Assignment 4 tips: Finding Texture Coordinates

Texture mapping onto a cone or a disc:

How to find texture coordinates for a cone or a disc?

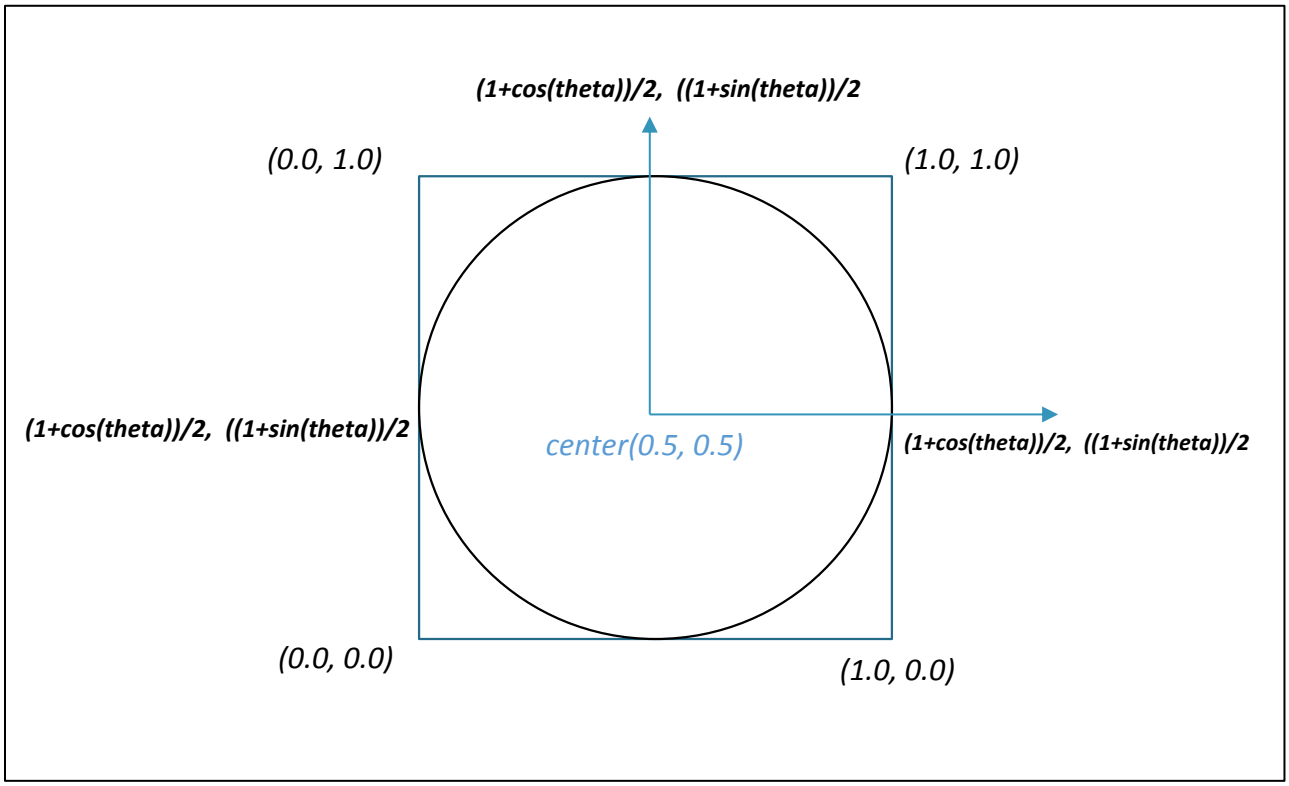

Figure 1: Finding texture coordinates in a cone or a disc.

Texture mapping onto a sphere:

Last vertex in a ring coincides with the very first vertex: Texture coordinates are two-dimensional. A 2D image needs to be wrapped around a sphere. We need to add redundant vertices (as shown in Figure 1) for proper texture mapping. Hence for texture mapping onto a sphere, last vertex in a particular ring needs to coincide with the very first vertex in that ring (Figure 2).

