Creating Mercator's Map Projection

Andrew Geldean
December 17, 2014

Abstract: This map developed by Gerardus Mercator in 1569 is created by producing a cylinder around the globe projecting the surface onto the cylinder. This type of map projection thus conserves the angles with the meridians (making the projection conformal.) The disadvantage to this kind of map, however, is that it distorts the size and shape of large objects, and the scale of distortion increases exponentially as it the latitude approaches the poles.

1. Mathematics behind the Projection

This projection, as previously stated, is created by encompassing the spherical earth within a cylinder.

1.1 Projecting the sphere

This projection, as previously stated, is created by encompassing the spherical earth within a cylinder. The surface of the globe is then stretched to fit the surface of the cylinder, with the poles being infinitely stretched.

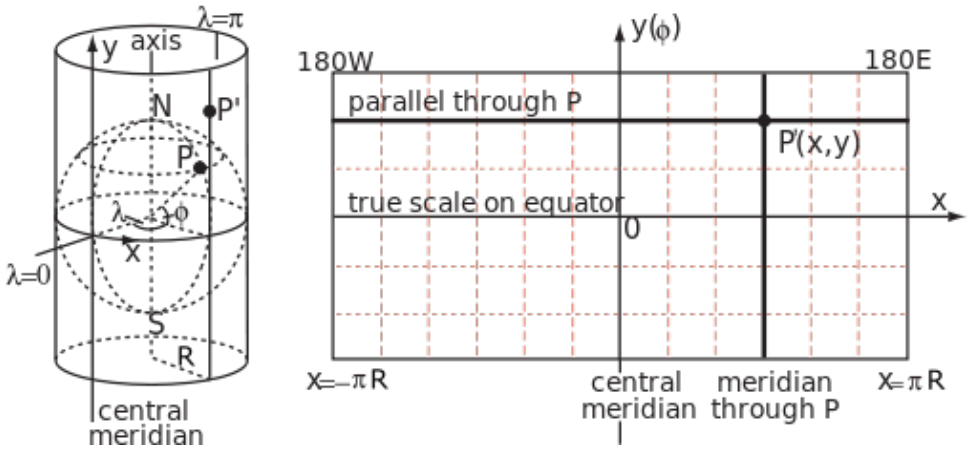

Figure 1 (i): The basic geometry for a cylindrical map projection

The previous diagram displays how the x and y coordinates are projected onto the map from the surface of the globe. Using Φ and λ as variables to represent the latitude and longitude angles in the spherical geometry of the earth, one can determine the scale factor between the sphere and the cylinder to be *sec (Φ)*. The parallel scale factor produced is denoted by *k* and the meridian scale factor is denoted by *h*.

1.2 Small Element Geometry

Using small element geometry (taking extremely small elements on the surface of the earth and the cylinder), one can produce the similar graphs.

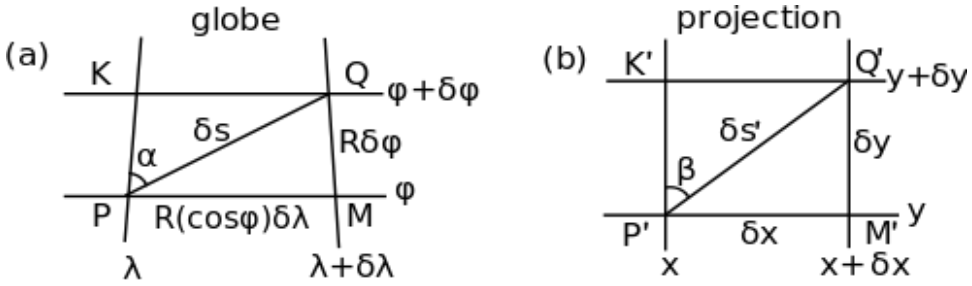

Figure 2 (ii): A comparison of infinitesimally small elements on a sphere (globe) and the plane (map)

Since in small element geometry, the angles α and β are approximately equal and the following derivations are produced:

The scaling factors k and h can also be calculated as well:

$$\tan\alpha \approx \frac{R\cos\phi\,\delta\lambda}{R\,\delta\phi}, \qquad \tan\beta = \frac{\delta x}{\delta y},$$

$$k(\phi) = \frac{P'M'}{PM} = \frac{\delta x}{R\cos\phi\,\delta\lambda},$$

$$h(\phi) = \frac{P'K'}{PK} = \frac{\delta y}{R\delta\phi}.$$

Finally, since the x coordinate on the map is aligned with the meridians, the three following equations represent the limit of infinitesimally small elements:

$$\tan\beta = \frac{R\sec\phi}{y'(\phi)}\tan\alpha\,, \qquad k = \sec\phi\,, \qquad h = \frac{y'(\phi)}{R}.$$

1.3 Derivation of Mercator Projection

In order to find the equation for *y*, one must make the map conformal by either setting the angles α and β equal to each other or the scale factors *h* and *k* equal to each other. By doing this, the equation y'(Φ) = R sec (Φ). Integrating this equation gives y(Φ) and the coordinate x can be calculated very easy with the variable λ:

$$x = R(\lambda - \lambda_0), \qquad y = R\ln\left[\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right].$$

2. Programming the Projection

For my project, I planned to create the spherical earth using the JavaScript THREE.js library and rotate the sphere encompassed by the cylinder while projecting the map.

2.1 Setting up the THREE.js scene

The first step necessary was to create the THREE.js scene. This required four different items: the scene, camera, renderer, and light source. Each one is necessary in creating 3D objects with this library. The scene variable creates the actual scene. The camera allows the user to view the scene. The renderer essentially renders the scene. Finally, the light source allows the objects to look more three-dimensional.

