

Fall 2012

# The Visualization of Historical Structures and Data in a 3D Virtual City

Bryant Panyarachun  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)

Part of the [Computer Sciences Commons](#)

---

## Recommended Citation

Panyarachun, Bryant, "The Visualization of Historical Structures and Data in a 3D Virtual City" (2012). *Master's Projects*. 261.

DOI: <https://doi.org/10.31979/etd.7z25-efvx>

[https://scholarworks.sjsu.edu/etd\\_projects/261](https://scholarworks.sjsu.edu/etd_projects/261)

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

**The Visualization of Historical Structures and Data in a 3D Virtual City**

**A Project Report**

**Presented to**

**The Faculty of the Department of Computer Science**

**San José State University**

**In Partial Fulfillment**

**of the Requirements for the Degree**

**Master of Science in Computer Science**

**by**

**Bryant Panyarachun**

**September 2012**

© 2012

Bryant Panyarachun

ALL RIGHTS RESERVED

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled  
The Visualization of Historical Structures and Data in a 3D Virtual City

by  
Bryant Panyarachun

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE  
SAN JOSÉ STATE UNIVERSITY

September 2012

-----  
Dr. Soon Tee Teoh, Department of Computer Science

Date

-----  
Dr. Chris Pollett, Department of Computer Science

Date

-----  
Prof. Debra Caires, Department of Computer Science

Date

APPROVED FOR THE UNIVERSITY

-----  
Associate Dean Office of Graduate Studies and Research

Date

## **Abstract**

The Visualization of Historical Structures and Data in a 3D Virtual City

by Bryant Panyarachun

Google Earth is a powerful tool that allows users to navigate through 3D representations of many cities and places all over the world. Google Earth has a huge collection of 3D models and it only continues to grow as users all over the world continue to contribute new models. As new buildings are built new models are also created. But what happens when a new building replaces another? The same thing that happens in reality also happens in Google Earth. Old models are replaced with new models. While Google Earth shows the most current data, many users would also benefit from being able to view historical data. Google Earth has acknowledged this with the ability to view historical images with the manipulation of a time slider. However, this feature does not apply to 3D models of buildings, which remain in the environment even when viewing a time before their existence. I would like to build upon this concept by proposing a system that stores 3D models of historical buildings that have been demolished and replaced by new developments. People may want to view the old cities that they grew up in which have undergone huge developments over the years. Old neighborhoods may be completely transformed with new road and buildings. In addition to being able to view historical buildings, users may want to view statistics of a given area. Users can view such data in their raw format but using 3D visualizations of

statistical data allows for a greater understanding and appreciation of historical changes. I propose to enhance the visualization of the 3D world by allowing users to graphically view statistical data such as population, ethnic groups, education, crime, and income. With this feature users will not only be able to see physical changes in the environment, but also statistical changes over time.

## **Acknowledgements**

First and foremost, I would like to thank Dr. Soon Tee Teoh for all of his help and guidance throughout the past year. I would like to thank Dr. Chris Pollet and Debra Caires for being a part of my committee and giving their advice and support. I would also like to thank my parents who have always believed in my abilities and have continuously supported me.

## Table of Contents

1. Introduction.....	8
2. Related Work.....	9
2.1. GRASS GIS.....	9
2.2. Google Earth Pro .....	10
2.3. Comparison to GE Collab.....	10
3. Design and Implementation.....	11
3.1. Initial Experimentation.....	12
3.2. Models .....	13
3.3. Visualization of data.....	14
3.4. Interface.....	16
3.4.1.Application Environment.....	17
3.4.2.Google Earth Plugin .....	18
3.4.3.MySQL Database.....	19
3.5. Functionality.....	20
3.5.1.Importing a model.....	21
3.5.2.Generating Heatmaps.....	23
3.5.3.Creating 3D Data Visualizations .....	25
3.5.4.Representing data with polygon heights.....	26
3.5.5.Creating 3D heatmaps.....	30
3.6. Additional Features .....	33
3.6.1.Persistance between sessions.....	33
3.6.2.Delete uploaded features.....	34
3.6.3.Hide features.....	36
4. Conclusion .....	37
5. Future work.....	37
6. References .....	39
7. Figures and Equations.....	40
7.1. Figures.....	40
7.2. Equations.....	40



## 1. Introduction

The Internet has made it possible to easily access all types of information and resources with the click of a button. Nearly any type of information or resource can be found or accessed online. One type of information that is not easily found is records regarding past land developments, structures, buildings, and more specifically, visual records. One may be able to find historical photos online about a well known or famous structure or area such as the White House or the city of Manhattan, but what about lesser known places such as Salinas, CA or even San Jose State University. To find historical data or images one would most likely have to visit the local authority archives or the main city library to view some old archives and maps. A tool that gives the ability to easily view past city developments or structures would be an invaluable asset to those in search of this type of information. Statistical information about an area may be much easier to obtain but interpreting large amounts of raw data can be extremely difficult and confusing. Visualizing raw data can make large amounts of data easier to understand and analyze. Trends, relationships, and variations in data can be easily seen through visualizations. Mapping technology has become increasingly advanced and there are a number of map software and applications but none of which give the ability to easily view or create historical data. The purpose of my project was to create a tool that can be used to visualize historical structures, developments, and data through time. As a result I have developed a web application that allows users to import models and assign them

a timespan in order to represent the period for which a model would have existed. Users can also represent 2D statistical data in the form of heatmaps or 3D polygons. While there exists tools that can be used to visualize 2D data, what most tools lack is the ability to visualize 3D data. An example of 3D data would be prices for individual condominiums that are all located in the same building. Each condo resides on its own floor of the building. The application I have developed has the ability to display the individual floor prices of the building. This function can also be used to visualize price changes over time with the use of the timespan functionality. Since the application I have developed is a web-based application, it is easily used as a collaboration or sharing tool with others. The name of my application is “GE Collab” which is short for Google Earth Collaboration since it utilizes the Google Earth Plugin. Another advantage of being a web application is that it takes advantage of many of the features of Google Earth without the need to download or install any software. Not only does GE Collab make use of existing Google Earth features, it extends current features to create new functionality and uses.

## **2. Related Work**

### **2.1 GRASS GIS**

GRASS GIS is a Geographic Information System used for geospatial data management and analysis, image processing, graphics/maps production, spatial modeling, and visualization. GRASS is currently used in academic and

commercial settings around the world, as well as by many governmental agencies and environmental consulting companies [1]. RASS is an open source desktop application that is written in C and supports multiple operating systems and architectures. GRASS contains over 350 programs and tools to render and manipulate data [1].

## **2.2 Google Earth Pro**

Google Earth Pro is based on the free version of the Google Earth desktop application with additional tools available to the user [3]. Some of these tools include: advanced measurements, high-resolution printing, exclusive pro data layers, spreadsheet importing, and more [2]. Google Earth has a large database of 3D models, which is mostly made from contributions from the online community.

