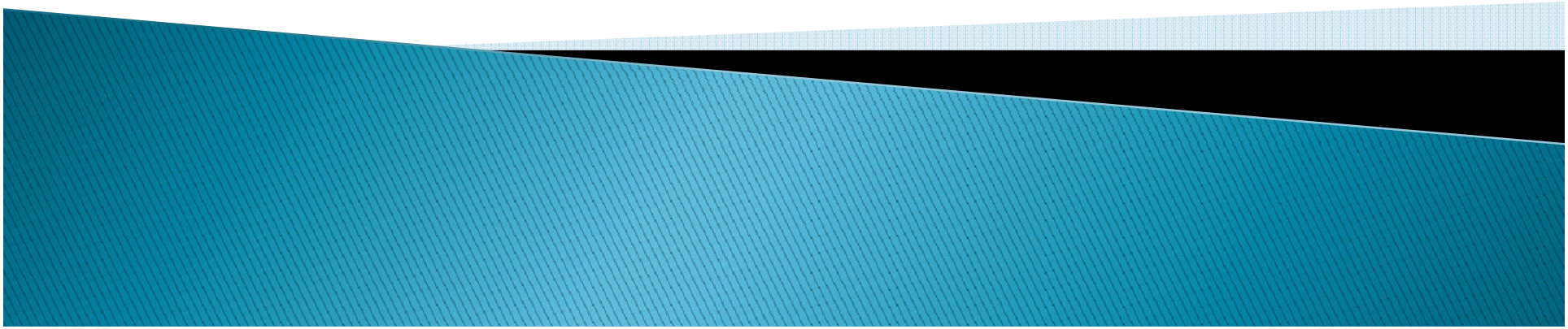
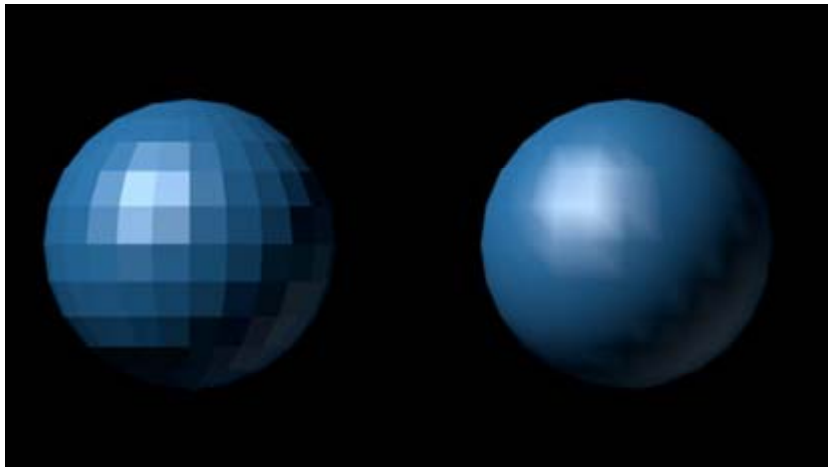
CS3241 : Let There Be Light!!!