2.2 Creating the Sphere

Creating the sphere is a fairly simple process. It requires the declaration of three different variables: the material, type of geometry, and then the mesh object being created. Then just add the newly created object to the scene.

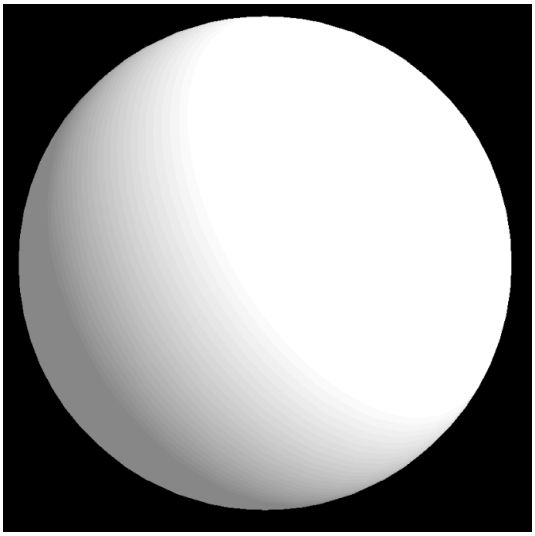

Figure 3: The newly created sphere (notice the shadows).

2.3 Adding Texture

The next step in creating the earth is to add the earth texture to the sphere. This proved to be a more complicated process than originally thought because of the library's likelihood to fail to load the image. I was able to correct this bug by adding an error function if in case the image texture failed to load. The result of this correction was a beautifully projected planet earth:

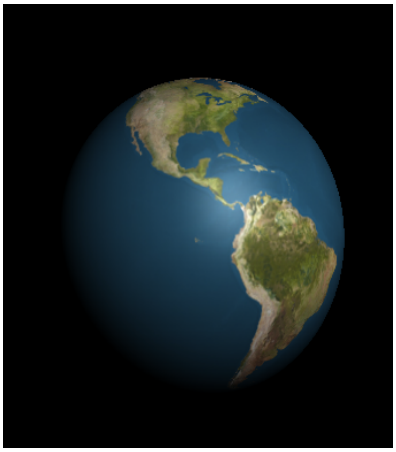

Figure 4: The sphere textured with the surface of the earth.

2.4 Rotation and Creating the Cylinder

The next step was to program keyboard commands to rotate the earth along the y-axis. This basically required the set up of two variables: leftpressed and rightpressed. Then, through creating key events, I was able to rotate the sphere left using the left arrow key and right using the right arrow key. Then adding the cylinder is the same process as the sphere except, instead of using spherical geometry, cylinder geometry is used instead. Adding a textured latitude and longitude grid to the cylinder results in the following:

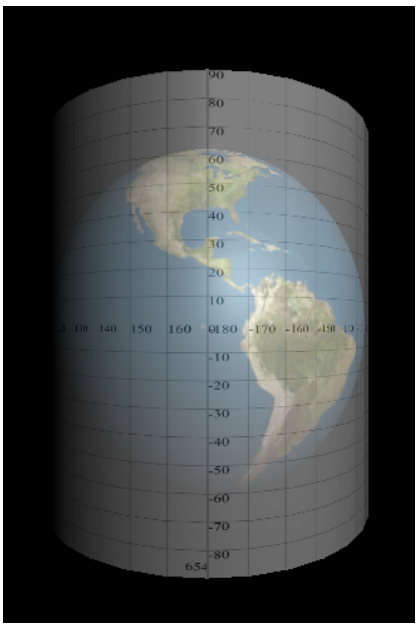

Figure 5: The textured cylinder encompassing the earth.

2.5 Adding Mercator's Map

There are two approaches to creating the actual projected map: one simple method and an extremely complicated method involving the equations previously mentioned. This time, since it was my first experience with the THREE.js library, I chose to choose the simpler method by just finding an image from the internet to use as the projection. Then I just load the texture onto another object with planar geometry and place it behind the sphere.

2.6 Realigning the Map with the Sphere

Finally, using the previously created keyboard commands, I give the map the ability to scroll left and right and to only display one half the earth. By readjusting the speed of the map as it scrolls, I am able to synchronize the speed of the map and the speed at which the earth rotates, thus completing the final project. I final preview without movement can be seen below:

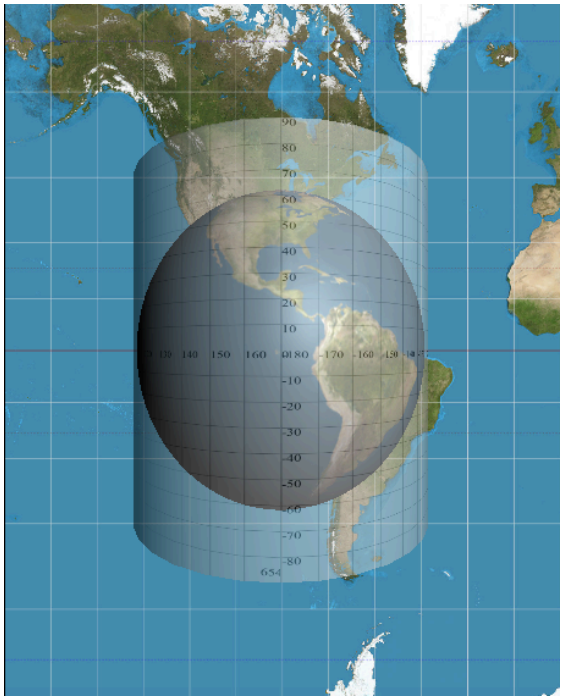

Figure 6: The completed cylindrical projection with the map in the background.

Bibliography:

(i & ii)
http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Mercator_projection.html

http://en.wikipedia.org/wiki/Mercator_projection

http://www.public.asu.edu/~aarios/resourcebank/maps/page10.html