Lab: Creating an Animation of Historical U.S. City Population Data

GEOG 319/658 **Exercise #10** **FALL 2014**

Creating an Animation of Historical U.S. City Population Data

Due: Monday, December 5

In this exercise you will be creating an animated map of the population of 196 U.S. cities for the years 1790-1990 at ten-year intervals. As discussed in class, the data for this exercise were originally used in the program MapTime and thus were initially available as the csv file Animation_Data.csv. You should download this file from Blackboard and examine its contents in Excel.

To utilize this file, we have modified it as follows:

1. Converted the csv file to a shapefile.
2. Unprojected the shapefile to WGS84.
3. Converted the resulting shapefile to a JSON file.

The resulting JSON file is called CityPopulationSeries.json and should be downloaded from Blackboard. Open this file in Notepad++ and examine its contents.

We are providing a skeleton html file that you will be modifying to create the final html file for the map animation. The skeleton file is known as CityPop_Animation_Skeleton.html and can be downloaded from Blackboard.

Also be sure that you have available the jquery -1.7.1.min.js file that we’ve been working with in some of the recent exercises.

The following is a summary of the steps that you will need to complete in order to modify the skeleton and create a complete map animation of the city population data. You will see associated numbers and asterisks in the skeleton file. Replace the numbers and asterisks with the appropriate documentation or computer code.

(1) ***
At the beginning of the program, indicate the following:
1. The input used for the program (i.e., the location and population data that are stored in the JSON file).
2. The nature of the map animation that is produced.
3. Any interactions that the user makes with the map.

(2) ***

In the “year_panel” section of the “style” section, experiment with positioning the year identifier for each frame of the map animation. You will need to replace ??? with an appropriate number of pixels. You may also wish to experiment with other parameters here and in the “panel” for the start and stop animation buttons, which appears just after the “year_panel”.

(3)***
Study the global variables as these will be used at various places in the program.
(4)***
Include two functions here, one called FindMaxValue that will find the maximum value of the elements of an array, and another called CircleRadius that determines the radius of a circle given a data value, the maximum data value, and the maximum desired radius. You should be able to use functions from exercise 7D.

(5)***
Note that we have included an “initialize” function that is very similar in structure to the initialize function from earlier exercises.

(6)***
This is followed by the CreateCityCircles function, which does a variety of things, including accessing the population data stored in the JSON file, finding the largest data value for all cities over all time periods, creates all circles for all time periods, and creates the first frame of the animation. We provide some code here and ask you to write some of the code.

(6a)
***

> Create a loop that will populate the cityLoc array with latLng objects representing the locations of the cities. You code should read the JSON file and obtain each city’s latitude and longitude and add a latLng object to the array.

(6b)
***

> In the section beginning “Find the max population…”, note that “yearlyMax” is an array that stores the maximum population value for each year. For (6b)***, you need to use the FindMaxValue function to determine the maximum population for a year and then store this result in the “yearlyMax” array.

(6c)
***

Again use the FindMaxValue function to determine the maximum population value for all the years. Store this result in a variable called maxPop.

(6d) ***

Specify the radius for the largest circle on your map. You could start with 150000, but you probably will want to modify this.

(6e)
***

Study the section of the code following the specification of the radius. To create all circles for the current year, you will create computer code corresponding to the following pseudo code:

```
For each city
If the population of the city > 0 then
Determine the radius for this city
```

```
Create a circle for this city and use the Array object’s “push” method to add the resulting circle to the cityPopCirs array.
End If
End For
```

(7)***
Include a function called CreateCircle that will actually create a circle at a specified location with a specified radius. Here again you should be able to use a similar function from exercise 7D. Since many of your circles will overlap, you may want to experiment with various parameters, such as the fillOpacity.

(8)***
The functions startAnimation and stopAnimation are used to start and stop the animation. Experiment with the
???? to determine an appropriate number of milliseconds between frames.

(9)***
Study the function AnimateCircles that displays various frames of the animation. Create the code that will “Add current year circles to the map” (9a) ***. Current year here means the year with the curYearIdx index in the cirsByYears array.

(10)***
Create buttons that will start and stop the animation.

Create a folder for exercise 10 on the webhosting service. Load all files related to this exercise into this folder. Post the resulting URL for the map animation to the following Google Worksheet . Hand in a hard copy version of your map animation code and documentation.