Lab #3



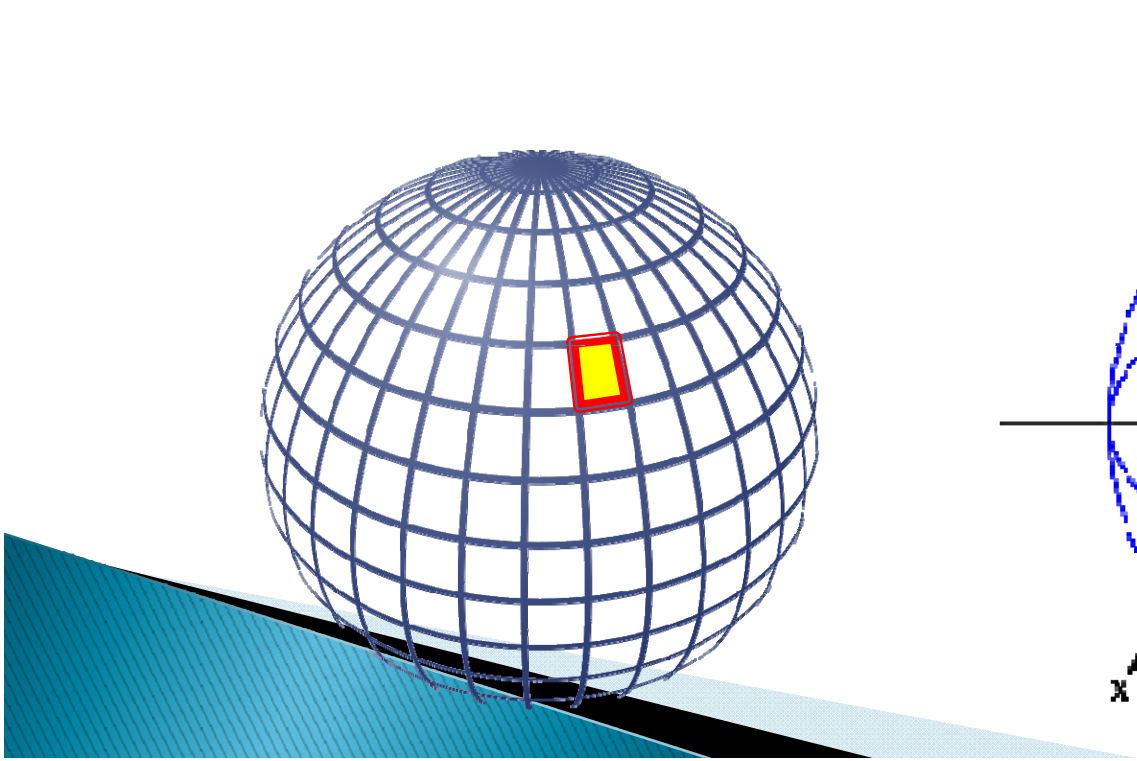
Goals

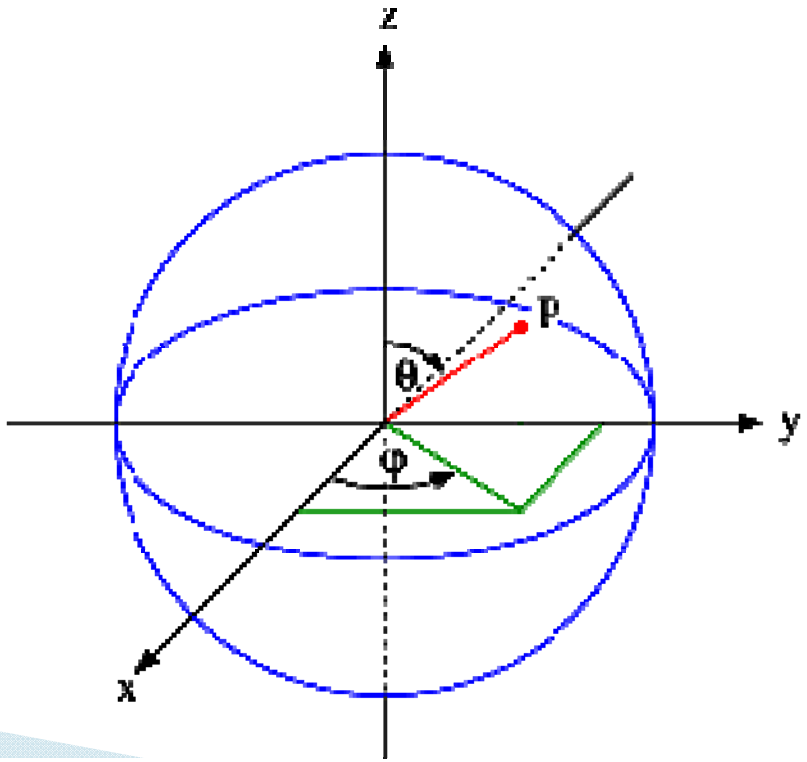
- Better understanding of flat and smooth shadings
- Making fun shapes
- Understanding of how light interacts with surface via calculation



