Techniques in Google Earth and Google Maps

Michael Peterson, Konal J. Dobson, Kevin Fandry, and William Shrader
University of Nebraska at Omaha
mpeterson@unomaha.edu

INTRODUCTION

The three contributions here come from a graduate course entitled "Cartographic Methods," taught by Michael Peterson at the University of Nebraska at Omaha during the 2012 Spring semester.

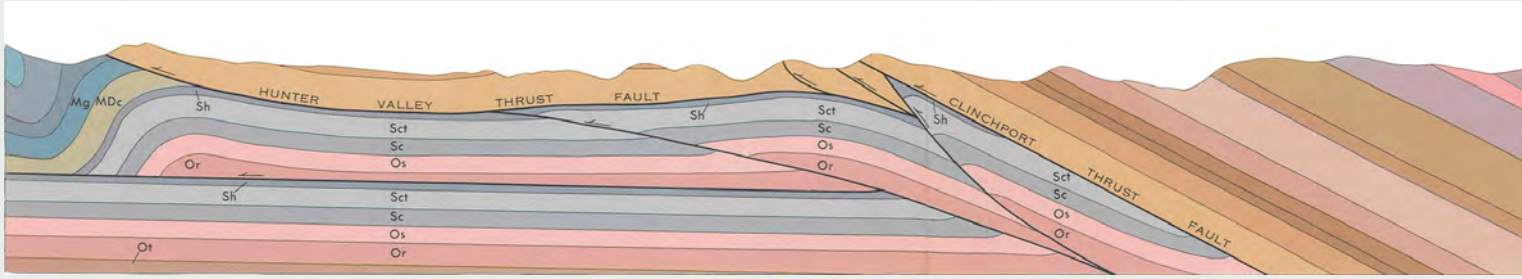
The course dealt with various ways of using cloud resources to make maps. Konal Dobson begins by examining how Google Earth can be used to depict subsurface geology through pop-up block diagrams. Kevin Fandry uses the timespan function in Google Earth to create an interactive animation. William Shrader then examines alternative ways for producing so-called "heat maps" through Google Maps.

—Michael Peterson

VISUALIZING SUBSURFACE GEOLOGY WITH GOOGLE EARTH
Konal J. Dobson

INTRODUCTION

Google Earth is a useful interactive mapping tool that allows users to control oblique perspective views of the earth. This perspective view makes it possible to blend three-dimensional features of the earth's surface with traditional two-dimensional map space. The Google Earth user can view a mountain range obliquely by controlling the angle of view. It is also possible to input a building as a 3-D object. This allows users to view a three-dimensional cityscape.



A 3-D element is added to Google Earth by building a model that consists of a block that's been draped with pixels to give a representation of the building's exterior. Users of Google Earth could create models other than buildings and share those models. One suggestion for a useful 3-D model other than a building is a model of subsurface geology. Since a building consists of a block model that extends in the positive z-direction, a user could create a block model of subsurface geology that would extend from the surface of the earth into the negative z-direction. This article reviews the procedure for creating a 3-D model of subsurface geology, sharing that model with Google Earth, and controlling its display so the user can elevate the block from within Google Earth, thereby making it possible to view subsurface geologic features.

PREVIOUS RESEARCH

Research into this concept had been previously conducted by Declan De Paor in 2009. Dr. De Paor states that geologists and geophysicists are interested in what is below the surface of the earth rather than what's above the surface. Google Earth is quite capable of displaying the earth's surface, and with the added modeling capabilities, Dr. De Paor has been able to create models of subsurface geology that can be viewed in Google Earth. Dr. De Paor considers it a powerful pedagogic tool to be able not only to view subsurface geology but to actually lift a block up out of the ground and see subsurface geologic features. Students can, in this way, connect surface features with the underlying subsurface geology. Many of the techniques reviewed in this article were developed by Dr. De Paor.

REQUIRED PROGRAMS AND MATERIALS

To begin, a user will need both Google Earth and Google's 3-D modeling program, SketchUp. Both of these programs are available as free downloads. Next, the user needs a geologic map of any area. The geologic map needs to include a cross-section and a traditional map view that shows the spatial location of the cross-section that is to be displayed. For this example, a geologic quadrangle map titled Swan Island Quadrangle, Tennessee is used (USGS 1971). Our objective is to create a block that displays a cross-section on the face of the block. For this purpose, a user will need a digital image of the cross-section in one of a variety of formats. JPEG, PNG, and TIFF are all suitable formats for the cross-section image. The Swan Island geologic map used in this example was scanned from paper and saved as a JPEG image. A simple text editor is needed to adjust the underlying KML code within Google Earth.

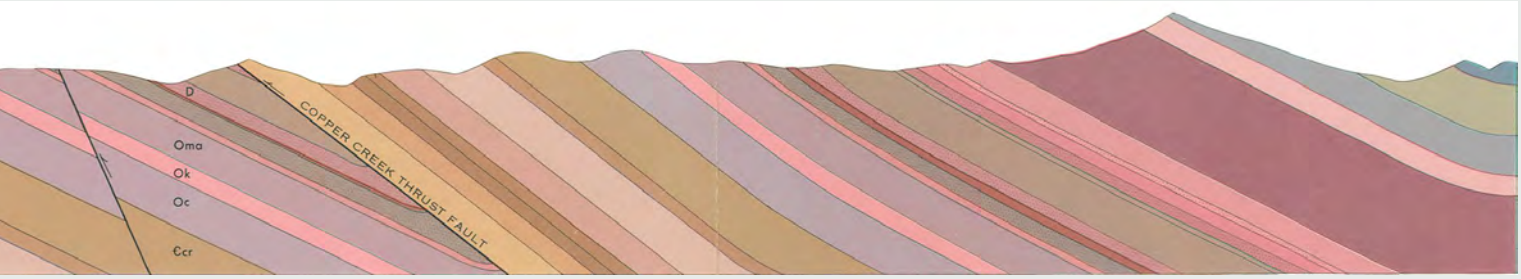


Figure 1. Swan Island Quadrangle, Tennessee, cross-section.

