,165.42389528007405,-176.329537268
3515,-27.281742794884494,195.22022354102592,75.852
3620509458,-13.290833751604424,194.28437725247764,
89.12685540976432,-141.97420200078307,111.08089832
317293,-67.75510784306641,229.2486167364129,46.130
26605921847,-89.50964071151921,173.4100614319632,9
0.03061465340473,160.9310187192081,-111.2944394801
1468,139.21921078942634,-50.04576678050063,-293.02
84193100634,19.84758308148158,193.96006796192128,2
.9471042530664704,266.18333415989605,18.9396464383
3037,137.71526067747104,109.38877671240826];

            var mapEditMode = false;

             var container, stats;
```

```
            var camera, controls, scene,
raycaster, renderer;
            var objects = [];
            var statesStl = [];
            var wdbs = [];
            var gdbs = [];
            var edbs = [];
            var map = [];
            var buildingCount = 20;
            init();
            animate();


            var w = window.innerWidth;
            var h = window.innerHeight;
            var fullWidth = w * 1;
            var fullHeight = h * 1;

            const material = new
THREE.MeshPhongMaterial( { color: 0xA2D9F0,
specular: 0x111111, shininess: 200 } );
            const materialx = new
THREE.MeshPhongMaterial( { color: 0xe7e7e7,
specular: 0x111111, shininess: 200 } );

            var lineMaterial = new
THREE.LineBasicMaterial({ color: 0xffffff });


            var waterDataNames = [];
            var waterDataNamesBS;
            var gasDataNamesBS;
            var elecDataNamesBS;
            var bldgIndex, sqft, bldgId, bldgData,
bldgState, bldgStateIndex;

            const material0 = new
THREE.MeshBasicMaterial( { color: 0xA2D9F0} );
            const material01 = new
THREE.MeshBasicMaterial( { color: 0xfbab47} );


            const material1 = new
THREE.MeshBasicMaterial( { color: 0xF8EFB6} );

            const material2 = new
THREE.MeshBasicMaterial( { color: 0xBFCCF9} );
            const dist0 = 0;
```

```
            const dist1 = 10;
            const dist2 = 20;



            //select vars
//            var raycaster;
//            var mouse;
//
//            var pickingData = [],
pickingTexture, pickingScene;
//
//            var highlightBox;
//            var mouse = new THREE.Vector2();
//            var offset = new THREE.Vector3( 10,
10, 10 );
            var mouse;
            var INTERSECTED;
             var radius = 100, theta = 0;

            //select vars end


            var effectController = {
                  showDots: true,
                  showLines: true,
                  minDistance: 150,
                  limitConnections: false,
                  maxConnections: 20,
                  particleCount: 500
             };




            function drawWaterData(dataPoints) {
                //loop through all buildings in
the name list
                for (var i = 0; i <
waterDataNames.length; i++) {
                    d3.json(waterDataNames[i],
function(error, dataPoints) {
                        //console.log(dataPoints)
                        bldgIndex =
+dataPoints.bldgIndex;
                        sqft = +dataPoints.sqft;
                        bldgId =
dataPoints.bldgId;
```

```
                        bldgData =
dataPoints.bldgData;
                        bldgState =
dataPoints.state;
                        bldgStateIndex =
dataPoints.stateIndex;


                        var xb, yb, zb;
                        zb = 3;
                        xb =
stateXs[bldgStateIndex];
                        yb =
stateYs[bldgStateIndex];
//
//                        var geometry = new
THREE.BoxGeometry( 10, 10, 0 );
//                         for ( var i = 0; i <
buildingCount; i ++ ) {
//                            const material0 =
new THREE.MeshBasicMaterial( { color: 0xffffff} );
//                            var object = new
THREE.Mesh( geometry, material0);//new
THREE.MeshStandardMaterial( { color: 0xa31a28 } )
);//Math.random() *
//                            object.position.x =
xb;// + 50*Math.random();//Math.random() * 1000 -
500;
//                            object.position.y =
yb;// + 50*Math.random();//Math.random() * 600 -
300;
//                            object.position.z =
zb;//boxz;//0;//Math.random() * 800 - 400;
//                            object.rotation.x =
0;//Math.random() * 2 * Math.PI;
//                            object.rotation.y =
0;//Math.random() * 2 * Math.PI;
//                            object.rotation.z =
0;//Math.random() * 2 * Math.PI;
//                            object.scale.x =
1;//Math.random() * 2 + 1;
//                            object.scale.y =
1;//Math.random() * 2 + 1;
//                            object.scale.z =
1;//Math.random() * 2 + 1;
//                            object.castShadow =
true;
```