Step 1: Draw your sphere

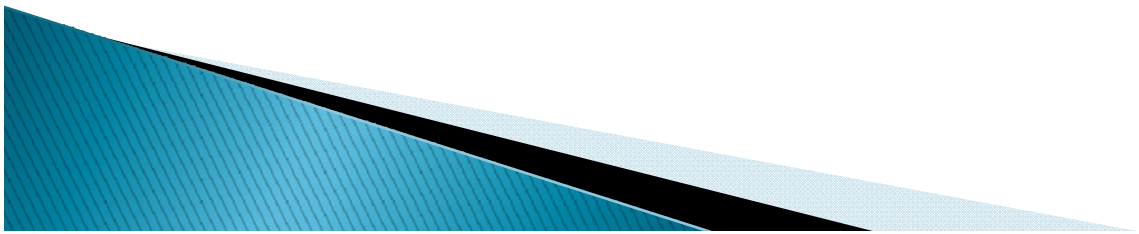
- Decompose a sphere into small polygons
  - And each polygon has a few vertices
- The coordinates of each vertex can be expressed by angle “theta” and “phi”



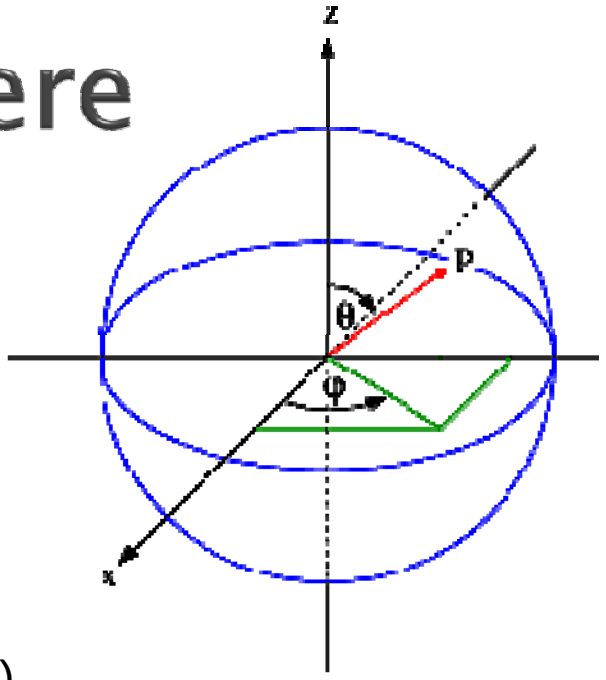


Step 1: Draw your sphere

- Go to the function
  - void drawSphere(double r)
- Look into the for-loop



Step 1: Draw your sphere

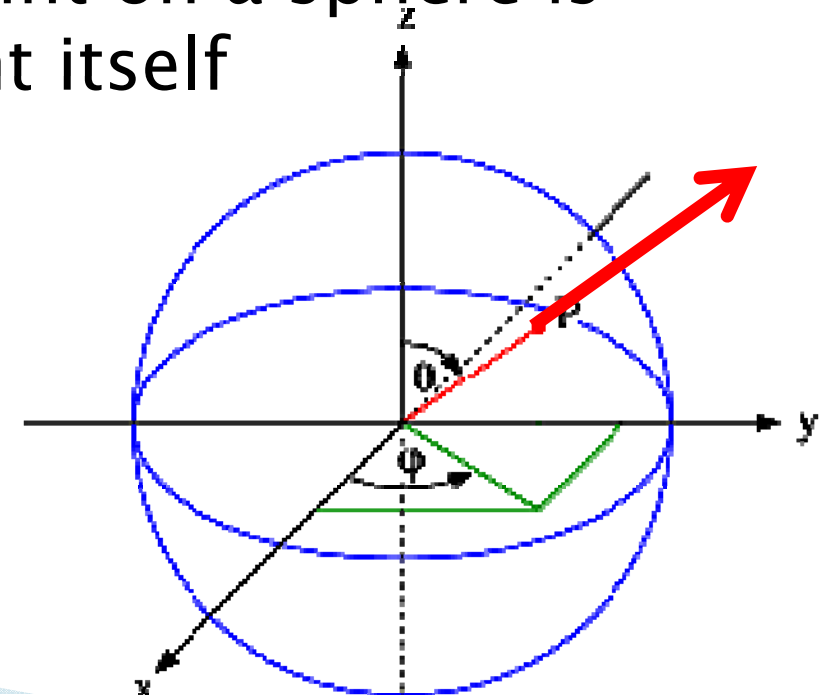


- Variable i is the movement along the latitude
- Variable j is the movement along the longtitude

```
glVertex3d(r*sin(i*M_PI/n)*cos(j*M_PI/n),
           r*cos(i*M_PI/n)*cos(j*M_PI/n),
           r*sin(j*M_PI/n));

glVertex3d(r*sin((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*sin(j*M_PI/n));

glVertex3d(r*sin((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));

glVertex3d(r*sin(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));
```