## **2.3 Comparison to GE Collab**

Although GE Collab is still very underdeveloped and lacks the variety of tools that GRASS GIS and Google Earth Pro have, the main features that set it apart is its ability to create 3D heatmaps as well as being a web-based application. GRASS GIS is a very powerful tool but can be extremely confusing due to the number of 3<sup>rd</sup> party modules and add-ons that are available. GRASS GIS also requires the user to be familiar using their system terminal/console. In contrast, Google Earth Pro is very user friendly and does not have plugins or add-ons for additional functionality. It supports 3D model manipulation and

sharing through Google 3D Warehouse. Both GRASS GIS and Google Earth Pro are desktop applications, which require installation on the users system. I kept these applications in mind in order to create a simple, useful, and collaborative friendly application.

### **3. Design and Implementation**

At the beginning of my project what I had in mind was to make use of an existing software program and to implement my desired features by making use of the software's application programming interface (API). The two main software applications that were considered were NASA World Wind and Google Earth.

NASA World Wind is opensource software developed by NASA and the open source community. Older versions of World Wind were developed for Windows using the .NET framework but the newest version of World Wind was developed in Java and is known as World Wind Java. World Wind by itself is simple and does not have many functions. Additional functionality can be added with the use of addons and plugins.

Google Earth is a virtual globe program that was initially created by Keyhole, Inc. and later on, acquired by Google [4]. Unlike NASA World Wind, Google Earth has a large selection of features and functions built into the software and ready to use right after installation. KML (keyhole markup language) files are used to load external models and features that are not included in the Google Earth installation. KML is similar to XML (extensible

markup language) and is used primarily for geographical data to be used in Internet based maps and virtual globe programs. One downside of Google Earth is that it is closed source software and therefore, the source code is not accessible to the public. Both programs have a significant amount of support through public forums, blogs, and articles. Google Earth turned out to have a bigger following and as a result more resources available to users looking to develop on Google Earth. NASA World Wind has a lot of available resources but many articles and references tend to be outdated due to its decrease in popularity.

### **3.1 Initial Experimentation**

I chose to use Google Earth as the platform to develop my initial features due to its already existing features and ease of use. Google Earth allows users to import externally created models in the COLLADA format. COLLADA defines an XML Namespace and database schema to make it easy to transport 3D assets between applications without loss of information [5]. COLLADA files are XML files that end with a .dae (digital asset exchange) file extension. Google Earth also allows for the viewing of historical imagery with the use of a time slider bar. When historical imagery is enabled, if images are available, a user can view images of a given geography at certain time periods by manipulating the time slider bar as shown in figure 1.



Figure 1 – Time Slider Bar

Google earth also gives users the ability to draw lines and create shapes in the 3D globe environment. I made use of these three main features to implement the model and data representation aspects of GE Collab.

### 3.2 Models

I used the Google Earth software to create my own fictional city using 3D models in order to test if the current functionality could be extended and used for my needs. I obtained various models from other cities in Google Earth to use for my city. I also used various online sources such as Google 3D Warehouse [6] for models. While I was able to create a fictional city by importing models, I found that the timespan could not be applied to the models in a way that would make use of the time slider. I soon found that I could make use of the time slider in Google Earth by creating a KML file specifying the timespan of the model by encapsulating it within placemark tags. The placemark tag is the main container for all of the models properties and attributes. Attributes such as name, description, and balloon style can be set in the within the placemark container. The timespan tag allows for the introduction of a beginning and ending time period for an object or image.

### 3.3 Visualization of data

There are a few different ways to visualize data in Google Earth. Polygons can be created in Google Earth and different properties can be used to demonstrate data values such as size, color, or intensity. A couple examples of this can be seen below.

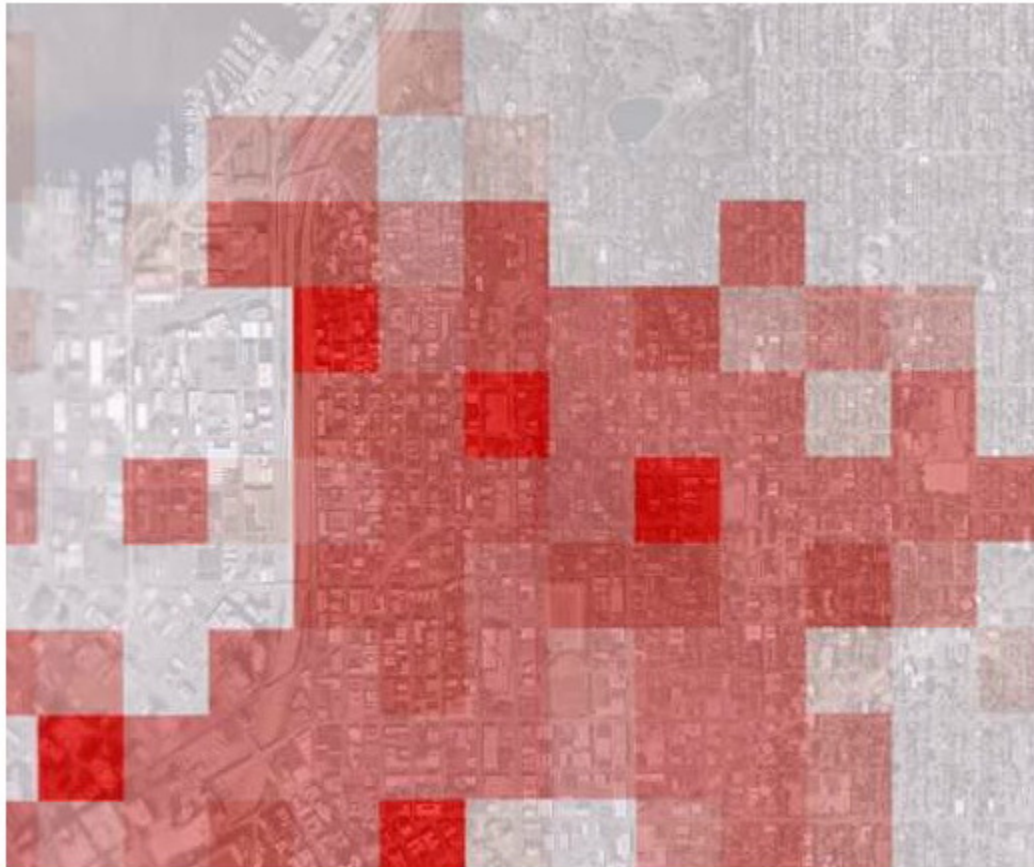


Figure 2 - Varying color densities of the block are used to represent different values in the data [7].



