Modeling Sinusoidal Climate Fluctuations

Mukhil Murugasamy

December 14, 2016

1 Abstract

The ClimaViz project consists of a JavaScript real time interactive computer animation for visualizing data animated in three dimensions. The program demonstrates how temperature fluctuations over the course of a year changes over time similar to a sinusoidal curve. The goal of the project is to depict a visual representation of climate change as it varies over time. Each modeled US city has a glyph used as the representation for the climatological data represented. Three different colored bars show how the temperature, humidity, and precipitation vary over time.

Figure 1: Display of one frame of the final project

2 Introduction

Local climate data [1] can be visually represented using oscillating sine curves that are modeled to fit real world data. The RTICA animates the humidity, precipitation, and temperature fluctuations of Austin, Chicago, Denver, Miami, San Francisco, and Washington DC. For each modeled city trigonometric functions are used to sinusoidally interpolate the

yearly fluctuations.

The mean monthly metrics for each modeled data set over a range of the past 20 years are used to model the average yearly fluctuations.[2] Typically the amount of variation in local weather for a single year does not fit a sinusoidal model due to random climate variability. The long-term average reduces the impact of very extreme events.

2.1 Context

Across the nation there is still a misunderstanding regarding climate change. There is a real problem of global warming that will affect everyone for many generations to come. The RTICA presents a model useful for demonstrating the changes in climate and regional variations across the US.

Anthony Leiserowitz's research group [3] has found huge differences across countries and world regions; for example, climate change is not seen as an issue in Latin America and South America showing just how misinformed the general public can be. Globally, 4 out of 10 people have never heard of climate change.

In the USA, forty-two percent of people incorrectly believe that since scientists can't predict the weather more than a few days in advance, they can't possibly predict the climate of the future. Thirty-seven percent believe that computer models are too unreliable to predict the climate of the future, neither of which is true.

It is our responsibility to ensure that the public is properly educated and understands the scale of how climate change can affect us. A simple two degree change in the average temperature can have catastrophic results on the environment.

3 Mathematics Used

For each modeled US city a separate sinusoidal equation is be used to model the temperature fluctuation. The purpose is to utilize real world data to create a sinusoidal interpolation that displays the average metrics for each month varying throughout the sample year.

A sinusoidal curve is given by the general form equation of $y = A\cos(Bx + C) + D$
Where A is the amplitude, B is the frequency of the function, C is the phase shift, and D is the vertical translation.

3.1 Deriving sinusoidal equation from data

Step 1: Determine the amplitude A of the function.
$A = (T - t)/2$
where T is the max temp and t the min temp.

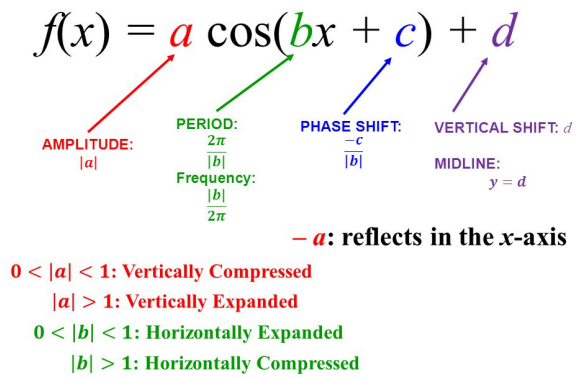

Figure 2: model of a basic cosine function [4]

Step 2: Determine the frequency B of the function.
In this case the period is 12 months for a yearly representation. Using the expression $2\pi/12$ we get $B = \pi/6$

Step 3: Determine the phase shift C by observing that in the parent cosine function the high point happens when t=0. Therefore, when modeling with the cosine, the phase shift (or delay) is the length of time between $t = 0$ and the first time the function reaches its maximum value. This is be calculated separately for each city since the maximum will occur during different months due to regional variations.

Step 4: Determine the vertical translation D of the function.
$D = (T + t)/2$

4 Project Details

The project uses JavaScript and HTML to create an RTICA. Three.js is the primary library used for creating three dimensional objects and rendering them in a JavaScript Canvas in the web browser. The application uses a `timePassed` variable which increment with each call to the rendered frame. The graphs are be modeled using sin curves and the resulting animation appears smooth to the viewer. The render function is called approximately sixty times per second and the `timePassed` variable is incremented for each rendered frame.

Each modeled US city contains a glyph with is an icon that displays the changes in climate with animated bars. Each glyph has three different colored bars that oscillate over time representing the changes in temperature with red, humidity with yellow, and precipitation with blue.

4.1 Creating the JavaScript Canvas

To be able to display objects with Three.js, three different components, a `scene`, a `camera`, and a `renderer`, are created so we can render the scene with the camera.

The `scene` is set and created using the following line of code inside the script tags.

```
   var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight,
0.2, 1000 );
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

4.2 Adding a point light source

A light source is necessary to illuminate objects and create shadows.
The pointlight object creates light that illuminates Lambert materials which is a material designed to specifically show shading of objects depending on the angle between the surface and the location of the light source. It casts light in all directions from a particular point using a specified color. 0xFFFFFF is used to create white light with the hexadecimal notation.