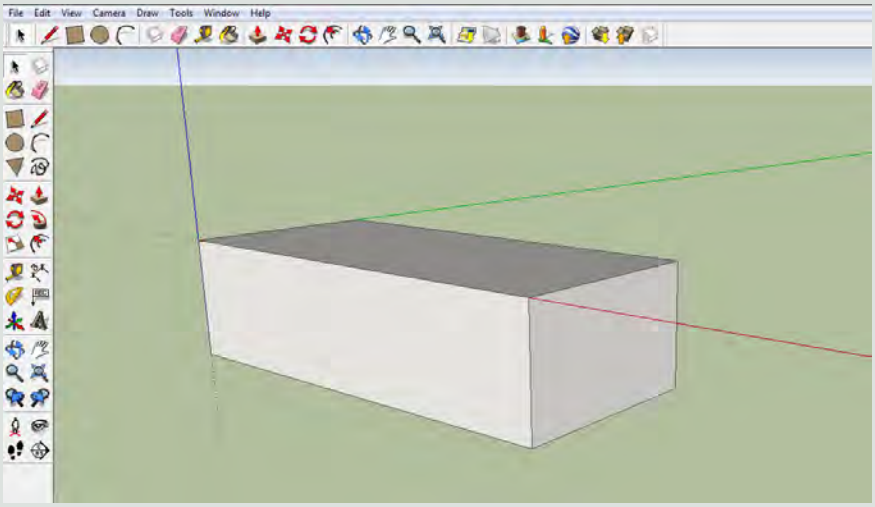


Figure 2. Block created in SketchUp extended in the negative Z direction.

STEPS IN MAKING A GEOLOGIC CROSS-SECTION FOR GOOGLE EARTH

1. In SketchUp, use the rectangle tool to create a rectangle on the ground surface.

2. Use the Push/Pull tool to extend the rectangle into a cube. The rectangle needs to extend from the subsurface in the negative Z direction (Figure 2).

3. For the cross-section that will be displayed on the face of the block select File > Import. Find the cross-section JPEG and open. Within the Import dialog box select the 'Use as texture' option.

4. Adjust the size of the block to fit the cross-section (Figure 3).

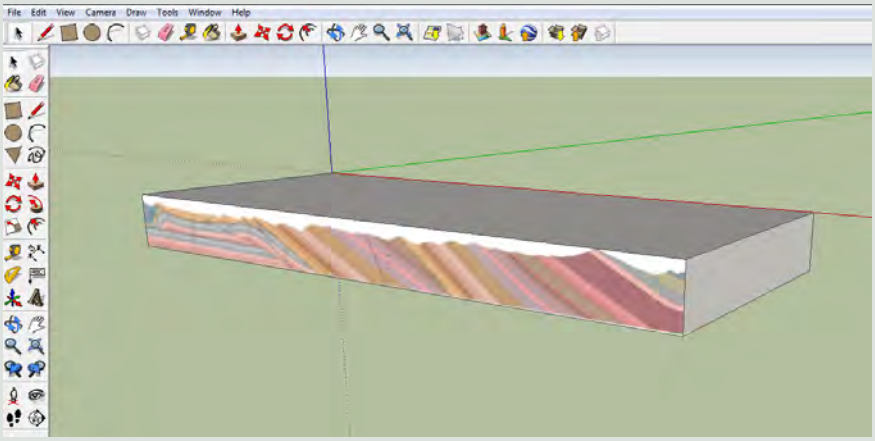


Figure 3. Block created in SketchUp with cross-section displayed on the face.

MARK GOOGLE EARTH WITH THE LOCATION OF THE CROSS-SECTION

With the model created in SketchUp, the next step is to find the location in Google Earth where this block is to be displayed. The Swan Island Quadrangle gives the lat/long coordinates of each corner. Use those coordinates to determine where to put the place marks in Google Earth. Next, use the 'Add Path' tool to mark the strike line that corresponds to the cross-section that is to be displayed.

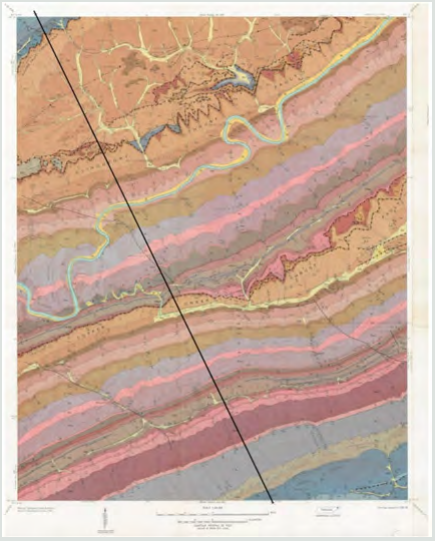


Figure 4. Swan Island geologic map with the strike line for the cross-section marked in bold black.

Figure 5. Screen capture from Google Earth showing the area of interest with strike line of cross-section marked in cyan.

STEPS FOR PLACING BLOCK IN GOOGLE EARTH

1. In SketchUp click the 'Preview Model in Google Earth' tool. This will open Google Earth and a layer titled 'SUPreview0' will have been added.

2. Expand SUPreview0 Layer to show two sub-layers.

3. Right click on the 'Model' sub-layer. Click 'Properties'.

4. With the 'Properties' dialog box open, the user can drag and resize the model so that it fits and lines up at the proper location.

5. Also within 'Properties' users can select the 'Altitude' tab and raise and lower the block to get a preview of how it will look in Google Earth (Figure 6).