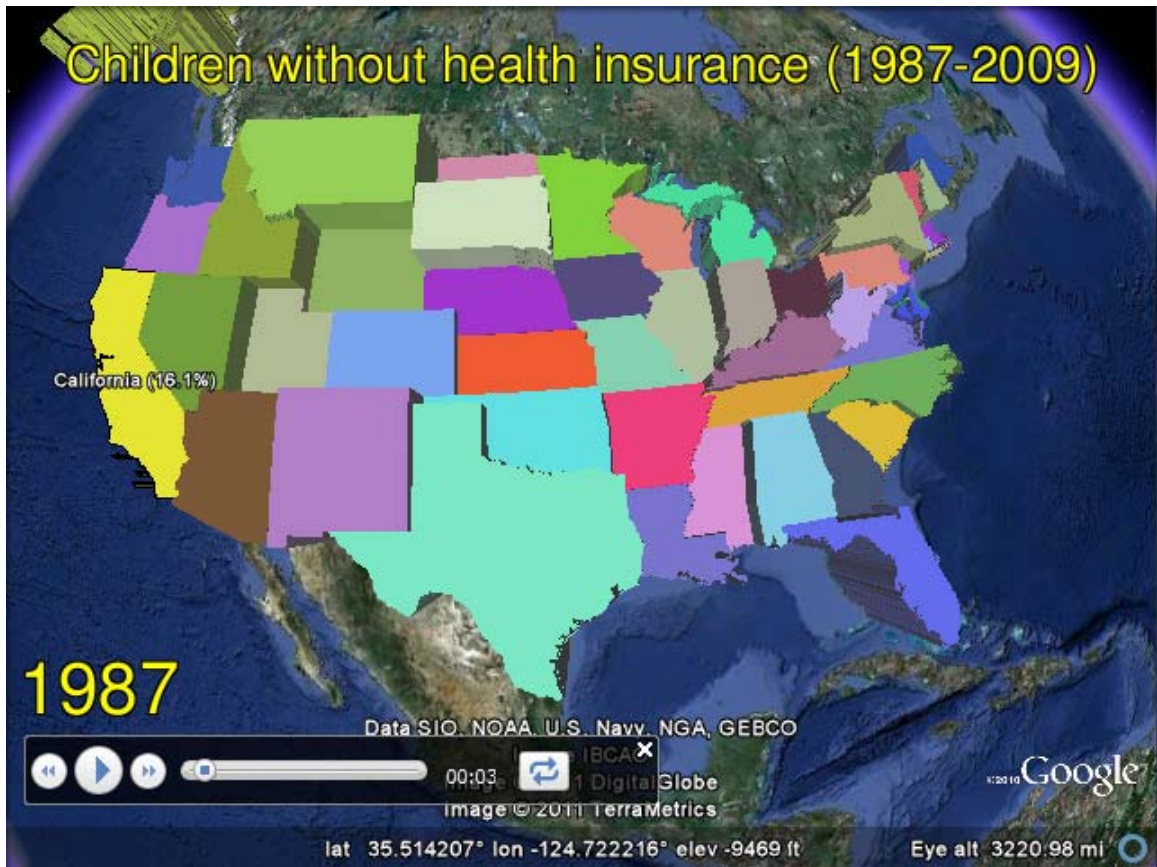


Figure 3 - Variations in data values are represented by varying the height of the polygons [8].

While these methods of visualization may be sufficient, I wanted to explore an alternate way of visualizing data, which is with the use of heatmaps.

Google Earth does not have built in functionality to generate heatmaps so they must be generated externally and placed into Google Earth using an image overlay. Gheat is an API which has been developed to generate heatmaps for use with Google Maps. Heatmap.py is a Python API that can be used to create heatmaps using Python. The technique and color scheme is based on the implementation used in gheat. This API generates a heatmap image describing



the density of a list of given coordinates. Creating heatmaps using images allows them to be placed into Google Earth using the same method as placing an image overlay. It then can be used to create a KML file with the heatmap image to be used with Google Earth. I have written a python script that uses the gheat API to create heatmaps that I can place in Google Earth. Once the KML file is created it must then be imported manually into Google Earth.

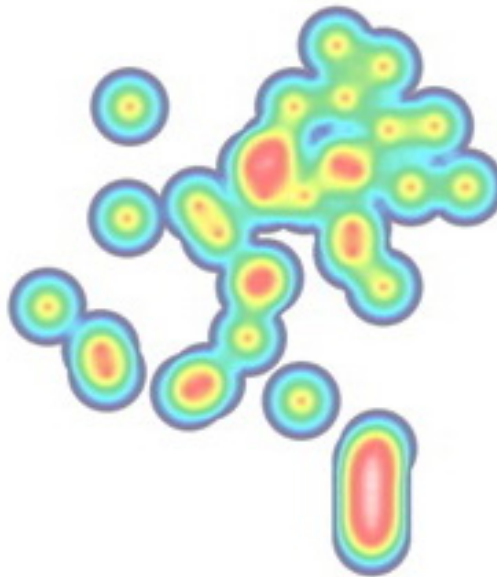


Figure 4 –Example of a generated heatmap image

### 3.4 Interface

After the implementation of the 3D city and heatmaps in using the Google Earth software I came to realize that the process for making use of these functions were very difficult and time consuming due to the need for manually

creating KML files and importing them into Google Earth. I wanted an end-to-end process that would incorporate the functionality that I wanted with Google Earth. Since the Google Earth software is closed source, I do not have the ability to add this functionality directly to Google Earth. However, Google Earth has a browser plugin available that allows users to incorporate the Google Earth 3D globe into a website. The plugin only provides a view of the globe a plugin API is available for users to add any additional functionality in order to manipulate the globe. I decided to make use of the Google Earth Plugin to implement a web-based application that supported my desired functionality. This is what would be known as GE Collab. The Google Earth Plugin can be manipulated using its JavaScript API. The challenge behind this approach was that I now needed to recreate the necessary Google Earth software interface and controls in my web application since all that is available with the initial plugin is the view of the Earth and the ability to move within the Earth view. A benefit of this approach was that GE Collab could be used as a collaborative tool and could be easily accessible without the need for downloading or installing any additional software.

### **3.4.1 Application Environment**

During the initial stages of the web application development, GE Collab was hosted using an online hosting service. As development progressed it became apparent that I would require GE Collab to be running on its own system since I relied on external python libraries such as heatmap.py and the Python

Imaging Library which I was not able to install on my online hosting service's system. As a result, I created my own hosting environment using a Linux machine and setting up an Apache server. The server would allow me to have complete access to the system in order to install any additional libraries that I required. In addition to Python 2.6 and Python Imaging Library, I installed MySQL database and PHP 5.3.2 all of which play key roles in the functionality of the web application and will be discussed in further detail later in the paper.

### **3.4.2 Google Earth Plugin**

The Google Earth Plugin is accessed using its JavaScript API. Creating a plugin instance allows me to manipulate and access the Earth view. With the plugin instance I can create various feature instances such as placemarks, models, icons, ground overlays, balloons, and many other similar KML features. KML features are children of the KML object and its siblings can be seen in figure 5.