```
//                              object.receiveShadow
= false;
//                              scene.add( object );
//                              objects.push( object
);
//                          }



                            var x, y, z;
                            z = 3;
                            x =
stateXs[bldgStateIndex];
                            y =
stateYs[bldgStateIndex];
                            var lineGeometry = new
THREE.Geometry();

//lineGeometry.vertices.push(new THREE.Vector3(x,
y, z));



                            //loop through all points
for each building
                            for (var j = 0; j <
bldgData.length; j++) {
                                x =
stateXs[bldgStateIndex] + bldgData[j].index;
                                y =
stateYs[bldgStateIndex] + bldgData[j].water*20;

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
                            }
                            y =
stateYs[bldgStateIndex];

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
                            x =
stateXs[bldgStateIndex];

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));
```

```
                        var lines = new
THREE.Line(lineGeometry, lineMaterial);
                        scene.add(lines);

                        console.log(lines);

                        var lineMesh = new
THREE.Mesh( lineGeometry, material);
                        scene.add(lineMesh);
                        console.log(lineMesh);

                    });
                }
            }


            function showWaterData() {
                while(scene.children.length > 0){

scene.remove(scene.children[0]);
                }
                d3.csv("./viz_file_names.csv",
function(error, dataPoints) {
                  //
console.log(dataPoints.columns[0]);
                    waterDataNames =
dataPoints.columns;
                    //console.log(waterDataNames);
                    drawWaterData(dataPoints);
                })
            }


            var idx = 0;

            function drawWaterData2(dataPoints) {

               // group = new THREE.Group();
                //group.position.y = 50;
                   //scene.add( group );
                   var loader = new
THREE.TextureLoader();
                   var texture = loader.load(
"wave.jpg" );
                   // it's necessary to apply
these settings in order to correctly display the
texture on a shape geometry
```

```
                    texture.wrapS = texture.wrapT =
THREE.RepeatWrapping;
                    texture.repeat.set( 0.008,
0.008 );

                  function addShape( shape,
extrudeSettings, color, x, y, z, rx, ry, rz, s ) {
                        // flat shape with texture
                        // note: default UVs
generated by ShapeBufferGeometry are simply the x-
and y-coordinates of the vertices
//                      var geometry = new
THREE.ShapeBufferGeometry( shape );
//                      var mesh = new THREE.Mesh(
geometry, new THREE.MeshPhongMaterial( { side:
THREE.DoubleSide} ) );//map: texture
//                      mesh.position.set( x, y, z
- 175 );
//                      mesh.rotation.set( rx, ry,
rz );
//                      mesh.scale.set( s, s, s );
//                      group.add( mesh );
                        // flat shape
//                      var geometry = new
THREE.ShapeBufferGeometry( shape );
//                      var mesh = new THREE.Mesh(
geometry, material0);//new
THREE.MeshPhongMaterial( { color: color, side:
THREE.DoubleSide } )
//                      mesh.position.set( x, y, z
- 125 );
//                      mesh.rotation.set( rx, ry,
rz );
//                      mesh.scale.set( s, s, s );
//                      group.add( mesh );
                        // extruded shape
                        var geometry = new
THREE.ExtrudeGeometry( shape, extrudeSettings );
                        var mesh = new THREE.Mesh(
geometry, material0 );
                        mesh.position.set( x, y,
z);
                        mesh.rotation.set( rx, ry,
rz );
                        mesh.scale.set( s, s, s );
                        //group.add( mesh );
                    scene.add(mesh);
                    objects.push( mesh );
```

```
                        //addLineShape( shape,
color, x, y, z, rx, ry, rz, s );
                    }
//                  function addLineShape( shape,
color, x, y, z, rx, ry, rz, s ) {
//                      // lines
//                      shape.autoClose = true;
//                      var points =
shape.getPoints();
//                      var spacedPoints =
shape.getSpacedPoints( 50 );
//                      var geometryPoints = new
THREE.BufferGeometry().setFromPoints( points );
//                      var geometrySpacedPoints =
new THREE.BufferGeometry().setFromPoints(
spacedPoints );
//                      // solid line
//                      var line = new THREE.Line(
geometryPoints, new THREE.LineBasicMaterial( {
color: color, linewidth: 3 } ) );
//                      line.position.set( x, y, z
- 25 );
//                      line.rotation.set( rx, ry,
rz );
//                      line.scale.set( s, s, s );
//                      group.add( line );
//                      // line from equidistance
sampled points
//                      var line = new THREE.Line(
geometrySpacedPoints, new THREE.LineBasicMaterial(
{ color: color, linewidth: 3 } ) );
//                      line.position.set( x, y, z
+ 25 );
//                      line.rotation.set( rx, ry,
rz );
//                      line.scale.set( s, s, s );
//                      group.add( line );
//                      // vertices from real
points
//                      var particles = new
THREE.Points( geometryPoints, new
THREE.PointsMaterial( { color: color, size: 4 } )
);
//                      particles.position.set( x,
y, z + 75 );
//                      particles.rotation.set(
rx, ry, rz );
```