6. After proper adjustments are made click 'OK' to close the properties dialog box.

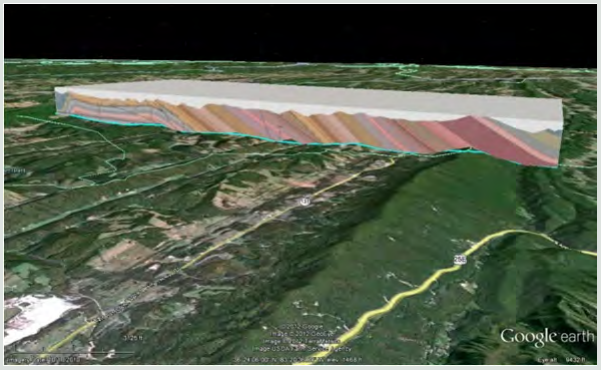


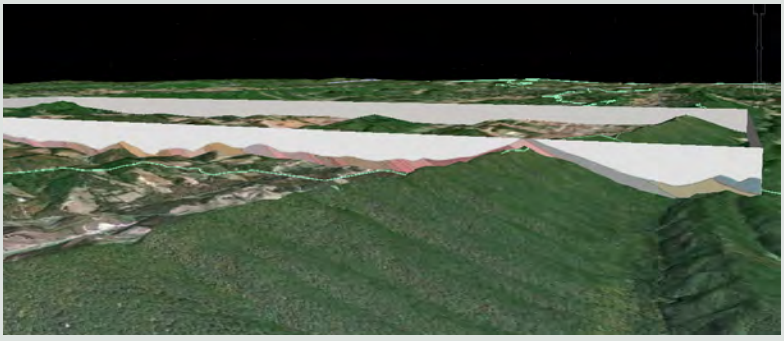
Figure 6. Screen capture from Google Earth showing the elevated block model with the cross-section displayed.

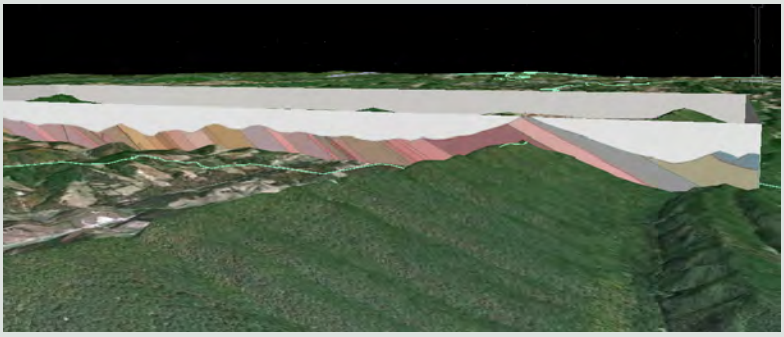
RAISING AND LOWERING THE BLOCK WITH THE TIMESPAN FUNCTION

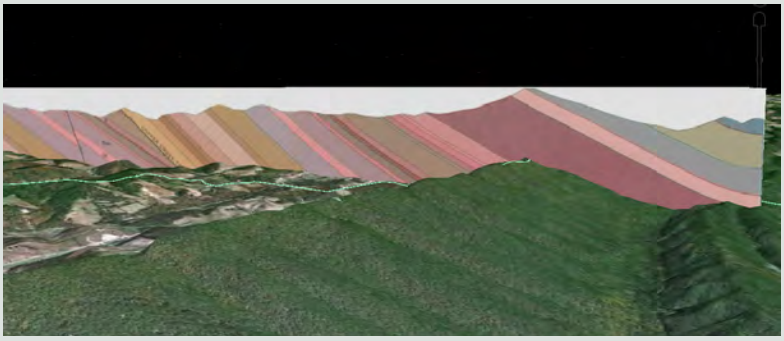
The model has now been created and shared in Google Earth. The next step is to make it possible to raise and lower the block. Raising and lowering this block is a powerful visual tool that will help students connect the surface terrain to

```
<Document>
        <name>Raise and Lower Block</name>
        <Folder>
                <name>Models</name>
                <open>1</open>
                <Placemark>
                        <name>1</name>
                        <TimeSpan>
                                <begin>0000</begin>
                                <end>0001</end>
                        </TimeSpan>
                        <Model id="model_2">
                                <altitudeMode>relativeToGround</altitudeMode>
                                <Location>
                                        <longitude>-83.3705302295524</longitude>
                                        <latitude>36.50006656192614</latitude>
                                        <altitude>0</altitude>
                                </Location>
                                <Orientation>
                                        <heading>242.6882871736554</heading>
                                        <tilt>0</tilt>
                                        <roll>0</roll>
                                </Orientation>
                                <Scale>
                                        <x>0.8405375957412673</x>
                                        <y>1.00000153230649</y>
                                        <z>0.9202695640238787</z>
                                </Scale>
                                <Link>
                                        <href>C:/Users/Konal/AppData/Roaming/Google/GoogleEarth/untitled.dae</href>
                                        <refreshInterval>1</refreshInterval>
                                        <viewRefreshTime>0</viewRefreshTime>
                                </Link>
                                <ResourceMap>
                                        <Alias>
                                                <targetHref>C:/Users/Konal/AppData/Roaming/Google/GoogleEarth/texture.jpg</targetHref>
                                                <sourceHref>untitled/texture.jpg</sourceHref>
                                        </Alias>
                                </ResourceMap>
                        </Model>
                </Placemark>
```

Figure 7. Code to implement timespan function.







Figures 8. Sequence of screen captures in Google Earth showing the block being raised.

the subsurface geologic features. For this, the timespan function can be used. The basic logic behind this function accepts an arbitrary span of time, 100 years for example. At year 1, the model can be set just below the Earth's surface. At year 10, the model can be set 10 meters in the air. At year 50, the model can be set at 50 meters in the air and so on. This will give users a slider bar that can be adjusted from 0 to 100 years which will raise and lower the model as the user slides the timespan slider back and forth. As adjustments are being made, Google Earth is writing underlying code in KML.

1. To view the code for the model, select 'SUPreview0' in the 'Layers' panel, right-click, copy and paste into a text editor. This is the underlying KML code in Google Earth for the block model.