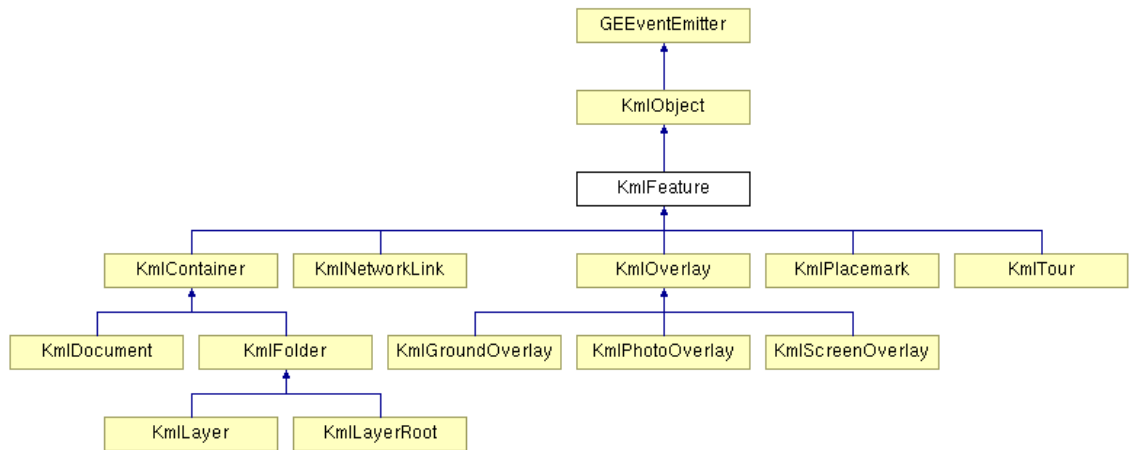


Figure 5 – KML Object inheritance tree [9]

### 3.4.3 MySQL Database

The MySQL database is used to store data regarding any models, heatmaps, or data that a user has imported into the application. The database contains tables for each type of feature: models, heatmaps, and 3D data. Each of these tables contains the name of the feature, the path to the files that have been stored on the server, and the beginning and ending timespan for the feature. The entries in the database are shown on the right hand side panel of GE Collab.

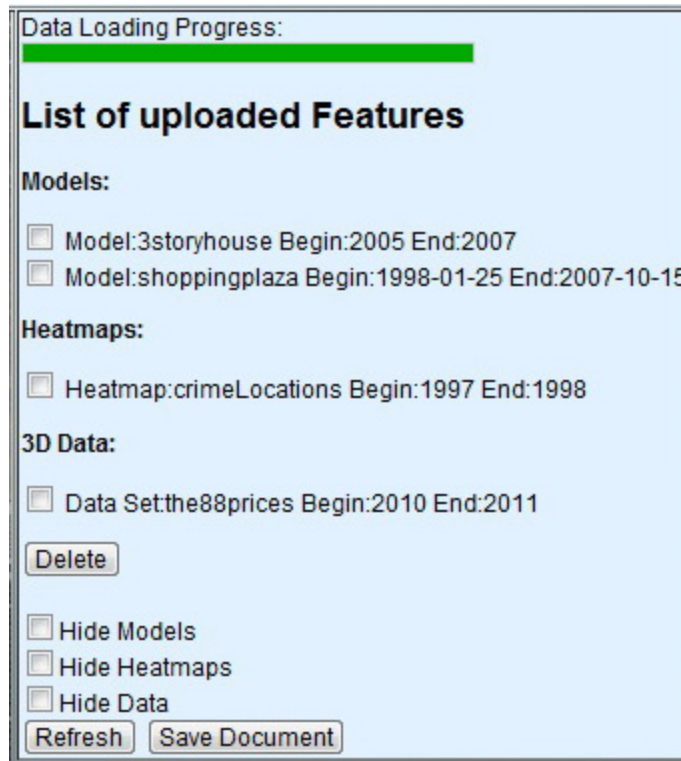
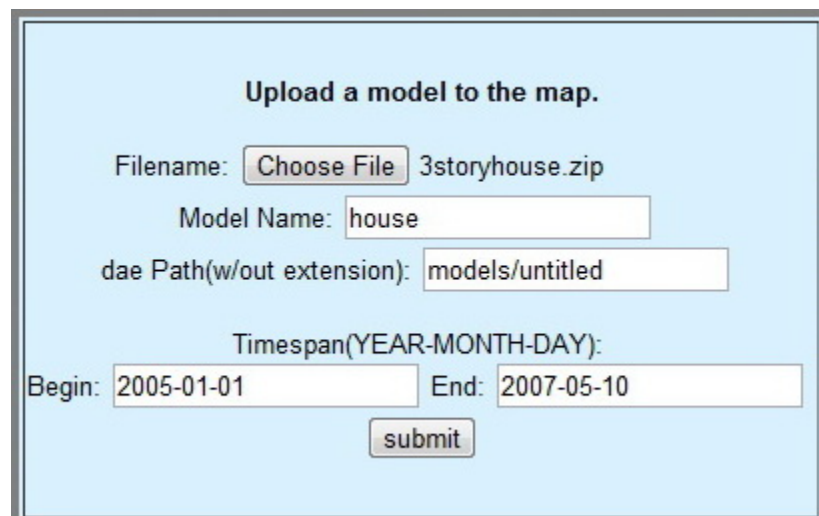


Figure 6 –List of Uploaded Features Panel

### 3.5 Functionality

There are three main functions implemented in GE Collab. The ability to import models, import data to create heatmaps, and import data to create 3D visualizations.

### 3.5.1 Importing a model



**Upload a model to the map.**

Filename:  3storyhouse.zip

Model Name:

dae Path(w/out extension):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

Figure 7 – Upload a model form

A user can import 3D models that are in the COLLADA format. The COLLADA file describes the structure of the model and references to external image files to build the model. Users are required to package their COLLADA model in a zip file containing the .dae file and the image artifacts. Users must also specify the directory path to the .dae file since there are no restrictions on how the files are packaged within the zip file. A unique name for the model must be specified in order for the upload to be successful. A beginning and end timespan can be specified for the model as well. If no values are specified, then the manipulation of the time slider bar will have no affect on the uploaded model when it has been added to the globe. The timespan is in the format of year-month-day. Valid timespan values can be general or specific in that only a year can be entered, a year and month, or a year, month, and day. Once the form is submitted, the values in the form are posted to a PHP script that runs on the

server. The script unzips the file specified by the user and stores the contents on the server. A connection to the MySQL database is made and the values in the form are inserted into the database. Once the PHP script completes the model is added to the plugin instance. A placemark is created as well as a model which references to the model that has just been uploaded. The placemark's geometry is set to the models geometry in order to correspond the placemark with the model. Various attributes such as altitude, tilt, position, heading, and orientation are set during this time. The model is then added to the globe in the center of the current view. The newly added model will appear in the list of uploaded features panel which is done by querying the database after a feature has been uploaded. The model's coordinate location can be changed by holding the alt key and clicking on the desired location on the globe. The model can be rotated in the x, y, and z-axis by using the control buttons in the GE Collab interface as shown in figure 8.

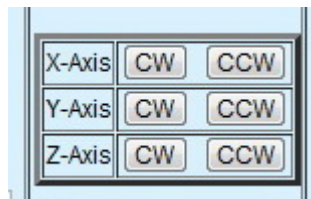


Figure 8 – Model position controls which change the heading, tilt, and roll of the model.