```
//                      particles.scale.set( s, s,
s );
//                      group.add( particles );
//                      // equidistance sampled
points
//                      var particles = new
THREE.Points( geometrySpacedPoints, new
THREE.PointsMaterial( { color: color, size: 4 } )
);
//                      particles.position.set( x,
y, z + 125 );
//                      particles.rotation.set(
rx, ry, rz );
//                      particles.scale.set( s, s,
s );
//                      group.add( particles );
//                }

                idx = 0;
                //loop through all buildings in
the name list
                for (var i = 0; i <
waterDataNames.length; i++) {
//waterDataNames.length
                    idx += 1;
                    //console.log(idx);




                    d3.json(waterDataNames[i],
function(error, dataPoints) {
                        var index;
                        //console.log(idx);
                        bldgIndex =
+dataPoints.bldgIndex;
                        sqft = +dataPoints.sqft;
                        bldgId =
dataPoints.bldgId;
                        bldgData =
dataPoints.bldgData;
                        bldgState =
dataPoints.state;
                        bldgStateIndex =
dataPoints.stateIndex;


//console.log(waterDataNames);
```

```
                        var x, y, z;
                        x =
stateXs[bldgStateIndex];
                        y =
stateYs[bldgStateIndex];



                        for (var k = 0; k <
waterDataNames.length; k++) {
                           if
(waterDataNames[k].indexOf(bldgId) !== -1) {
                              //
console.log(bldgId);
                               z = k*20;
                           }
                       }




                        var lineGeometry = new
THREE.Geometry();

//lineGeometry.vertices.push(new THREE.Vector3(x,
y, z));
                      //  var textGeo = new
THREE.TextGeometry();
                        var shape = new
THREE.Shape();



//
//                        textGeo.position.x = x;
//                        textGeo.position.y = y;
//                        textGeo.position.z = z;
//
//                       // vertices.push(new
THREE.Vector3(x, y, z));
//                        textGeo.text = bldgId;
```

```
//
//


//
shape.moveTo(bldgData[0].index*3,
bldgData[0].water*20, z);
                        //loop through all points
for each building
                        for (var j = 0; j <
bldgData.length; j++) {
                            x =
bldgData[j].index*3;

//stateXs[bldgStateIndex] + bldgData[j].index;
                            y =
bldgData[j].water*20;

//stateYs[bldgStateIndex] + bldgData[j].water*20;
                            //z = idx;
                            //console.log(z);

lineGeometry.vertices.push(new THREE.Vector3(x, y,
z));

                            shape.lineTo(x, y, z);

                        }

shape.lineTo(bldgData[bldgData.length-1].index*3,
0, z);

shape.lineTo(bldgData[0].index*3, 0, z);


shape.lineTo(bldgData[0].index*3-80, 0, z);

shape.lineTo(bldgData[0].index*3-80, 5, z);

shape.lineTo(bldgData[0].index*3-75, 5, z);

shape.lineTo(bldgData[0].index*3-75, 5, z);

shape.lineTo(bldgData[0].index*3-75, .5, z);

shape.lineTo(bldgData[0].index*3, .5, z);

shape.lineTo(bldgData[0].index*3, 0, z);
```

```
                        var lines = new
THREE.Line(lineGeometry, lineMaterial);
                        scene.add(lines);


                        //scene.add(textGeo);

                        //console.log(lines);


                        var extrudeSettings = {
amount: 2, bevelEnabled: true, bevelSegments: 1,
steps: 1, bevelSize: 1, bevelThickness: .1 };

                        addShape( shape,
extrudeSettings, 0xffffff, 0, 0, z, 0, 0, 0, 1
);//bldgData[0].index*3, bldgData[0].water*20