2. Add 'Timespan' code shown in Figure 7 into the KML file in the text editor.

3. Since this code is specific to the Swan Island, TN, cross-section and location, fields within this timespan code will need to be replaced. Replace location, orientation, scale, link, and alias with figures specific to the user's location. These fields can all be found within the code that was copied and pasted in step 1 of this section.

4. Copy and Paste this section repeatedly to create each different timespan segment. This first section goes from <begin> 000 to <end> 001 with an altitude of 0. The next section would go from <begin> 001 to <end> 002 at an altitude of 10, for example. With repetition the user can create as many timespan segments as are needed, each with a sequentially higher elevation.

5. Save this code in the text editor with an extension of .kml.

6. Double-click the KML file; this will open it in Google Earth and raise and lower the block using the timespan slider.

SUMMARY

Google Earth is designed to visualize three-dimensional features in the environment. 3-D buildings have been added for many cities. An alternative application of this feature is viewing subsurface geology. Creating a block model of subsurface geology, sharing the model with Google Earth, and adding the timespan function to raise and lower the block in Google Earth results in a very useful representation. The presentation of the subsurface in this way could be a powerful tool for viewers to connect surface terrain to subsurface geology.

REFERENCES

De Paor, D. "AGU Scientists Tech Talks—Using Google SketchUp with Google Earth for Scientific Applications." January 27, 2009. Online video clip. Youtube. Accessed April 2, 2012.

USGS. 1971. Geologic Quadrangle Map, Swan Island Quadrangle, Tennessee, 1:24,000.

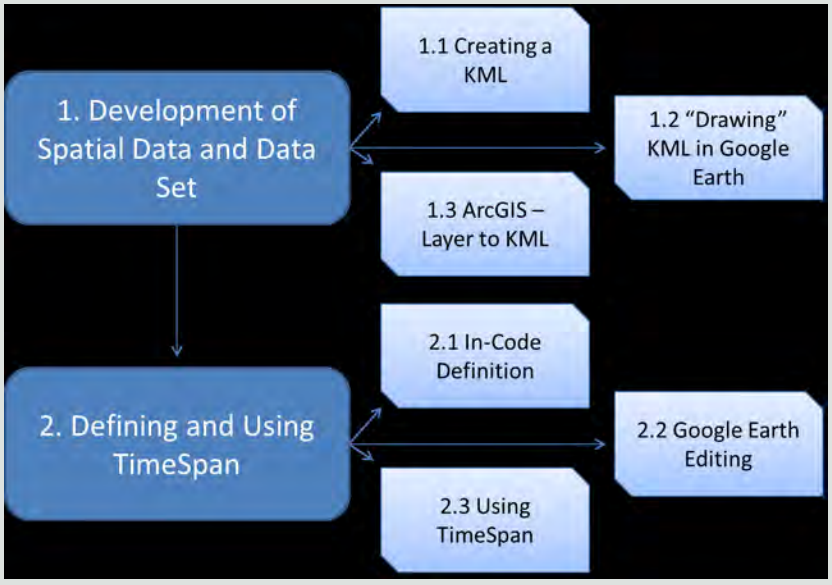
INTERACTIVE ANIMATION WITH GOOGLE EARTH'S TIMESPAN FUNCTION

Kevin Fandry

INTRODUCTION

Online mapping tools have reinvented the map animation interface, once criticized for its lack of interactivity. For example, the tiling system used by Google Maps allows for seamless zooming and scrolling between map images, redefining what animation within a map can do. Google Earth has expanded on animation by allowing a multitude of pre-programmed functions to be easily tied to a map. One such function is TimeSpan, a form of dating that allows users to turn data sets off and on through the use of a slider. As an improvement for interactivity within map animations, functions such as TimeSpan will increase user response to animated maps, helping create new ways of expressing information within maps.

The TimeSpan function is designed to show historical satellite imagery. Combining the functionality of TimeSpan with point, line, or polygon data would allow for the generation of multiple maps with increased functionality and interactivity. TimeSpan is a reference to a given period of time associated with a feature being represented through Google Earth. It can be contiguous, an example being the addition of US state polygons to show the order of addition to the Union. As the slider crosses the year in which a state was added, the associated polygons appear on the map. TimeSpan can also be non-contiguous by representing overlays of deforestation, advancement of glaciers—or, as in the example below, changes in agricultural practices.



Figures 1. Outline of key instructions to implement TimeSpan function within Google Earth.

IMPLEMENTING TIMESPAN

Using TimeSpan within Google Earth can be explained with a series of step-by-step instructions, as represented in Figure 1.

DEVELOPMENT OF SPATIAL DATA AND DATA SET

Google Earth uses the KML (Keyhole Markup Language) file as the basis for map generation and data storage. To take advantage of the TimeSpan function, a KML (or multiple KMLs) will need to be generated. There are different methods for generating a KML file; however, the desired product may dictate how the KML is created. The example provided is

representing changes in intensity in hog production at the county level for the state of North Carolina from 1950 to 2005. The data sets are from the United States Department of Agriculture (USDA) and the National Agricultural Statistics Service (NASS). Esri's ArcGIS is initially used for data processing and spatial representation.

When creating a KML from a layer within ArcGIS, the user must first design the layer to its desired representation. The example provided is a choropleth map, designed to aid users in noticing changes in quantitative data. Setting a projection for the map is not necessary, as it will be converted to fit Google Earth. With the layer designed appropriately, open the Layer to KML tool (which can be accessed through a tool search). The parameters needed for the Layer to KML tool consist of the Layer to be converted, an output name and location for the KML, and the output scale, as depicted in Figure 2. Output scale can be directly taken from the extent measure on the ArcGIS interface. The extent shown was 1:4,385,249; therefore, the output scale is simply 4385249. Repeat the process for the number of layers necessary, with each layer creating a new KML.