### 3.5.2 Generating Heatmaps

To create a heatmap in GE Collab, a user must import a text file containing the data coordinates. The heatmaps are created based on the density of a given list of coordinates. The input file must be in the format of list of comma separated coordinate tuples such as is figure 9.

```
(-120.426689, 36.894435),  
(-120.407265, 36.880195),  
(-120.411507,36.884977),  
(-120.414530,36.885022),  
(-120.416779,36.886493),  
(-120.417913, 36.886742),  
(-120.415633,36.886493),  
(-120.418151, 36.887857),  
(-120.419806,36.885109),  
(-120.417341,36.884978),  
(-120.422612, 36.887707),  
(-120.418, 36.88588),...,  
(longitude, latitude)
```

Figure 9 – Format for input file to generate a heatmap

**Create a heatmap by uploading a file.**

Filename:  file.txt

Heatmap Name:

dot size (px):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

Figure 10 – Import heatmap data form



A unique name must be entered for the heatmap. The dotsize field specifies the size of a single point in pixels. If the data spans a small area then a user may want to use a smaller dotsize to prevent the points from being too large that they combine with each other. On the other hand, if the data spans a large area then a user may want to use a large dotsize so that the data can be viewable when viewing from a far distance/altitude. The beginning and ending timespan can be specified for the heatmap to be created. With the timespan functionality, users can add multiple heatmaps to represent different datasets during different periods of time. Once the form is submitted, the form values are submitted to a cgi-python script that parses the input file and generates the heatmap image and a KML file to be read is imported into the Google Earth plugin. The rest of the form values are stored into the MySQL database. The KML file describes the position of the heatmap, which is represented as a ground overlay. The KML is parsed by GE Collab and is added to the globe. The list of uploaded features is updated appropriately.

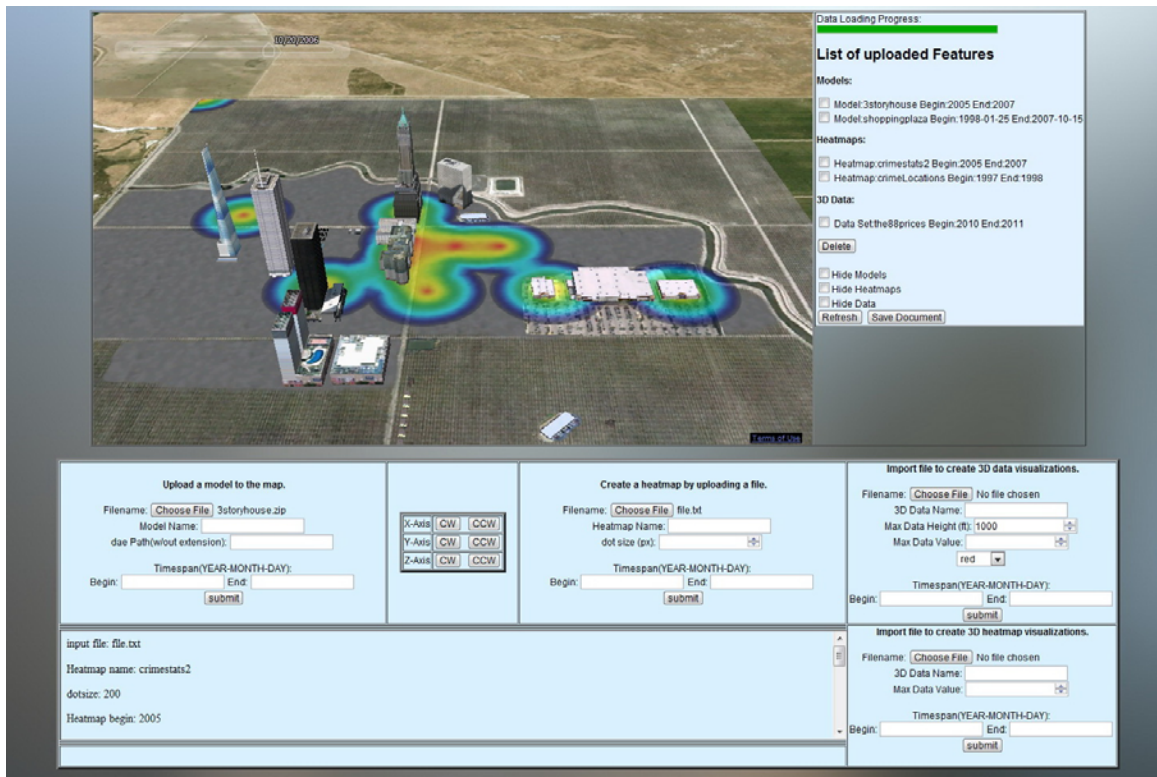


Figure 11 – Heatmap data imported into GE Collab

### 3.5.3 Creating 3D Data Visualizations

There are two different types of 3D visualizations implemented. The first type represents data values using polygon heights. The second type allows users to create in a sense 3D heatmaps that can represent different data values within a building or structure.

### 3.5.4 Representing data with polygon heights

Import file to create 3D data visualizations.

Filename:  testData.txt

3D Data Name:

Max Data Height (ft):

Max Data Value:

Timespan(YEAR-MONTH-DAY):

Begin:  End:

Figure 12 – Import 3D data form

To create 3D data visualizations, the user must import a file containing the data in the format shown in figure 13:

```
(latitude,longitude),  
...,  
(latitude,longitude),  
Data Value;
```

Sample data is shown below

```
(36.8767, -120.4114),  
(36.8755, -120.4148),  
(36.8774, -120.4154),  
(36.8775, -120.4125),  
700;  
(36.8882, -120.4164),  
(36.8880, -120.4166),  
(36.8883,-120.4174),  
(36.8886, -120.4171),  
50;  
(36.882572174072266,-120.41795349121094),  
(36.88255310058594, -120.41897583007812),  
(36.88350296020508, -120.41897583007812),  
(36.883667,-120.418503),  
(36.88351821899414,-120.4179458618164),  
100;
```

Figure 13 – Input file format and sample data for importing a 3D data file.

The user can add as many coordinate points as needed to represent the desired shape. The coordinate points represent the perimeter of the shape and must be listed in the order in which to draw the shape. In the 3D data form, the user can specify the maximum height to represent the greatest value of the data to be imported. The maximum data height is represented in feet from the ground. The default maximum data height value is 1000 feet. A user can also specify the maximum value of data to correspond with the maximum data height. If a value is not specified, then the maximum value found in the file is used. A user may want to specify the maximum data value if importing multiple files that need to be normalized to the same scale. If a user imports multiple files that have different maximum data values and does not specify it in the form, then the representation of the maximum data height will be inconsistent between the sets of data. There are 4 color choices that can be chosen from to draw the data: red, green, blue, and yellow. The color red is selected by default.

