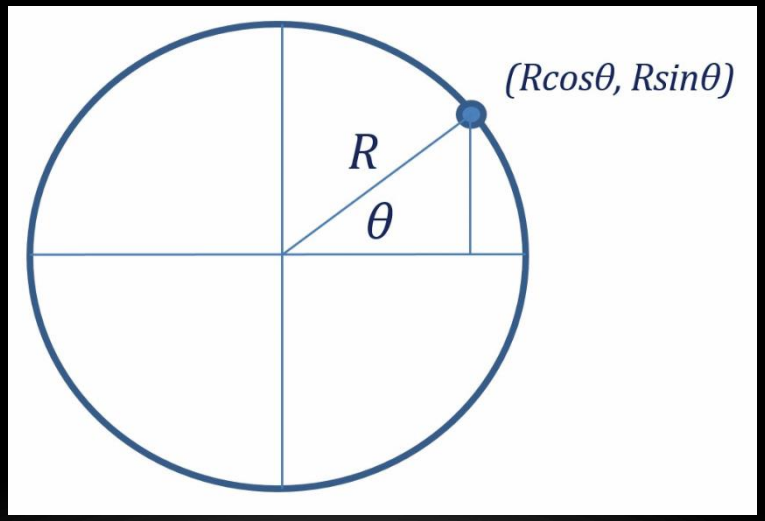
PROCEDURAL OBJECT MODELING

OUTLINE

- Building a Sphere
- Building a Torus

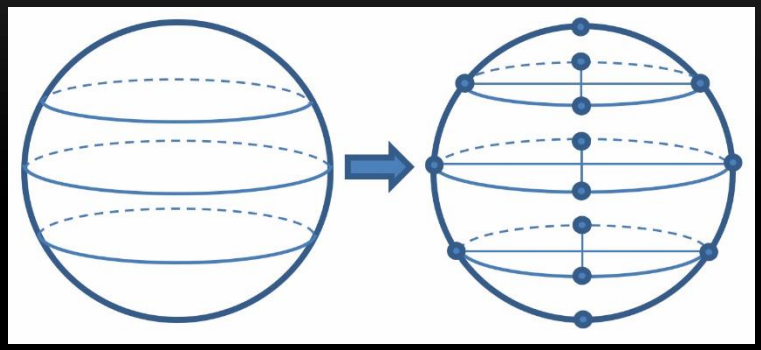
A SPHERE!!

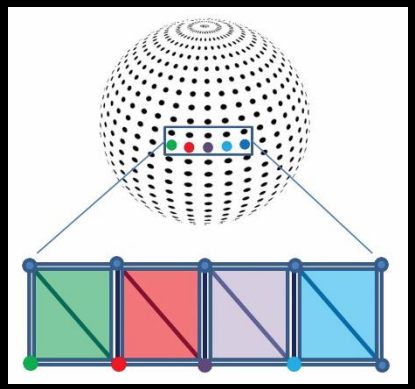
- Specifying vertices by hand is tedious
- Some objects are better represented mathematically
  - These are better expressed algorithmically



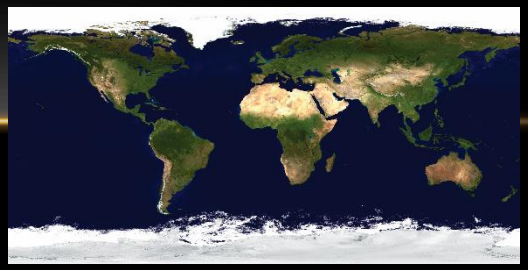
STRATEGY

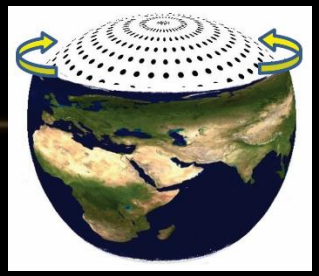
- Select a “precision”
  - The number of horizontal slices in the sphere
- Subdivide horizontal / circular slices into points