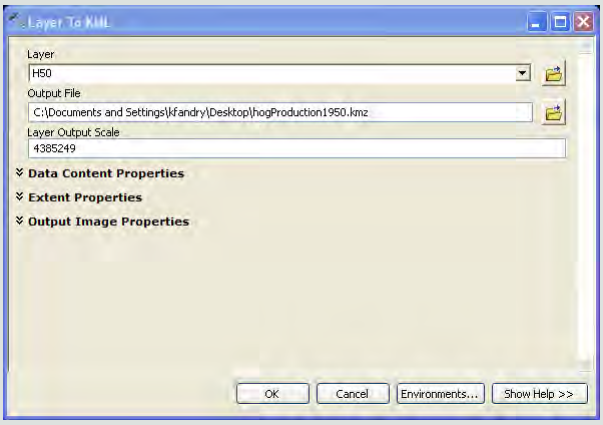


Figure 2. Display of the Layer to KML tool within ArcGIS, and necessary parameters.

Another method for generating a KML would be to use Google Earth itself. This can be done by either typing in the point locations needed to define the polygons or "drawing" the polygons using the tools provided by Google Earth. Both options are time-consuming. The geo-referencing capabilities provided through ArcGIS make it ideal for most scenarios.

```
1180  </body>
1181
1182  </html>]]></description>
1183  <TimeSpan>
1184      <begin>1950</begin>
1185      <end>2005</end>
1186  </TimeSpan>
1187      <styleUrl>#PolyStyle00</styleUrl>
1188      <MultiGeometry>
1189          <Polygon>
1190              <tessellate>1</tessellate>
1191              <outerBoundaryIs>
1192                  <LinearRing>
1193                      <coordinates>
1194                          -81.23970418775782,36.36549354576037,0 -81.17648398854539,36.415573(
1195                      </coordinates>
```

Figure 3. Displaying the placement of the TimeSpan within KML code.

DEFINING TIMESPAN

With the desired KMLs created, they can now be viewed within Google Earth. Applying the TimeSpan function is done in a number of ways, the first being to define TimeSpan within the code itself. Within Google Earth, right-click the folder labeled Temporary Places; this will copy and combine the code for all KML files currently active. Next, paste the copy into a preferred editing application, such as TextEditor, Xcode, NotePad, or NotePad++.

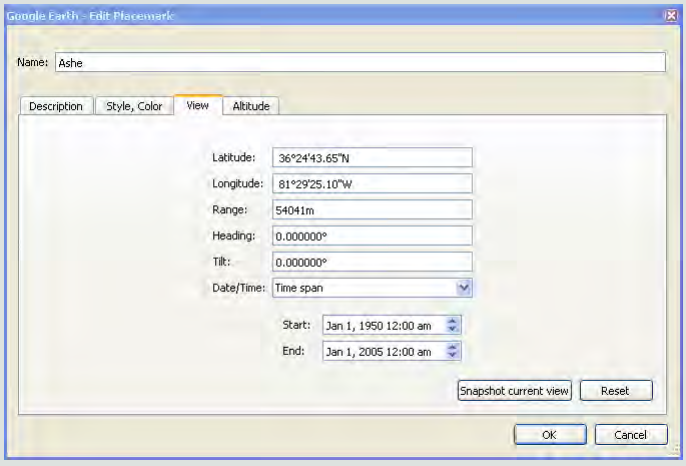


Figure 4. Displaying the Editor Window with Google Earth and necessary parameters to use TimeSpan.

When editing TimeSpan within the code itself, there are two key components: Placement and Format. Placement for TimeSpan generally occurs following the description call—or the call referencing the data set (hog production in this example) located within the KML—and precedes the geographic extent (coordinates) of the file, as shown in Figure 3. The TimeSpan function will need to be added to every geographic feature individually; for the example, this means every county for every layer. The Format for TimeSpan is as follows:

YYYY-MM-DDTHH:MM:SSZZZZZZ

- "Y" represents Year, the minimal requirement for TimeSpan to work
- The first "M" represents Month
- "D" represents Date
- "T" is a divider between Date and Time
- "H" represents Hour
- The second "M" represents Minute
- "S" represents Second
- "Z" represents the difference from UTC

Example: 1950-02-23T10:35:47+4:00

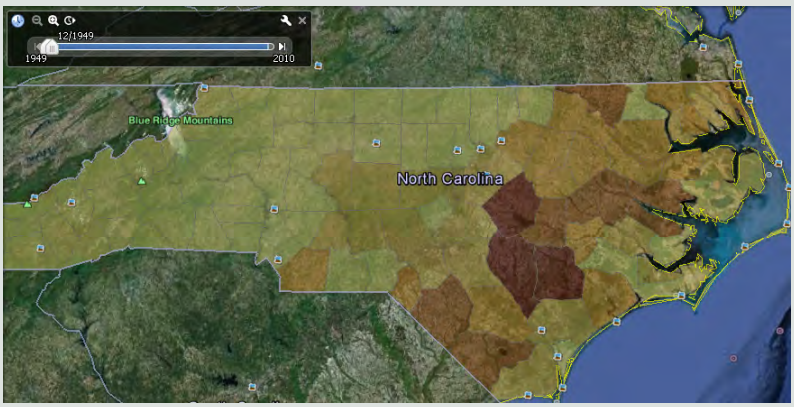


Figure 5. Active KML displaying hog production numbers in 1950 for North Carolina.

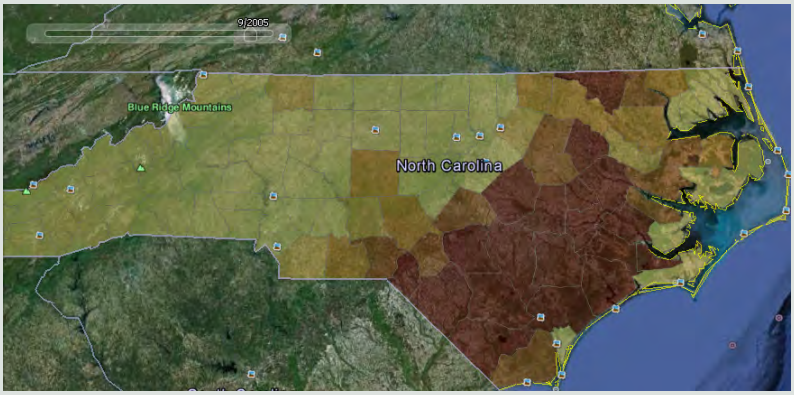


Figure 6. After transition through the slider bar, the KML now shows production numbers for 2005.