**Data Loading Progress:**

**List of uploaded Features**

**Models:**

- Model:3storeyhouse Begin:2005 End:2007
- Model:shoppingplaza Begin:1998-01-25 End:2007-10-15

**Heatmaps:**

- Heatmap:crimestats2 Begin:2005 End:2007
- Heatmap:crimeLocations Begin:1997 End:1998

**3D Data:**

- Data Set:data1 Begin:2001 End:2002
- Data Set:data2 Begin:2002 End:2003
- Data Set:the88prices Begin:2010 End:2011

Delete

Hide Models

Hide Heatmaps

Hide Data

**Upload a model to the map.**

Filename:  No file chosen

Model Name:

dae Path(w/out extension):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Create a heatmap by uploading a file.**

Filename:  No file chosen

Heatmap Name:

dot size (px):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Import file to create 3D data visualizations.**

Filename:  No file chosen

3D Data Name:

Max Data Height (ft):

Max Data Value:

red

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Import file to create 3D heatmap visualizations.**

Filename:  No file chosen

3D Data Name:

Max Data Value:

Timespan(YEAR-MONTH-DAY):

Begin:  End:

Upload: testData2.txt  
 Type: text/plain  
 Size: 0.5078125 Kb  
 Temporarily stored in: /tmp-phpC40Hva  
 Data Name is: data2  
 Connected successfully  
 data1 upload data1.txt 2001 2002  
 data2 upload data2.txt 2002 2003

**Data Loading Progress:**

**List of uploaded Features**

**Models:**

- Model:3storeyhouse Begin:2005 End:2007
- Model:shoppingplaza Begin:1998-01-25 End:2007-10-15

**Heatmaps:**

- Heatmap:crimestats2 Begin:2005 End:2007
- Heatmap:crimeLocations Begin:1997 End:1998

**3D Data:**

- Data Set:data1 Begin:2001 End:2002
- Data Set:data2 Begin:2002 End:2003
- Data Set:the88prices Begin:2010 End:2011

Delete

Hide Models

Hide Heatmaps

Hide Data

**Upload a model to the map.**

Filename:  No file chosen

Model Name:

dae Path(w/out extension):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Create a heatmap by uploading a file.**

Filename:  No file chosen

Heatmap Name:

dot size (px):

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Import file to create 3D data visualizations.**

Filename:  No file chosen

3D Data Name:

Max Data Height (ft):

Max Data Value:

red

Timespan(YEAR-MONTH-DAY):

Begin:  End:

**Import file to create 3D heatmap visualizations.**

Filename:  No file chosen

3D Data Name:

Max Data Value:

Timespan(YEAR-MONTH-DAY):

Begin:  End:

Upload: testData2.txt  
 Type: text/plain  
 Size: 0.5078125 Kb  
 Temporarily stored in: /tmp-phpC40Hva  
 Data Name is: data2  
 Connected successfully  
 data1 upload data1.txt 2001 2002  
 data2 upload data2.txt 2002 2003

Figure 14 – Examples of imported 3D data

Similar to the other functions, the beginning and end timespan for the data can be specified as well. When the form is submitted, the form values are posted to a PHP script. The script writes the imported file to the server with the corresponding data name specified in the form. The values such as file name, file path, beginning and ending timespan are then inserted into the MySQL database. In the web application, the file is retrieved using a synchronous AJAX request. The file is then parsed, separating the data into its individual buildings/shapes. For each shape, a polygon is created and the coordinate points are parsed to create the shape of the polygon. The height of the polygon is determined by normalizing the height by using the maximum data value and the specified maximum data height. The formula used for normalization is shown in equation 1.

$$normalizedheight = \frac{datavalue - mindatavalue}{greatestdatavalue - mindatavalue} \times maxdataheight$$

Equation 1 - Currently 0 is used for the minimum data value in to represent a height value of 0. As a result, the application does not support negative data values.

Initially the polygon is created as a flat shape and then extruded upwards using the normalized height value relative to the ground. Other attributes such as face color and line width are set and the polygon is appended to the placemarks geometry. A single placemark is created which contains each of the children

polygons. If available, the beginning and ending timespans are set for the placemark and the placemark is added to the Google Earth Plugin instance. The list of uploaded features panel is then updated to show the newly updated data.

### 3.5.5 Creating 3D heatmaps

Similarly to the previously discussed functionality, creating a 3D heatmap requires the input of a text file describing the polygon shapes and the value associated with only this functionality includes the ability to represent values within the polygon itself. The input file must follow the format shown below:

(latitude, longitude), (latitude, longitude), ..., n, height=float, floors=float, float, float, float, float, float, float, ..., k;

```
(36.876708984375, -120.41138458251953),(36.87548065185547, -  
120.41477966308594),(36.877384185791016, -  
120.41539764404297),(36.87754440307617,-  
120.41249084472656),height=700,floors=200000,600000,750000,90000,150000  
,1500000,900000;  
(36.8882, -120.4164),(36.8880, -120.4166),(36.8883,-120.4174),(36.8886, -  
120.4171),height=50,floors=20000,30000,10000;  
(36.882572174072266,-120.41795349121094),(36.88255310058594, -  
120.41897583007812),(36.88350296020508, -  
120.41897583007812),(36.883667,-120.418503), (36.88351821899414,-  
120.4179458618164), height=100,floors=20000,30000,10000,5000;
```

Figure 15 – Format and sample data for importing 3D heatmap data

The coordinate points define the perimeter of the building/polygon and must be listed in the order in which to draw the outer edges of the polygon. The

height of the polygon must be in feet relative to the ground. The value of each floor will be represented with a certain color and the number of floors defined in the input file will be used to determine the number of floors to display on the created polygon. The imported file is stored on the server using the specified data name. The data set name and the additional information such as the path to the imported file and beginning and ending timespan are inserted into its respective table in the MySQL database. The data values can be represented with seven different colors: red, orange, yellow, green, light blue, dark blue, black; which represent the data values from greatest to least. To determine the color of each data value, the range from the minimum data value, which is 0, to the maximum data value is divided between the seven colors which results in a data value range for each color. Again, the file is accessed with a synchronous AJAX request and the file is parsed and split by each building. There are various string manipulations done in order to correct any spacing and letter case inconsistencies in the data. Initially, I thought I could represent the different floors of a building by stacking polygons on top of each other to make the building. However, the Google Earth Plugin has very limited functionality in regards to drawing polygons which made the implementation of this feature much more complex. The Google Earth Plugin does not have the ability to draw 3D polygons floating above the ground. Polygons can only be drawn on the surface of the earth and extruded upwards. I was able to overcome this limitation and display the floors of each building by representing a building with multiple polygons



within each other. The height of each floor is determined by dividing the height of the building with the number of floors. The bottom floor of the building is drawn using the original coordinates specified in the input file. For each successive floor, a new set of coordinates is calculated resulting in a polygon that is slightly smaller and fits within the previous polygon. The height of each successive floor is determined by multiplying the floor height with the floor number. As a result, only the top of a polygon is shown, representing the floor, while the rest of the polygon is hidden behind the floors below. Calculating the new coordinates is done by comparing the current coordinate values  $n$  with the  $n+2$  coordinate values. If the latitude of the  $n+2$  coordinate is greater than the latitude of  $n$ , the new coordinate value will be incremented by  $(0.000001 * \text{the current floor level})$ . If the latitude of the  $n+2$  coordinate is less than the latitude of  $n$ , then the new coordinate value will be decremented by  $(0.000001 * \text{the current floor level})$ . The same comparison is done for longitude as well. For each floor, new coordinates are calculated and a new polygon is created and appended to the features of the Google Earth Plugin. Each polygon is represented with its own placemark since a placemark is restricted to having a single color. If defined, the beginning and ending timespan values are set for each building and the list of uploaded features panel is updated accordingly to display the imported data.