The Normal Vectores

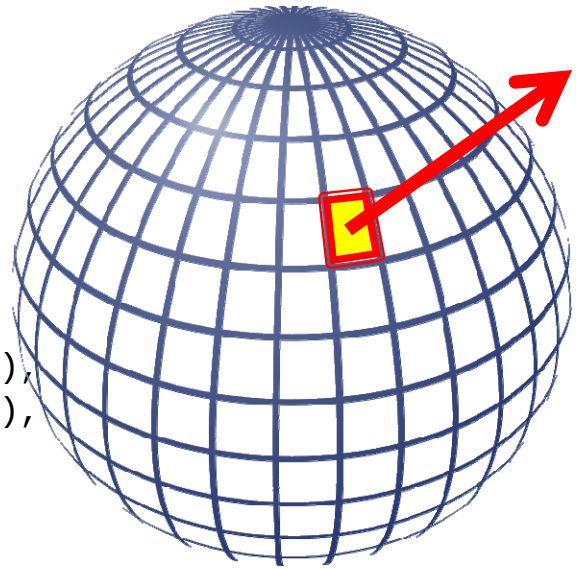
- Usually we can following the lecture slides, computing the vertex normal by the polygons
- However, the sphere is a special case that we can compute the normals in an easier manner
- The normal vector of a point on a sphere is the unit vector of the point itself



Step 2: Flat shading

- The normal of the whole polygon is the coordinate of the center of the polygon for a sphere
- Use the center of polygon to be the normal

```
glNormal3d(sin((i+0.5)*M_PI/n)*cos((j+0.5)*M_PI/n),
           cos((i+0.5)*M_PI/n)*cos((j+0.5)*M_PI/n),
           sin((j+0.5)*M_PI/n));

glVertex3d(r*sin(i*M_PI/n)*cos(j*M_PI/n),
           r*cos(i*M PI/n)*cos(j*M PI/n),
           r*sin(j*M_PI/n));

glVertex3d(r*sin((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*sin(j*M_PI/n));

glVertex3d(r*sin((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));

glVertex3d(r*sin(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));
```



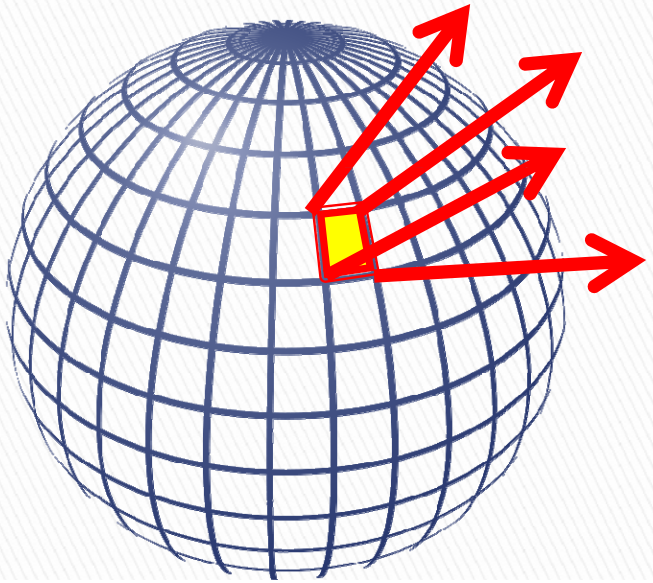
Step 3: Smooth Shading

Set different normal vectors for different vertices

```
glNormal3d(sin(i*M_PI/n)*cos(j*M_PI/n),
           cos(i*M_PI/n)*cos(j*M_PI/n),
           sin(j*M_PI/n));
glVertex3d(r*sin(i*M_PI/n)*cos(j*M_PI/n),
           r*cos(i*M_PI/n)*cos(j*M_PI/n),
           r*sin(j*M_PI/n));

glNormal3d(sin((i+1)*M_PI/n)*cos(j*M_PI/n),
           cos((i+1)*M_PI/n)*cos(j*M_PI/n),
           sin(j*M_PI/n));
glVertex3d(r*sin((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos(j*M_PI/n),
           r*sin(j*M_PI/n));
```