To implement the TimeSpan function, make a call to it and then include a "begin" time. For a contiguous map, an "end" time is not necessary. For the model example, which is non-contiguous, an end time is needed. Note: the time slider will stop in reference to the last "end" call. If creating a non-contiguous map, add an "end" call to the final KML layer, as shown in Figure 3.

Another method of implementing TimeSpan is within Google Earth through its editing window. Again, this will have to be applied to every geographic feature individually in each layer. To successfully edit within Google Earth, follow these steps in reference to Figure 4:

- Right-click on a feature and select "Properties/ Get Info" (Microsoft/Apple)
- Select "View"
- Select "Reset" if necessary
- Under "Date/Time" select "Time span"
- Add the desired "Start" and "End" times
- Click "OK"

Whichever method used, the result will be the same when displayed in Google Earth. When reopened, a slider bar now opens automatically with the KML, with a start time of 1950. As the slider bar reaches 2005, the image now switches to a new choropleth map as shown in Figures 5 and 6.

SUMMARY

The TimeSpan function allows for new methods of interactive animated mapping. The interactivity inherent within Google Earth and the TimeSpan function makes it possible for a multitude of data to be presented in this way, re-defining animated mapping. By adding in methods of interactivity to animated maps, their overall appeal to general map users may increase. Increasing appeal and use of animated maps will help advance the creation and a publication of new methods of spatial representation, therefore increasing the amount of knowledge that can be gained from maps.

REFERENCES

"KML Reference." Keyhole Markup Language. Google Developers. **https://developers.google.com/kml/documentation/kmlreference** (accessed 5 April, 2012)

"Time and Animation." Keyhole Markup Language. Google Developers. **https://developers.google.com/kml/documentation/kmlreference** (accessed 5 April, 2012)

GOOGLE HEAT MAPS
William Shrader

INTRODUCTION

Heat maps, more properly termed density or shaded isarithmic maps, use semi-transparent overlays to show the density or frequency of events with a yellow, orange, and red sequence of colors. Heat maps are effective for showing density information. The default options in Google Maps API Version 3 are limited. It is not possible to adjust the intensity or diffusion of the points. In addition, the map does not stay constant at different scales. These problems lead to poor representations of the information and can be misleading to a map viewer that is not aware of the problem.

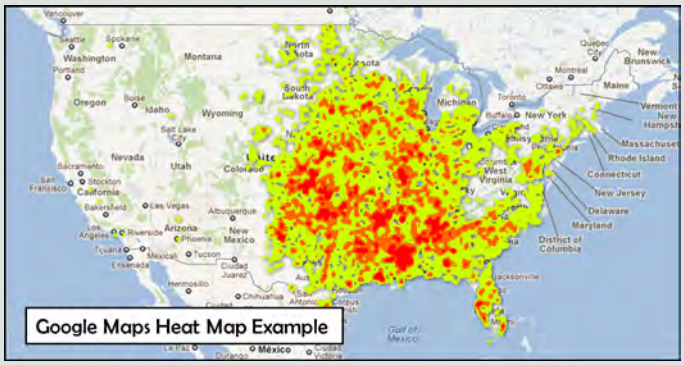


Figure 1. "Tornado Tracks 1950–2010," Fusion Table ID: 894938 from the Storm Prediction Center.

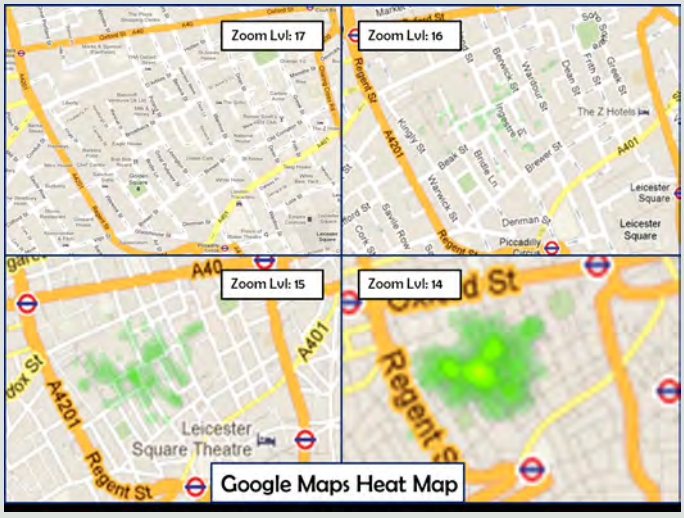


Figure 2. "Dr. John Snow's Ghost Map—Deaths," Fusion Table ID: 418541 (Dobson 2012).

An alternative to using Google's implementation of the heat map is to use the separate HeatMapAPI. A free version of this software enables users to adjust the intensity of heat maps by configuring the "boost" and "decay." This article will explain the advantages of the HeatMapAPI. The free version can be used as a tool for creating custom map mash-ups for a limited number of points.

HEAT MAPPING WITH GOOGLE MAPS

Figure 1 is an example of a heat map created using Google Maps. This map depicts the intensity of reported tornado occurrences in the United States from 1950 to 2010. This map is effective because it represents a large amount of points—a total of 28,916. Figure 2 shows another heat map, with far fewer points, of the famous "John Snow Cholera Map." Snow mapped the locations of deaths from a Cholera outbreak in London in 1854, helping him discover the source of the outbreak—a contaminated community water pump. The representation in Figure 2 is not as effective in showing this geographic distribution.