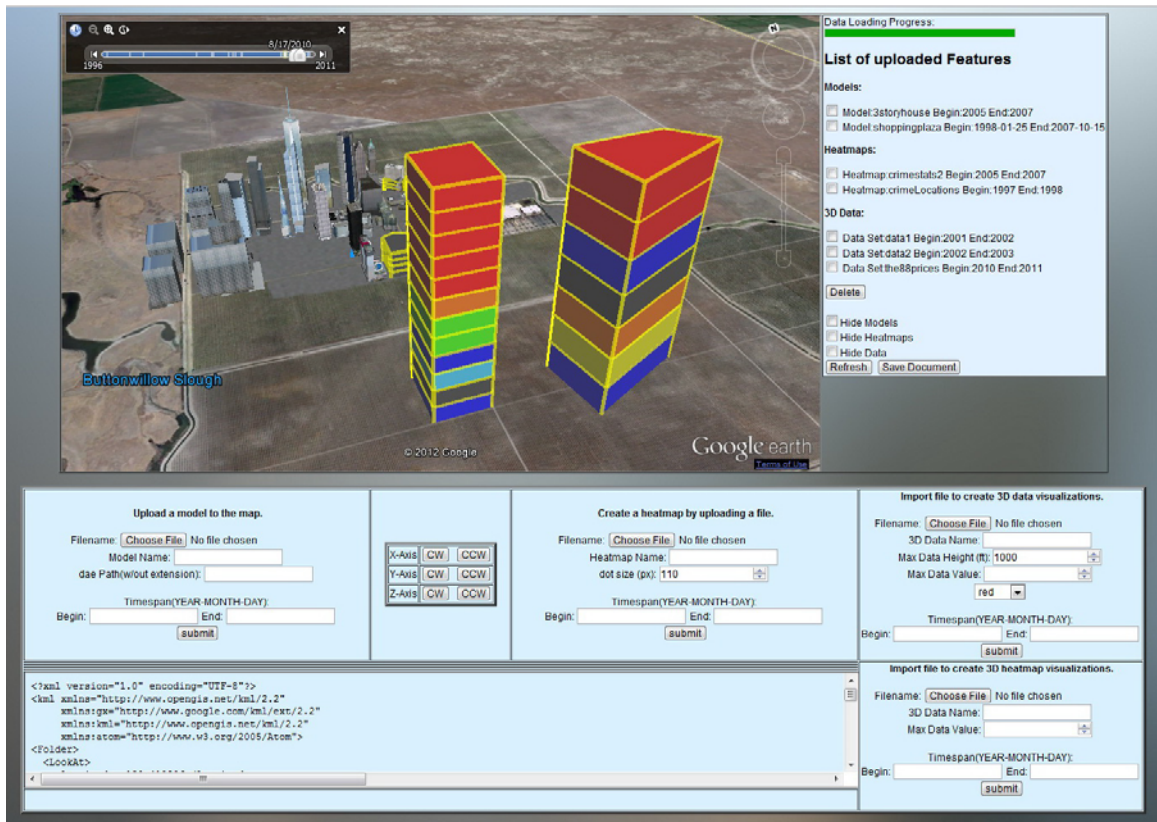


Figure 16 – 3D heatmap data imported into GE Collab

### 3.6 Additional Features

#### 3.6.1 Persistence between sessions

The Google Earth Plugin has no form of persistence. Whenever the page/plugin is refreshed, all features that have been loaded into the plugin instance are lost. In order to create persistence for the web application I implemented a function which would save all imported features to KML documents which are loaded upon the initialization of the Google Earth Plugin. The Google Earth Plugin API has available a function to get the KML

representation of a plugin feature. For each feature that has been loaded, a KML document is written to the server. When the Google Earth Plugin is re-initialized, an asynchronous AJAX request calls a PHP script which returns a list of KML files to be loaded. Each file is read and the features are loaded into the plugin just as they were originally.

### **3.6.2 Delete uploaded features**

The list of uploaded features panel shows all features uploaded and organized by feature type; Models, Heatmaps, 3D Data. Each feature is listed with a checkbox which is used to select the feature for deletion. This is done by submitting an XMLHttpRequest to a PHP script which queries the MySQL database for all features that have been uploaded. The response is HTML formatted checkboxes which are then inserted into the web application document using the innerHTML function. A single or multiple items can be selected for deletion.

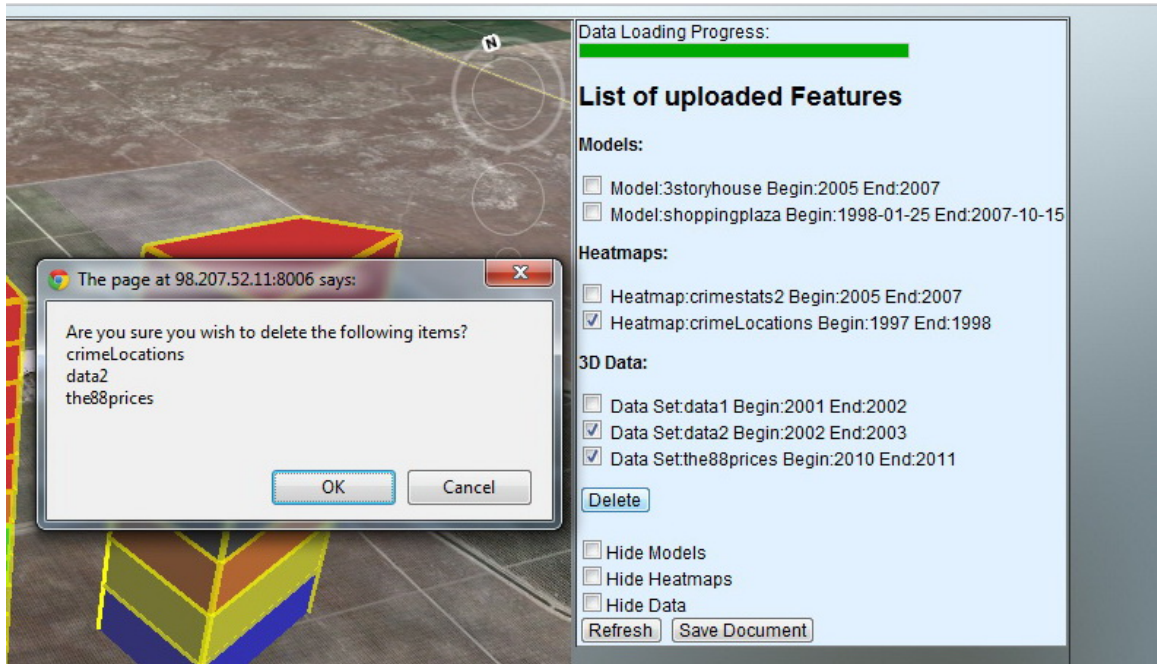


Figure 17 – Deleting multiple items

When items are selected and submitted for deleting, an alert box is presented to confirm the deletion of the selected items. If deletion is confirmed, an array of the selected items is posted to a PHP script that deletes all necessary artifacts from the server such as saved data files, heatmap images, and KML files. The features selected for deletions are also deleted from their respective tables in the database. Each of the features are then removed from the Google Earth Plugin instance by matching the features id with its unique id within the features of the plugin instance and removing it. Once deleted, a new request is sent to query the database and update the list of uploaded features accordingly.