//
//
//                          var geometry = new
THREE.SphereGeometry( 5, 16,16 );
//                          var material = new
THREE.MeshBasicMaterial( {color: 0xff0000} );
//                          var sphere = new
THREE.Mesh( geometry, material );
//                          sphere.position.x =
bldgData[0].index*3;
//                          sphere.position.y = 0;
//                          sphere.position.z = z;
//                          scene.add( sphere );
//                          objects.push(sphere);
//


                    });
                }
            }



            function showWaterDatabldg() {
                while(scene.children.length > 0){

scene.remove(scene.children[0]);
                }
```

```
                d3.csv("./viz_file_names.csv",
function(error, dataPoints) {
                  //
console.log(dataPoints.columns[0]);
                    waterDataNames =
dataPoints.columns;
                    //console.log(waterDataNames);
                    drawWaterData2(dataPoints);
                })
            }



            function makeMap() {
                var loader = new
THREE.STLLoader();
                //stl 3d model from
https://www.thingiverse.com/thing:1524543
                var counter = 0;
                for ( var i = 0; i <
states.length; i ++ ) {
                    loader.load(
'usmap/'+states[i]+'.stl', function ( geometry ) {
                        var mesh = new THREE.Mesh(
geometry, material );
                        mesh.position.set(
stateXs[counter], stateYs[counter], -50 );

//x:-1200+250*(counter%10)
y:200+180*Math.floor(counter/10)
                        //console.log(i,
mesh.position.y);
                        mesh.rotation.set(
Math.PI/2, 0, Math.PI*2);
                        mesh.scale.set(.6, .6,.6);
                        mesh.castShadow = true;
                        mesh.receiveShadow = true;
                        scene.add(mesh);
                        statesStl.push(mesh);

                        counter ++;
                    } );
                }
            }


            function setClickListeners() {
```

```
//
document.getElementById("resetall").addEventListen
er('click', init);
//
document.getElementById("shot").addEventListener('
click', takeScreenshot);
document.getElementById("camerareset").addEventLis
tener('click', cameraReset);
//
document.getElementById("mapcoordinates").addEvent
Listener('click', mapCoordinates);
//
document.getElementById("resetmap").addEventListen
er('click', resetMap);
//
document.getElementById("editmap").addEventListene
r('click', editMap);
//
document.getElementById("colorbytemp").addEventLis
tener('click', colorByTemp);
//
document.getElementById("showWater").addEventListe
ner('click', showWaterData);
//
//
document.getElementById("showWaterbldg").addEventL
istener('click', showWaterDatabldg);
//

                document.addEventListener(
'mousemove', onDocumentMouseMove, false );



//
document.getElementById("showwdbs").addEventListen
er('click', showWaterDataByState);


//
document.getElementById("showgdbs").addEventListen
er('click', showGasDataByState);
//
document.getElementById("showedbs").addEventListen
er('click', showElecDataByState);
```

```
                //
document.getElementById("showAlldbs").addEventList
ener('click', showAllDataByState);


document.addEventListener('keydown', onkeydown,
false );
//
document.addEventListener('mousedown',
updatemappo, false );


            }

            function initGUI() {
                  var gui = new dat.GUI();
                  gui.add( effectController,
"showDots" ).onChange( function( value ) {
pointCloud.visible = value; } );
                  gui.add( effectController,
"showLines" ).onChange( function( value ) {
linesMesh.visible = value; } );
                  gui.add( effectController,
"minDistance", 10, 300 );
                  gui.add( effectController,
"limitConnections" );
                  gui.add( effectController,
"maxConnections", 0, 30, 1 );
                  gui.add( effectController,
"particleCount", 0, maxParticleCount, 1
).onChange( function( value ) {
                        particleCount = parseInt(
value );
                        particles.setDrawRange( 0,
particleCount );
                  });
             }

             function init() {
               // initGUI();
                 stateXs = xpstn;
                 stateYs = ypstn;

                  container =
document.createElement( 'div' );
                  document.body.appendChild(
container );
```