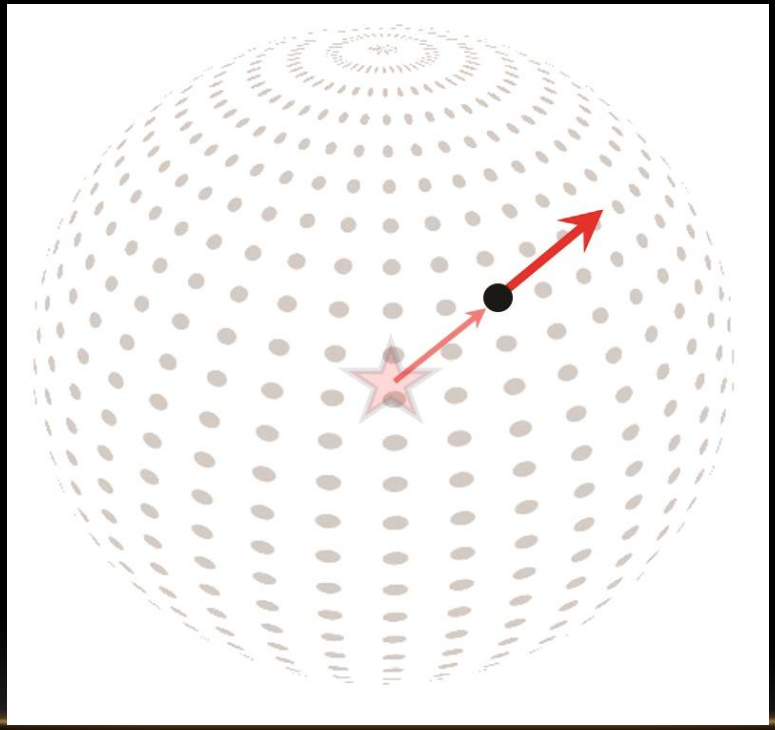
- Group vertices into triangles
- Select texture coordinates to wrap the sphere with





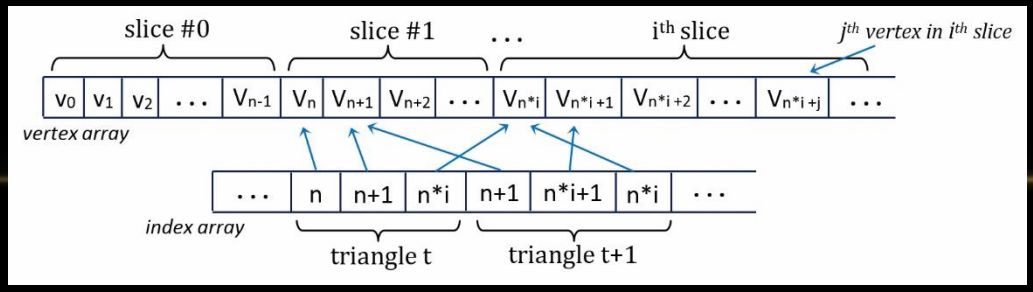
ONE MORE STEP IN THE STRATEGY

- Generate normal vectors
  - We're not using these yet, but we will when we get to lighting