At zoom level 17, the 591 latitude and longitude points of the individual deaths are not visible. This is because they are located too far apart from each other at this zoom level. Zoom level 14 shows a good distribution of the points but there is not much variance in the intensity of the colors and the scale of the map prevents users from seeing any of the underlying detail.

HEAT MAPPING WITH THE HEATMAP API

The HeatMapAPI creates a layer that is placed on top of a Google Map. Figure 3 shows an example of the HeatMapAPI, that is using a smaller data set of the same latitude and longitude points as Figure 2. The data set is smaller because the free version of the HeatMapAPI is limited to 100 points. There are still

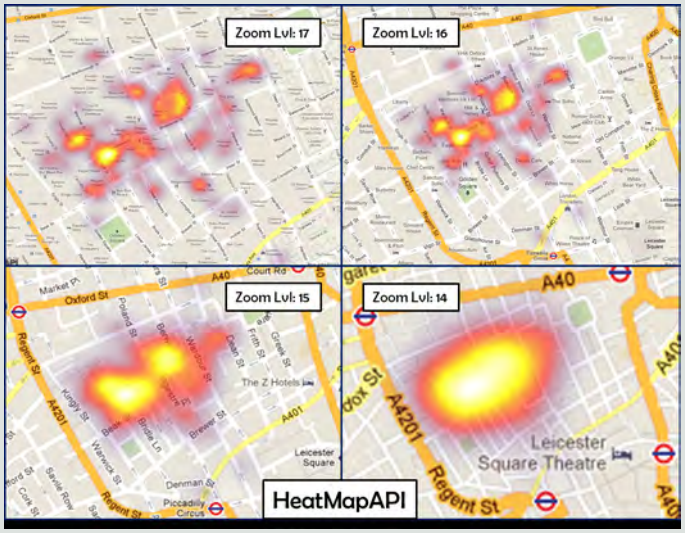


Figure 3. "Dr. John Snow's Ghost Map—Deaths." Data exported from Fusion Table ID: 418541. Created with HeatMapAPI and viewable at http://maps.unomaha.edu/GoogleMapGallery

differences between the various zoom levels. The map was made to be displayed at zoom level 17, for which the boost and decay have been adjusted.

The HeatMapAPI enables you to adjust the intensity of the heat map by adjusting the "boost" and "decay." By increasing the boost, the value of a single point will be greater over a larger distance. By increasing the decay, the value of a single point will be greater when it is adjacent to other nearby points. By decreasing the decay, the value will be less when adjacent to other nearby points. However, when adjusting the decay, points that aren't that near to others will hold their value, meaning that they will still be visible (Geospatial Analytics Inc. 2011).

Before using the HeatMapAPI, a key from the HeatMapAPI's website needs to be acquired, and then a simple Google map will need to be created using the appropriate code. To make the HeatMapAPI work, three scripts will need to be linked, as shown in Figure 4. Underneath the Google Maps API script, add the script for JQuery and the two scripts for the HeatMapAPI. This is where the personal HeatMapAPI key is needed (Geospatial Analytics Inc. 2011).

When using the free version of the HeatMapAPI, users are limited to inputting 100 or fewer latitude and longitude points. Paying for a license, which starts

at $39 a month, enables users to input more points. Inputting the points is relatively simple. Figure 5 shows one method for entering the latitude and longitude points.

The next section of code, Figure 6, pushes information to the proxy page and configures the style of the heat map layer and the Google map. Line 29 sets the pixel size for the heat map layer. For best results, make it the same as the Google map. Next is adjusting the boost and decay. Before this is done, decide on the appropriate zoom level for viewing the map, as this is the level at which adjustments should be made.

Figure 4. Adding Scripts (Hume 2012).

https://github.com/ahume/mapping-workshop/blob/master/code-examples/heatmap-api/libya.php

```
var points = [
{"lat": 51.513279, "lon": -0.136681 },     // input the latitudes and longitudes in this format
{"lat": 51.513690, "lon": -0.134675 }      // free version of HeatMapAPI is limited to 100 points
];
```

Figure 5. Format of latitude and longitude points (Hume 2012).

https://github.com/ahume/mapping-workshop/blob/master/code-examples/heatmap-api/libya.php

```
20  function init() {
21  var myHeatmap = new GEOHeatmap();                  // names the heatmap
22  data = [];                                         // collects the data for the proxy page
23  for (var i = 0, j = points.length; i<j; ++i) {
24  data.push(points[i].lat);
25  data.push(points[i].lon);
26  data.push(5);                                      // pushes the data to the proxy page
27  }
28                                  // configure HeatMapAPI
29  myHeatmap.Init(900, 700);       // set pixels for your maps size
30  myHeatmap.SetBoost(.1);         // increase the intensity of the heat
31  myHeatmap.SetDecay(0);          // decreases the intensity of the heat for points tha
32  myHeatmap.SetData(data);
33  myHeatmap.SetProxyURL('proxy.php');  // communicates with the HeatMapAPI proxy page
34  var latlng = new google.maps.LatLng(51.513353, -0.136621);
35  var myOptions = {
36  zoom: 17,
37  center: latlng,
38  mapTypeId: google.maps.MapTypeId.ROADMAP
39  };
40  var map = new google.maps.Map(document.querySelector("#map"), myOptions);
41  google.maps.event.addListener(map, 'idle', function(event) {
42  myHeatmap.AddOverlay(this, myHeatmap);  // places the HeatMapAPI on top of the Google map
43  });
44  }
45  window.onload = init;
46  })();
```

Figure 6. Proxy page that configures the heat map, and the Google map (Hume 2012).

https://github.com/ahume/mapping-workshop/blob/master/code-examples/heatmap-api/libya.php

The boost gives the input points a greater value that results in an increase in the resultant intensity. If it is left blank, it will defer to its default value, which is 1. Increasing the boost too much will result in the layer rendering very slowly, especially when using larger map sizes, so it is recommended to keep that value smaller, if possible (Geospatial Analytics Inc. 2011) .