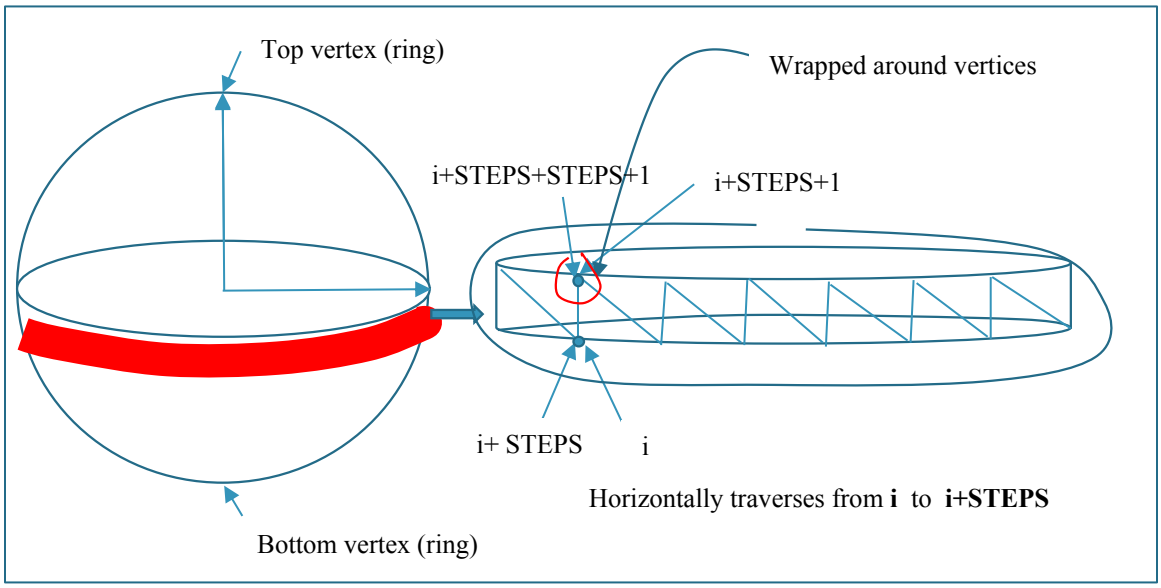

Figure 2: Constructing surface from two consecutive vertice rings and redundant vertices.

Top and bottom ring: Some other redundancy also needs to be considered for proper texture mapping. For top and bottom vertices, a number of texture coordinates (STEP +1) will be generated. All texture coordinates need to be mapped to equal number of vertices, i.e., same vertices for top and bottom Hence instead of one point at the top and one at the bottom (as we did before), STEP+1 texture coordinates will map to (STEP+1) vertices (same vertex) both for top and bottom. Now the following code calculates the vertex coordinates and texture coordinates

```
for (int b = -STEPS / 2; b <=  STEPS/2; b++) {
     for (int a = 0; a <= STEPS; a++) {
          phi = (a / STEPS) * 2 * PI;
          theta = (b / STEPS) *PI;
          vertex[i].x = cos(theta)*sin(phi);
          vertex[i].y = sin(theta);
          vertex[i].z = cos(theta)*cos(phi);
          vertex[i].w = 1.0f;

          textures[i].x = 1.0 * a / STEP;
          textures[i].y = (1.0 * b / STEP) + 0.5;
          i++;
     }
}
```

Tex coordinate:x
Along the ring

Tex coordinate: y
Across the ring.
0.5 added to make it positive (ranging from 0 to 1)

Figure 3: Code snippet for calculating vertices and texture coordinates.

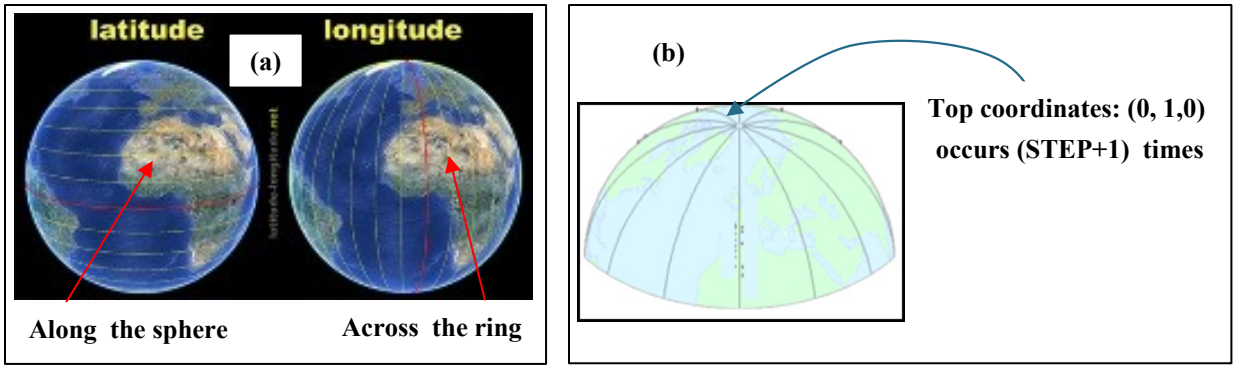

Figure 3(a) Lines along the sphere (x coordinates of texture) and across the sphere (y coordinates of texture) and (b) Top coordinates concentrated to one point (0, 1, 0) against a number of texture coordinates.

Considering the above-mentioned redundancies, NUMINDICES and NUMVERTICES can be calculated as follows:

```
#define NUMINDICES STEP*(STEP+1)*6
#define NUMVERTICES (STEP+1)*(STEP+1)
```

Triangulation:

From the code snippets for calculating vertex coordinates and texture coordinates (Figure 2), it is observed that at **b = -STEP/2** and at **b= STEP/2**, vertex coordinates are the same; (0, -1, 0) for bottom coordinate and (0, 1, 0) for top coordinate. But texture coordinates vary. As texture coordinates and vertex coordinates need to be equal, for the top and bottom vertices, the same vertex coordinates are mapped against a number of different texture coordinates.

Now, surface reconstruction (triangulation) is performed by considering two consecutive rings along the sphere. Triangulation is the same when middle rings are considered or top / bottom vertices (here ring) are considered. Why?

Because now we consider texture coordinates as well. Vertex coordinates and texture coordinates are equal for top and bottom coordinates, i.e., STEP + 1.

For calculation of indices with redundant vertices, the code can be simplified as follows:

```
for (int i = 0; i < (NUMVERTICES - STEP - 1); i += STEP + 1) {

            for (int j = i; j < (i + STEP); j++) {
                 indexOne = j;
                 indexTwo = j + STEP + 1;
                 indexThree = (j + 1) + STEP + 1;
                 indexFour = j + 1;

                 indices[index++] = indexOne;
                 indices[index++] = indexTwo;
                  :
                  :
                  :
        }
 }
```