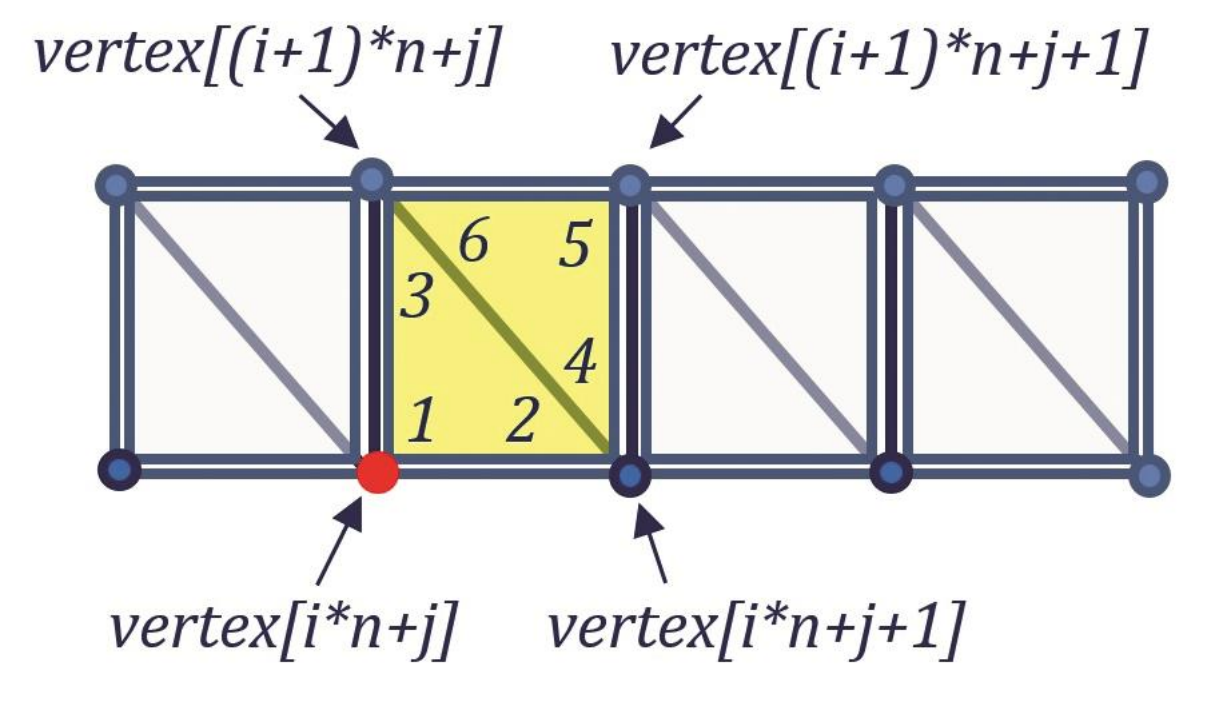
STORING OBJECT VERTICES

- Recall storing cube vertices
  - Even though each vertex participated in several triangle primitives, we stored each vertex associated with each primitive
    - Vertices end up getting stored several times
    - OK for cubes, but more complex models will require too much storage
- Instead, we can store each vertex once
  - In the case of the sphere, start with bottom horizontal slice and store in array
  - Use indices into this array to access each triangle primitive
  - End up with array of vertices and array of indices into vertex array



TRAVERSING VERTICES

- Start at the bottom of the sphere
- Traverse vertices in a circular fashion
- Build triangle primitives from each square region above and to the right



THE CODE TO CREATE A SPHERE CLASS

```
public class Sphere
{
    private int numVertices, numIndices, prec=48;
    private int[] indices;
    private Vertex3D[] vertices;

    public Sphere(int p)
    {
        prec = p;
        InitSphere();
    }

…
```

THE SPHERE CODE …

```
private void InitSphere()
{
    numVertices = (prec+1) * (prec+1);
    numIndices = prec * prec * 6;
    vertices = new Vertex3D[numVertices];
    indices = new int[numIndices];

    for (int i=0; i<numVertices; i++)
    {
        vertices[i] = new Vertex3D();
    }
```

…

THE SPHERE CODE …

```
…
        // calculate triangle vertices
        for (int i=0; i<=prec; i++)
        {
            for (int j=0; j<=prec; j++)
            {
                float y = (float)cos(toRadians(180-i*180/prec));
                float x =
                  -(float)cos(toRadians(j*360.0/prec))*(float)abs(cos(asin(y)));
                float z =
                  (float)sin(toRadians(j*360.0f/(float)(prec)))
                  *(float)abs(cos(asin(y)));
                vertices[i*(prec+1)+j].setLocation(new Point3D(x,y,z));
                vertices[i*(prec+1)+j].setS((float)j/prec);
                vertices[i*(prec+1)+j].setT((float)i/prec);
                vertices[i*(prec+1)+j].setNormal(new
                  Vector3D(vertices[i*(prec+1)+j].getLocation()));
             }
         }
…
```

THE SPHERE CODE …

```
…
        // calculate triangle indices
        for(int i=0; i<prec; i++)
        {
            for(int j=0; j<prec; j++)
            {
                indices[6*(i*prec+j)+0] = i*(prec+1)+j;
                indices[6*(i*prec+j)+1] = i*(prec+1)+j+1;
                indices[6*(i*prec+j)+2] = (i+1)*(prec+1)+j;
                indices[6*(i*prec+j)+3] = i*(prec+1)+j+1;
                indices[6*(i*prec+j)+4] = (i+1)*(prec+1)+j+1;
                indices[6*(i*prec+j)+5] = (i+1)*(prec+1)+j;
            }
         }
    }

…
```

THE SPHERE CODE …

```
…
        public int[] getIndices()
        {
            return indices;
        }

        public Vertex3D[] getVertices()
        {
            return vertices;
        }
}
```