Let's try it all together

```
glNormal3d(sin((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           cos((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           sin((j+1)*M_PI/n));
glVertex3d(r*sin((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos((i+1)*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));

glNormal3d(sin(i*M_PI/n)*cos((j+1)*M_PI/n),
           cos(i*M_PI/n)*cos((j+1)*M_PI/n),
           sin((j+1)*M_PI/n));
glVertex3d(r*sin(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*cos(i*M_PI/n)*cos((j+1)*M_PI/n),
           r*sin((j+1)*M_PI/n));
```



Changing Lighting and Material Properties

Lighting

- In the function "setupLighting", we already set up the basic lightings and material properties for you

```
glShadeModel(GL_SMOOTH);

// Lights, material properties
GLfloat    ambientProperties[]  = {0.7f, 0.7f, 0.7f, 1.0f};
GLfloat    diffuseProperties[]  = {0.8f, 0.8f, 0.8f, 1.0f};
GLfloat    specularProperties[] = {1.0f, 1.0f, 1.0f, 1.0f};
GLfloat    lightPosition[] = {-100.0f,100.0f,100.0f,1.0f};


glLightfv( GL_LIGHT0, GL_POSITION, lightPosition);

glLightfv( GL_LIGHT0, GL_AMBIENT, ambientProperties);
glLightfv( GL_LIGHT0, GL_DIFFUSE, diffuseProperties);
glLightfv( GL_LIGHT0, GL_SPECULAR, specularProperties);
glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, 0.0);

// Default : lighting
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

k_a, k_d, k_s in the lecture notes (the last entry is "alpha" for transparency)

1.0 for NOT at infinity

You can set more then 1 light

Lighting not in effect if you don't enable it

Changing Material Properties for EACH OBJECT

- Try looking into the function drawSphere again

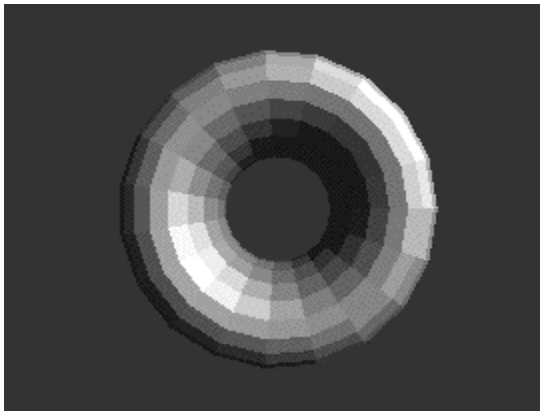
```
float mat_ambient[] = {0.8f, 0.8f, 0.2f, 1.0f};
float mat_diffuse[] = {0.1f, 0.5f, 0.8f, 1.0f};