```
   var pointLight = new THREE.PointLight(0xFFFFFF);
pointLight.position.x = 10;
pointLight.position.y = 50;
pointLight.position.z = 130;
scene.add(pointLight);
```

4.3 Creating JavaScript Objects to be rendered

Three.js library provides classes to create many different 3D objects. Each 3D object from the javascript library requires a geometry shape and a mesh material. Since a point light sources was created the Lambert mesh material suits the needs of the project to provide the necessary shading creating the three dimensional perspective.

```
   var boxShape1 = new THREE.BoxGeometry(5, 1, 5);
var boxMesh1 = new THREE.MeshLambertMaterial(color:  0xaacc00);
var cube1 = new THREE.Mesh(boxShape1, boxMesh1);
scene.add(cube1);
```

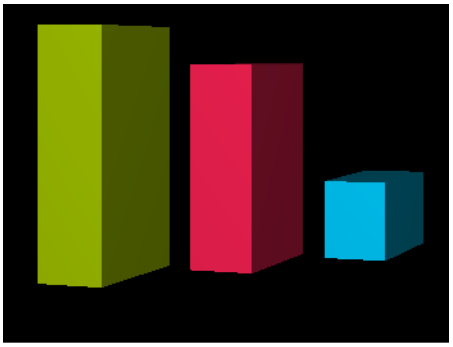

Figure 3: The three different color coded cubes

4.4 Creating the Pivot Container-object

Three.js by default creates an object starting from the point of its origin, outwards in the positive direction of each axis. It will still rotate the object based on the origin. In order to rotate the glyph from the center the three individual bars are added to a glyph object to combine them. Then the glyph object is added another mesh object to serve as the pivot. The glyph is translated within the pivot object by half the width and depth so that the center of the glyph will be at the origin on the pivot.

```
   var glyph = new THREE.Object3D();
var pivot = new THREE.Object3D();
var cube1 = new THREE.Mesh(boxShape1, boxMesh1);
glyph.add(cube1);
glyph.add(cube2);
glyph.add(cube3);

   pivot.add(glyph);
glyph.position.x = -(5.0/2.0);
glyph.position.z = -(5.0/2.0);
scene.add(pivot);
```

4.5 Creating the plane for the US map

To create an image of the US map to place the glyph objects on a plane geometry object is used. The plane geometry serves as the flat 2D object that the US map image will be projected on. In order to texture map an image of the US map to the plane a `loader` object is created. The load function adds the png image file to the flat plane object. A texture function is used as the second parameter of the load function which dynamically renders the png image texture onto a created plane. Inside the texture function the `usMap` object is created with the proper plane geometry and the map material loaded from the image file and the object is added to the `scene` to be rendered.

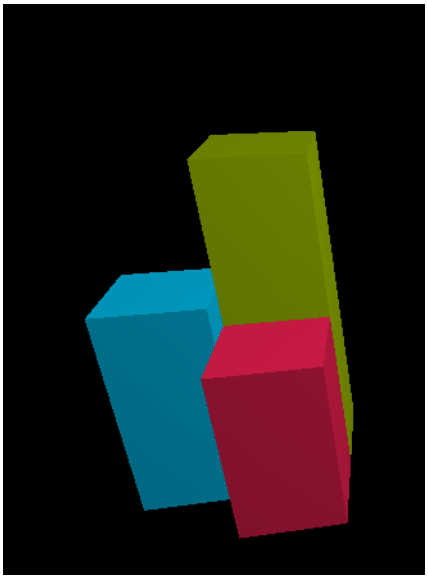

Figure 4: View of the glyph that is added to each modeled US city.

```
   var loader = new THREE.TextureLoader();
loader.load(
"usmap.png",
function texture {
      var mapGeometry = new THREE.PlaneGeometry(200,70,200);
      var mapMaterial = new THREE.MeshBasicMaterial(map:  texture);
      var usMap = new THREE.Mesh(mapGeometry, mapMaterial);
      usMap.rotation.x = -1.3;
      scene.add(usMap);
});
```

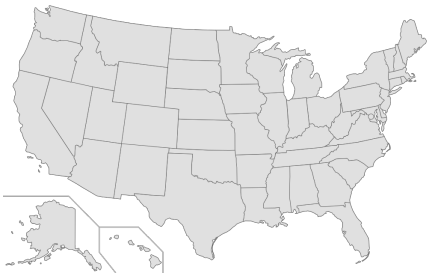

Figure 5: US map added to a plane

References

[1] "Western Regional Climate Center" *National Weather Service.* (n.d.). Retrieved October 31, 2016, from http://www.wrcc.dri.edu/CLIMATEDATA.html

[2] "NOAA" *National Centers for Environmental Information.* (n.d.). Retrieved October 31, 2016, from https://www.ncdc.noaa.gov/

[3] Leiserowitz, Anthony. "Americans' Knowledge of Climate Change." *Americans' Knowledge of Climate Change.* Yale University, n.d. Web. 8 Nov. 2016.

[4] Freeman, John. "Graphing Cosine and Tangent Functions." *Ppt.* N.p., 12 Nov. 2015. Web. 05 Dec. 2016.