The decay will be a number that is smaller than 1 and where the default is 0. At the default setting there is no decay. A number that is closer to 1 will increase the decay, or diffusion, to the points that are adjacent to each other. A number that is less than 0 will decrease the decay to adjacent points. Most users find that leaving this adjustment at 0 is sufficient for their purposes (Geospatial Analytics Inc. 2011). The best way to learn how to adjust the boost and decay is by simple trial and error.

The next step is to define the location and name of the proxy page, which is "proxy.php," and the proxy page will need to be located in the same directory as the web page (Geospatial Analytics Inc. 2011). The next step is to configure the Google Map. Set the latitude and longitude, zoom level, and map type. The last step is to add the HeatMapAPI layer onto the Google Map, as shown on line 42 in Figure 6.

The web page coding is complete, but there is one more crucial step to take before the map will work—creating the proxy page.

The purpose of the proxy page is to communicate between the client and the server; also, it prevents users from gaining access to the HeatMapAPI's private network. The proxy collects the latitude and longitude data and sends it to the HeatMapAPI web service. The web service takes the information and uses an algorithm to create the heat map layer, then sends it back to the proxy page and onto the web page.

The site **http://heatmapapi.com** has premade proxy pages for their users to copy, and they are available in six different web languages. For this demonstration, the proxy that was written in PHP was chosen. Save the proxy in the same directory as your web page or they will not be able to communicate with each other and provide the appropriate name such as "proxy.php" (Geospatial Analytics Inc. 2011). The heat map is now complete. Figure 7 shows an example of what the completed code should look like.

```
<html>
<head>
<style type="text/css">
html, body,#map {
 margin: 5;
 padding: 5;
 height: 95%;
 }
</style>            // add scripts. in addition to the Google Map API, add JQuery, HeatMapAPI, and HeatMapAPI key
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.js"></script>
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="http://www.heatmapapi.com/Javascript/HeatmapAPIGoogle3.js"></script>
<script type="text/javascript" src="http://www.heatmapapi.com/Javascript/HeatMapAPI3.aspx?k=YOUR_KEY_HERE"></script>
<script type="text/javascript">
(function() {
var points = [
{"lat": 51.513279, "lon": -0.136681 },      // input the latitudes and longitudes in this format
{"lat": 51.513690, "lon": -0.134675 }       // free version of HeatMapAPI is limited to 100 points
];
function init() {
var myHeatmap = new GEOHeatmap();               // names the heatmap
data = [];                                      // collects the data for the proxy page
for (var i = 0, j = points.length; i<j; ++i) {
data.push(points[i].lat);
data.push(points[i].lon);
data.push(5);                                   // pushes the data to the proxy page
}
                                    // configure HeatMapAPI
myHeatmap.Init(900, 700);           // set pixels for your maps size
myHeatmap.SetBoost(.1);             // increase the intensity of the heat
myHeatmap.SetDecay(0);              // decreases the intensity of the heat for points that are adjacent to each oth
myHeatmap.SetData(data);
myHeatmap.SetProxyURL('proxy.php');  // communicates with the HeatMapAPI proxy page
var latlng = new google.maps.LatLng(51.513353, -0.136621);
var myOptions = {
zoom: 17,
center: latlng,
mapTypeId: google.maps.MapTypeId.ROADMAP
};
var map = new google.maps.Map(document.querySelector("#map"), myOptions);
google.maps.event.addListener(map, 'idle', function(event) {
myHeatmap.AddOverlay(this, myHeatmap);  // places the HeatMapAPI on top of the Google map
});
}
window.onload = init;
})();
</script>
</head>
<body>
 <div id="map" style="width: 100%; height: 95%"></div>
</body>
</html>
```

Figure 7. Complete example of the code (Hume 2012).

https://github.com/ahume/mapping-workshop/blob/master/code-examples/heatmap-api/libya.php

SUMMARY

The HeatMapAPI has a number of advantages over Google's default heat map. Being able to adjust the boost and decay are the best features of this product. However, the free version is limited to 100 points or less, and paying for a HeatMapAPI license can be expensive. Also, the HeatMapAPI name is displayed on the maps when using the free version of the API. Another downfall is that at different zoom levels, the densities of the points change, which alter the representation of their intensity; however, this issue isn't nearly as much of a problem as it is in Google Maps implementation. Additionally, the rendering and display can be very slow. This could be fixed if the HeatMapAPI were to tile the heat map layer.

A useful additional feature would be a slider bar for adjusting the boost and decay. This would eliminate the trial and error when configuring the map, as

well as addressing discrepancies when changing zoom levels. Adding a variety of color schemes would be a nice feature as well. Despite these minor downfalls, the free HeatMapAPI is a valuable educational tool for understanding density mapping.

REFERENCES

Dodson, R. "Dr. John Snow's Ghost Map—Deaths." Fusion Tables ID: 418541, April 7, 2012 **https://www.google.com/fusiontables/DataSource?docid=1DIlOjeERBEaWHlu47L3_7EJEmbl2hinNDmsP9w** (accessed April 7, 2012)

Geospatial Analytics Inc. HeatMapAPI. **http://www.heatmapapi.com/Default.aspx** (accessed April 7, 2012)

Hume, A. Mapping—Workshop. **https://github.com/ahume/mapping-workshop/blob/master/code-examples/heatmap-api/libya.php** (accessed April 7, 2012)

Shrader, William, Dr. John Snow's Cholera Map, Density Heat Map. **http://maps.unomaha.edu/GoogleMapGallery** (accessed April 7, 2012)

Storm Prediction Center. "Tornado Tracks 1950-2010." Fusion Tables ID: 894938. **https://www.google.com/fusiontables/DataSource?docid=1vUFBtzPXPM1HSmwDTnNHcFvq2ch-UF49Vrpg8g** (accessed April 7, 2012)