USING THE SPHERE CLASS

- In init():

```
mySphere = new Sphere(precision);
```

- In display():

```
int numVers = mySPhere.getIndices().length;
gl.glDrawArrays(GL_TRIANGLES, 0, numVerts);
```

USING THE SPHERE CLASS

```
private void setupVertices()
{
    GL4 gl = (GL4) GLContext.getCurrentGL();

    Vertex3D[] vertices = mySphere.getVertices();
    int[] indices = mySphere.getIndices();

    float[] pvalues = new float[indices.length*3];
    float[] tvalues = new float[indices.length*2];
    float[] nvalues = new float[indices.length*3];

    for (int i=0; i<indices.length; i++)
    {
        pvalues[i*3] = (float) (vertices[indices[i]]).getX();
        pvalues[i*3+1] = (float) (vertices[indices[i]]).getY();
        pvalues[i*3+2] = (float) (vertices[indices[i]]).getZ();
        tvalues[i*2] = (float) (vertices[indices[i]]).getS();
        tvalues[i*2+1] = (float) (vertices[indices[i]]).getT();
        nvalues[i*3] = (float) (vertices[indices[i]]).getNormalX();
        nvalues[i*3+1]= (float)(vertices[indices[i]]).getNormalY();
        nvalues[i*3+2]=(float) (vertices[indices[i]]).getNormalZ();
    }
…
```

USING THE SPHERE CLASS

…

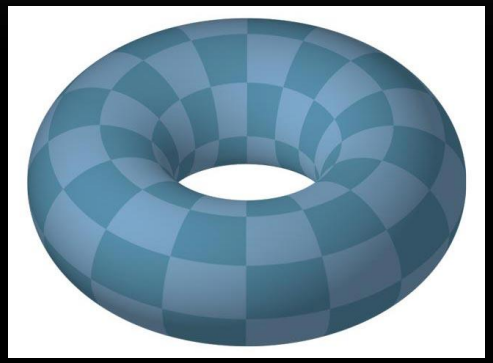
```
    gl.glGenVertexArrays(vao.length, vao, 0);
    gl.glBindVertexArray(vao[0]);
    gl.glGenBuffers(3, vbo, 0);

    gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
    FloatBuffer vertBuf = Buffers.newDirectFloatBuffer(pvalues);
    gl.glBufferData(GL_ARRAY_BUFFER, vertBuf.limit()*4, vertBuf,
                    GL_STATIC_DRAW);

    gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
    FloatBuffer texBuf = Buffers.newDirectFloatBuffer(tvalues);
    gl.glBufferData(GL_ARRAY_BUFFER, texBuf.limit()*4, texBuf,
                    GL_STATIC_DRAW);

    gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[2]);
    FloatBuffer norBuf = Buffers.newDirectFloatBuffer(nvalues);
    gl.glBufferData(GL_ARRAY_BUFFER, norBuf.limit()*4,norBuf,
                    GL_STATIC_DRAW);
    }
```