```
                    camera = new
THREE.PerspectiveCamera( 50, window.innerWidth /
window.innerHeight, 1, 100000 );
                    camera.position.z = 9000;
                camera.position.y = 1800;
                camera.position.x = -500;



                    controls = new
THREE.TrackballControls( camera );
                    controls.rotateSpeed = 1.0;
                    controls.zoomSpeed = 1.2;
                    controls.panSpeed = 0.8;
                    controls.noZoom = false;
                    controls.noPan = false;
                    controls.staticMoving = false;
                    controls.dynamicDampingFactor =
0.3;

                    scene = new THREE.Scene();
                    //scene.background = new
THREE.Color( 0x898989 );
                    scene.add( new
THREE.AmbientLight( 0x505050 ) );
                    var light = new
THREE.SpotLight( 0xB8B8B8, 0.5 );
                    light.position.set( 0, 500,
2000 );
                    light.castShadow = true;
                    light.shadow = new
THREE.LightShadow( new THREE.PerspectiveCamera(
50, 1, 200, 10000 ) );
                    light.shadow.bias = - 0.00022;
                    light.shadow.mapSize.width =
2048;
                    light.shadow.mapSize.height =
2048;
                    scene.add( light );



                ////////////////////make
boxes/////////////////

//                  makeBoxes();

                mouse = new THREE.Vector2();
```

```
                raycaster = new THREE.Raycaster();


                    renderer = new
THREE.WebGLRenderer( { antialias: true, alpha:
true } );
                    renderer.setPixelRatio(
window.devicePixelRatio );
                    renderer.setSize(
window.innerWidth, window.innerHeight );
                    renderer.shadowMap.enabled =
true;
                    renderer.shadowMap.type =
THREE.PCFShadowMap;
                    container.appendChild(
renderer.domElement );




                setClickListeners();

                    var dragControls = new
THREE.DragControls( objects, camera,
renderer.domElement );
                    dragControls.addEventListener(
'dragstart', function ( event ) { controls.enabled
= false; } );
                    dragControls.addEventListener(
'dragend', function ( event ) { controls.enabled =
true; } );




                    var info =
document.createElement( 'div' );
                    info.style.position =
'absolute';
                    info.style.top = '10px';
                    info.style.width = '100%';
                    info.style.textAlign = 'left';
                    info.innerHTML = '<a
href="3dWaterVizfly2.html">View Water Data in Each
Building</a><br><a>W: zoom in</a><br><a>S: zoom
out</a><br><a>A: turn left</a><br><a>D: turn
```

```
right</a><br><a><: move screen to the
left</a><br><a>>: move screen to the
right</a><br><a>Hover on model to
lock;</a><br><a>Click and drag to move
model.</a>';
                    container.appendChild( info );
                    //stats = new Stats();
                    //container.appendChild(
stats.dom );

                    window.addEventListener(
'resize', onWindowResize, false );




                 ////////////map
settings///////////////

                // makeMap();
                 showWaterDatabldg();




              }

              function onWindowResize() {
                    camera.aspect =
window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();
                    renderer.setSize(
window.innerWidth, window.innerHeight );
              }

             function onDocumentMouseMove( event )
{
                    //event.preventDefault();
                    mouse.x = ( event.clientX /
window.innerWidth ) * 2 - 1;
                    mouse.y = - ( event.clientY /
window.innerHeight ) * 2 + 1;
                 //console.log(mouse.x, mouse.y);
              }

              function animate() {
```

```
                    requestAnimationFrame( animate
);
                    render();
                    //stats.update();
               }

               function render() {
                    controls.update();
                  //flyControls.update();
                    renderer.render( scene, camera
);

                  //console.log(scene.children);

                  raycaster.setFromCamera( mouse,
camera );
                    var intersects =
raycaster.intersectObjects( scene.children);
                // console.log(intersects.length);

                    if ( intersects.length > 0 ) {
                     console.log("have
intersects");
                         if ( INTERSECTED !=
intersects[ 0 ].object ) {
                              if ( INTERSECTED )
INTERSECTED.material = material0;
                              INTERSECTED =
intersects[ 0 ].object;
                              INTERSECTED.material
= material01;

//INTERSECTED.material.emissive.setHex( 0xff0000
);
                         console.log("intersected
1");
                         console.log(INTERSECTED);

                         }
                    }

                  else {
                      if ( INTERSECTED )
INTERSECTED.material = material0;

                    INTERSECTED = null;
                    //console.log("intersected
2");
```

```
                    }
                    renderer.render( scene, camera
);



                }

            function takeScreenshot() {
                console.log("taking
screeshot...");
                // open in new window like this
                var w = window.open('', '');
                w.document.title = "Screenshot";

