Brett Stelly

Rendering Geometry

**How to generate a sphere given a half circle, and number of meridians.**

My function for generating a sphere takes in an int for the radius, an int for the number of points, and an int for the number of meridians.

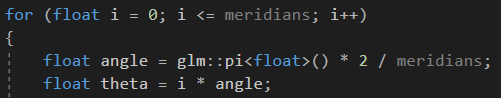


To generate a sphere, first declare two variables of type vector of type vec4. One variable will be named “points” and the other variable will be named totalPoints. The first variable “points” will be assigned the returned value of the function genHalfCircle. The arguments for the genSphere function will be passed into the genHalfCircle function.



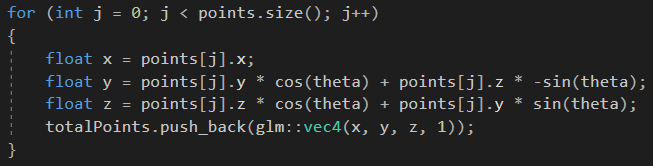
Next, we create a for loop. This for loop declares a variable of type float called ‘i’ and initializes it with zero. The for loop’s condition is that ‘i’ is less than the meridians passed into the genSphere function. The ‘i’ variable will increment by one every iteration of the for loop.

Within this for loop, we declare a variable of type float and call it angle. We assign angle with the result of pi multiplied by two divided by meridians. Now declare a variable of type float called theta. Theta is assigned ‘i’ multiplied by angle.



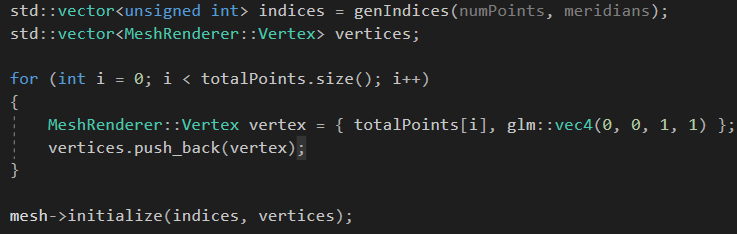
After declaring angle and theta, we create another for loop within the current for loop. This new for loop declares a variable of type float called ‘j’ and initializes it with zero. This for loop’s condition is that ‘j’ is less than the size of the points vector. Every iteration of the loop will increment ‘j’ by one.

This new for loop is going to create all of the points that will make up the sphere by rotating the points of the half circle. To do this, declare three new variables of type float. The first variable, called ‘x’, will be assigned the x value of points[j]. The second variable, ‘y’, will be assigned the sum of points[j] y value multiplied by cosine theta and points[j] z value multiplied by negative sine theta. Once the point has been rotated, push back a new vec4 with the arguments being x, y, z, and 1, onto totalPoints.



The totalPoints variable contains all of the points needed to generate a sphere, but my genSphere function draws the sphere.

Using the passed in number of points and number of meridians arguments I call the genIndices function and assign it to a new variable of type vector of type unsigned int. Next, I create a vector of the MeshRenderer class’s Vertex struct and call it “vertices”. A for loop is used to loop through all of the points in totalPoints and creates a new Vertex using the current index of totalPoints and a vec4 for color. After creating the new vertex, push back the vertex onto vertices. Once the for loop is finished running I call the MeshRenderer class’s initialize function and pass in the indices and vertices variables as arguments.



**Full Code:**