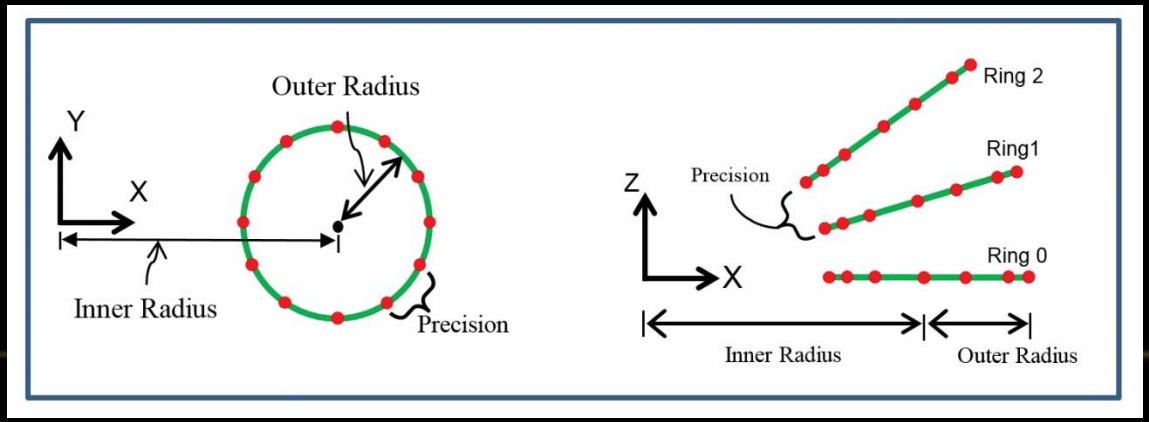
THE RESULT

Note: The previous code still stored vertices in the VBO redundantly

BUILDING A TORUS – USING OPENGL INDEXING

- Approach:
  - Define a radius about the origin (inner radius)
  - Build a circle around that radius
    - Radius of circle is outer radius
  - Rotate around the origin and build more circles
  - Generate texture coordinates and normal as we go





OPENGL INDEXING

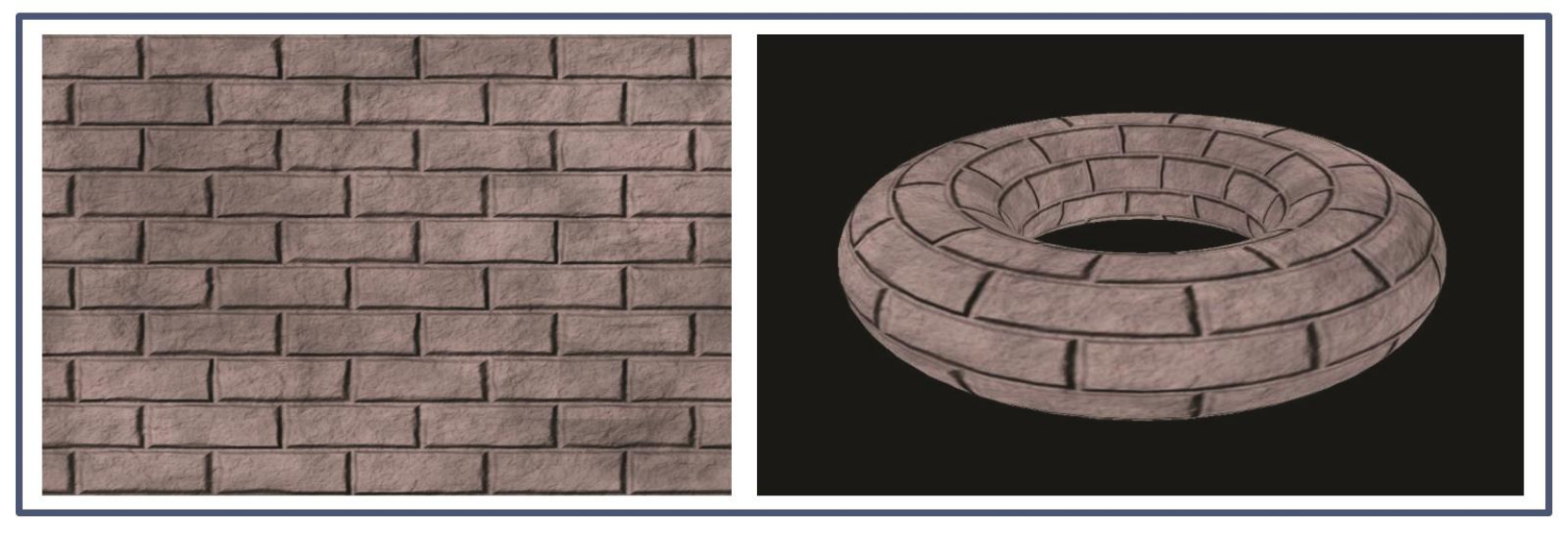
- Use an additional VBO to store indices (in setupVertices()):

```
gl.glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, vbo[3]);  // indices
IntBuffer idxBuf = Buffers.newDirectIntBuffer(indices);
gl.glBufferData(GL_ELEMENT_ARRAY_BUFFER, idxBuf.limit()*4,
                idxBuf, GL_STATIC_DRAW);
```

- In display(), use glDrawElements instead of glDrawArrays:

```
int numIndices = myTorus.getIndices().length;
gl.glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, vbo[3]);
gl.glDrawElements(GL_TRIANGLES, numIndices,
                  GL_UNSIGNED_INT, 0);
```

THE RESULT



SUMMARY

- Building a Sphere
- Building a Torus