### 3.6.3 Hide features

Each type of feature uploaded can be shown or hidden from view by toggling the check boxes for each type of feature as shown below.

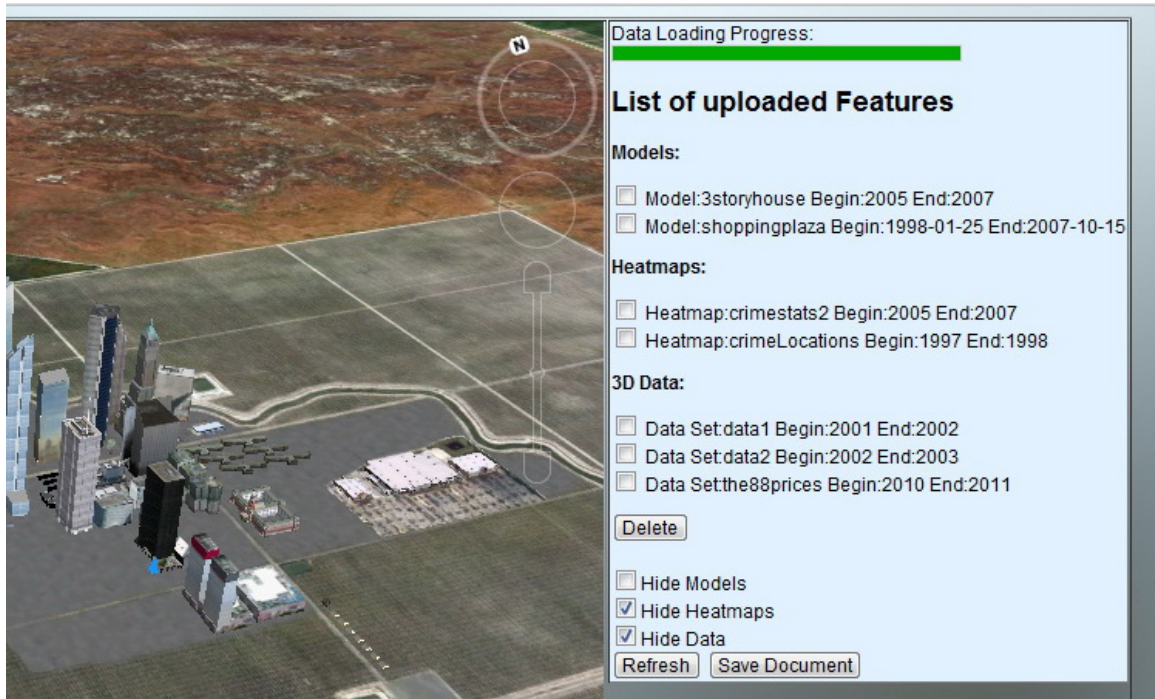


Figure 18 – Heatmaps and 3D data hidden from view

This is done by checking the feature type of each feature listed in the uploaded features panel and comparing it with each feature loaded into the plugin instance. Based on the feature type selected, the visibility of the feature is set to either true or false. This functionality is convenient for users to selectively display or hide uploaded features.

## 4. Conclusion

The GE Collab web application gives the ability to visualize historical structures and statistical data through changes in time. It has the ability to visualize both 2D and 3D data which most mapping tools lack. The 3D data visualization functionality is a useful tool that can be used to visualize and analyze complex and detailed data. GE Collab allows for easy collaboration and sharing of data due to its web accessibility. Bringing together all of these functions into a single application makes for a convenient and useful visualization tool.

## 5. Future work

There are a number of additional functions and modifications that can be implemented in GE Collab to improve its usability and functionality such as:

- Support for multiple sessions which will correspond to multiple Google Earth Plugin Instances.
- Automatic saving to prevent accidental loss of work.
- In addition to automatic saving, automatic refreshing and detection of changes if done by another user or machine to make application more collaboration friendly.

- Support for greater range of data colors.
- Model manipulation and control improvements.
- User Interface organization

## 6. References

- [1] Welcome to GRASS GIS, retrieved August 2012 from <http://grass.osgeo.org/index.php>
- [2] What is Google Earth Pro, retrieved September 2011 from <http://support.google.com/earth/bin/answer.py?hl=en&answer=189188&topic=2376997&ctx=topic>
- [3] Google Earth and Maps Enterprise retrieved August 2012 from <http://www.google.com/enterprise/earthmaps/earthpro-compare.html>
- [4] Google Earth, retrieved December 2011 from [http://en.wikipedia.org/wiki/Google\\_Earth](http://en.wikipedia.org/wiki/Google_Earth)
- [5] COLLADA – 3D Asset Exchange Schema, retrieved August 2012 from <http://www.khronos.org/collada/>
- [6] Trimble 3D Warehouse, retrieved September 2011 from <http://sketchup.google.com/3dwarehouse/>
- [7] Juice analytics: Heatmapping Google Earth, retrieved November 17, 2011 from <http://www.juiceanalytics.com/writing/heatmapping-google-earth/>
- [8] Unchartable: Time-animated Thematic Maps with US Census Data, retrieved on September 12, 2011 from <http://www.unchartable.com/2011/05/time-animated-thematic-maps-w-us-census.html>
- [9] KML Feature Interface Reference, retrieved August 2012 from [https://developers.google.com/earth/documentation/reference/interface\\_kml\\_feature](https://developers.google.com/earth/documentation/reference/interface_kml_feature)



## 7. Figures and Equations

### 7.1 Figures

1. Time Slider Bar	13
2. Simple heatmap example	14
3. Variations in data with polygon heights	15
4. Generated heatmap picture	16
5. KML object inheritance tree	19
6. Uploaded features panel	20
7. Uploaded model form	21
8. Model position controls	22
9. Heatmap input file format	23
10. Heatmap data form	23
11. Heatmap data imported into GE Collab	25
12. Import 3D data form	26
13. 3D data visualizations format	26
14. Examples of imported 3d data	28
15. 3D heatmap data input file format	30
16. 3d heatmap data uploaded	33
17. Deleting multiple items	35
18. Hide features checkboxes	36

### 7.2 Equations

1. Calculating normalized height	29
----------------------------------	----