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
```

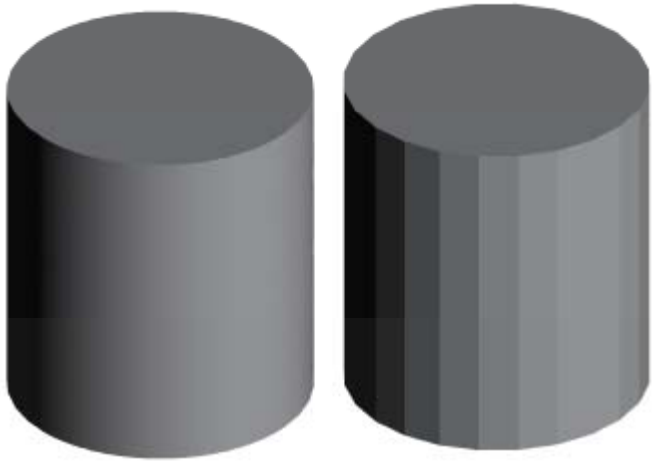
Set up the colors for different properties (ambient, diffuse, specular)

Let's try to change the ball into red color?

Step 4 : Create another shape

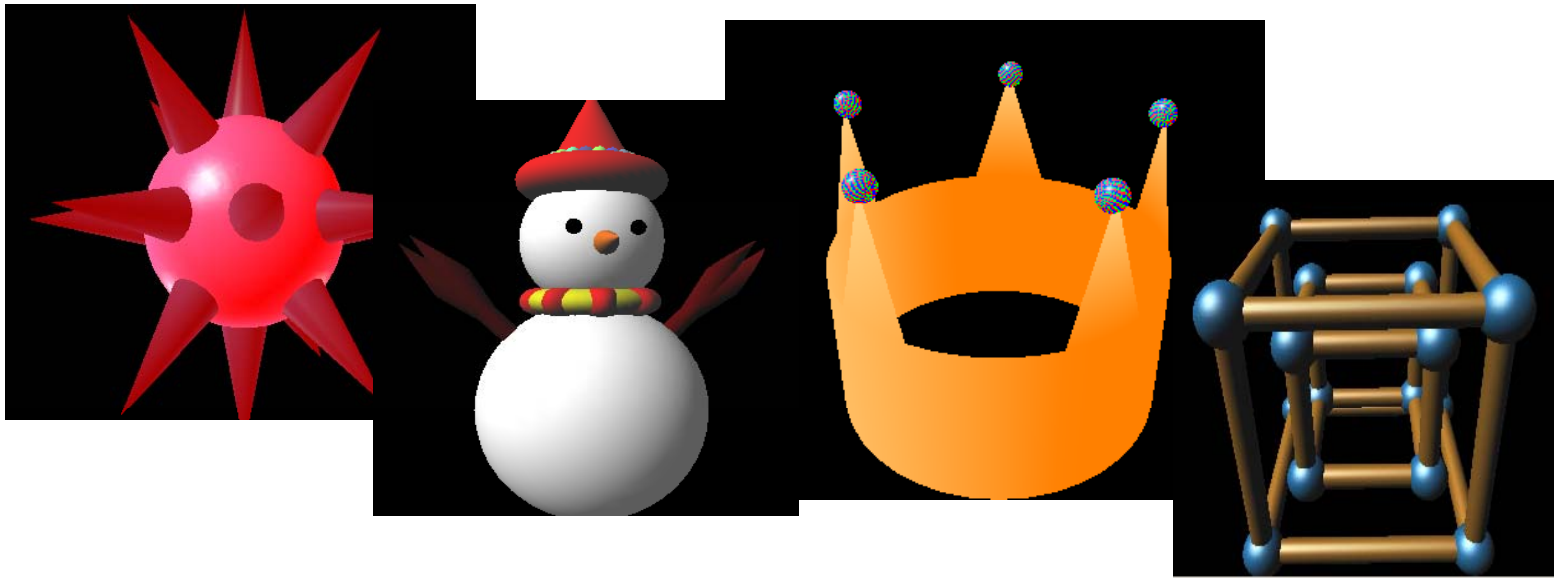
- Draw more fun shapes! Torus, Hearts, cylinders, cones, etc.





Step 5 : Composite Objects

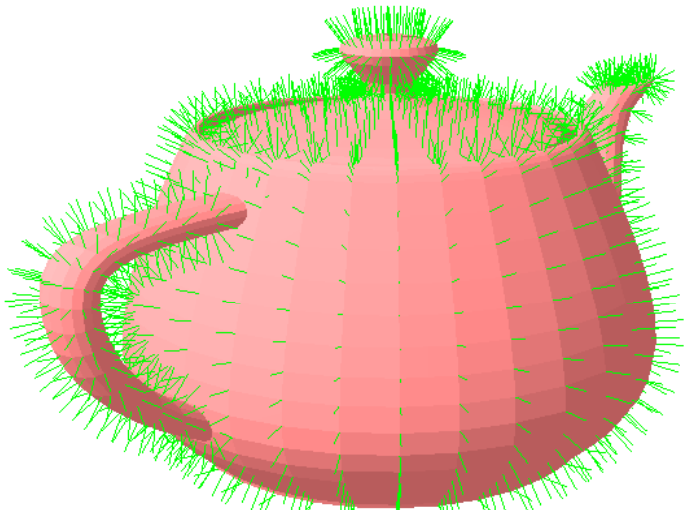
- Compose your primitive objects to form unique shapes