//w.document.body.style.backgroundColor = "red";
                var img = new Image();
                // Without 'preserveDrawingBuffer'
set to true, we must render now
                renderer.render(scene, camera);
                img.src =
renderer.domElement.toDataURL();
                w.document.body.appendChild(img);
                // download file
                var a =
document.createElement('a');
                // Without 'preserveDrawingBuffer'
set to true, we must render now
                renderer.render(scene, camera);
                a.href =
renderer.domElement.toDataURL().replace("image/png
", "image/octet-stream");
                a.download = 'canvas.png';
                a.click();
            }

            function cameraReset() {
                console.log("Reseting camera...");
                //console.log(camera);
                camera.position.x = 0;
                camera.position.y = 0;
                camera.position.z = 1000;
                camera.rotation.x = 0;
                camera.rotation.y = 0;
                camera.rotation.z = 0;
                camera.fov = 50;
                camera.filmOffset = 0;
```

```
                camera.updateProjectionMatrix();
                controls.reset();
                //window.innerWidth /
window.innerHeight, 1, 10000
            }


            function mapCoordinates() {
                //statesStl = [];
                var xs = [];
                var ys = [];
                //console.log(statesStl.length,
buildingCount);
                //locate all state stl files
                for (var i = 0; i <
statesStl.length; i++) {
                    //statesStl.push(objects[i]);

xs.push(statesStl[i].position.x);

ys.push(statesStl[i].position.y);
                    console.log(statesStl.length);

                }
                console.log(statesStl.length);

                const linkx =
document.createElement( 'a' );
                linkx.style.display = 'none';
                document.body.appendChild( linkx
);

                const blobx = new Blob( [ xs ], {
type: 'text/plain' } );
                const objectURLx =
URL.createObjectURL( blobx );

                linkx.href = objectURLx;
                linkx.href = URL.createObjectURL(
blobx );
                linkx.download =
'x_map_setting.json';

                linkx.click();
```

```
                const linky =
document.createElement( 'a' );
                linky.style.display = 'none';
                document.body.appendChild( linky
);

                const bloby = new Blob( [ ys ], {
type: 'text/plain' } );
                const objectURLy =
URL.createObjectURL( bloby );

                linky.href = objectURLy;
                linky.href = URL.createObjectURL(
bloby );
                linky.download =
'y_map_setting.json';

                linky.click();




//https://discourse.threejs.org/t/how-to-create-a-
new-file-and-save-it-with-arraybuffer-content/628/
3

            }

            const zzz = -100;

            function resetMap() {
                var counter = 0;
                for (var i = 0; i <
statesStl.length; i++) {
                    statesStl[i].position.x =
xpstn[counter];
                    statesStl[i].position.y =
ypstn[counter];
                    statesStl[i].position.z =
zzz;//-100;
                    statesStl[i].material =
material;

                    counter++;
                }

            }
```

```
            function colorByTemp() {
                var material2 = new
THREE.MeshPhongMaterial( { color: 0xffffff,
specular: 0x111111, shininess: 20 } );
                var counter = 0;
                for (var i = 0; i <
statesStl.length; i++) {
                    statesStl[i].material =
material2;
                    counter++;
                }
            }

            function editMap() {
                if (mapEditMode == false) {
                    mapEditMode = true;
                    var dragControlsMap = new
THREE.DragControls( statesStl, camera,
renderer.domElement );

dragControlsMap.addEventListener( 'dragstart',
function ( event ) { controls.enabled = false; }
);

dragControlsMap.addEventListener( 'dragend',
function ( event ) { controls.enabled = true; } );
                }
                else if (mapEditMode == true){
                    mapEditMode = false;

                    //controls.enabled = true;
                   // dragControlsMap.abort;

                    //TODO
                }

                console.log("edit mode:",
mapEditMode);

            }

            var mode;
            var MODE = { TRACKBALL: 0, FLY: 1 };

            function onkeydown() {
                var keyCode = event.which;
```

```
                    if (keyCode == 87) {
                        console.log("up");
                        //camera.position.y += 10;
                        //camera.zoom += 2;
                        camera.fov -= 3;
                    } else if (keyCode == 83) {
                        console.log("down");
                        camera.fov += 3;
                    } else if (keyCode == 65) {
                        camera.position.x -= 10;
                    } else if (keyCode == 68) {
                        camera.position.x += 10;
                    } else if (keyCode == 32) {
                        cameraReset();
                   } else if (keyCode == 188) {
                       camera.filmOffset -= 1;
                    } else if (keyCode == 190) {
                       camera.filmOffset += 1;
                    }
                    camera.updateProjectionMatrix();


                }


                //https://jsfiddle.net/2pha/art388yv/
                //chrome://flags/#enable-webvr
                            //VRViewer({THREE});
//python -m SimpleHTTPServer
                //localhost:8000
            </script>

       </body>
</html>
```