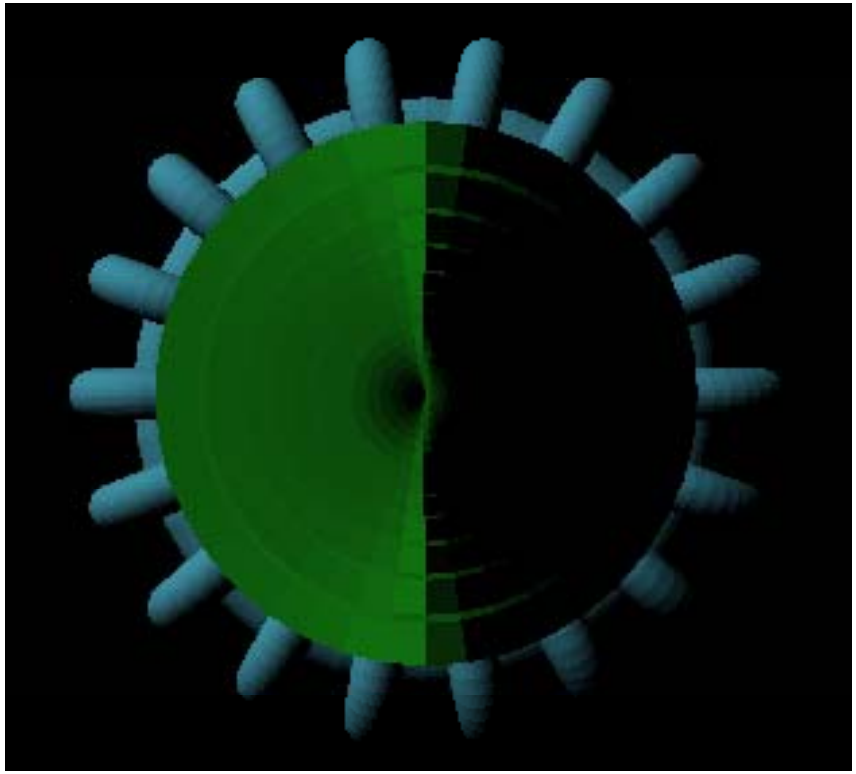
Some small tips

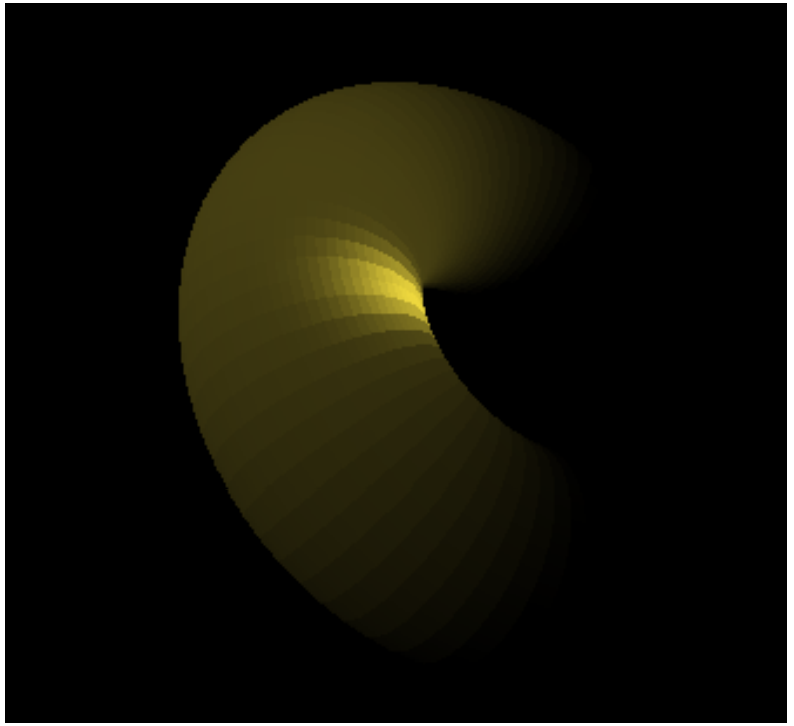
- Avoid using square roots or power functions if there are other ways to calculate.
  - These functions are really expensive and will slow down your graphic

- If you really have troubles with bad lighting, write a function to draw out your normal vectors for debugging



Examples of bad/half-lit shading





- Try to avoid these, check your normal calculations and see whether they're facing inwards or outwards