3D Graphics and 3D Modeling

Lecture 8

Outline

- 3D Graphics
- Built-in Surfaces
- 3D Modeling

3D Graphics

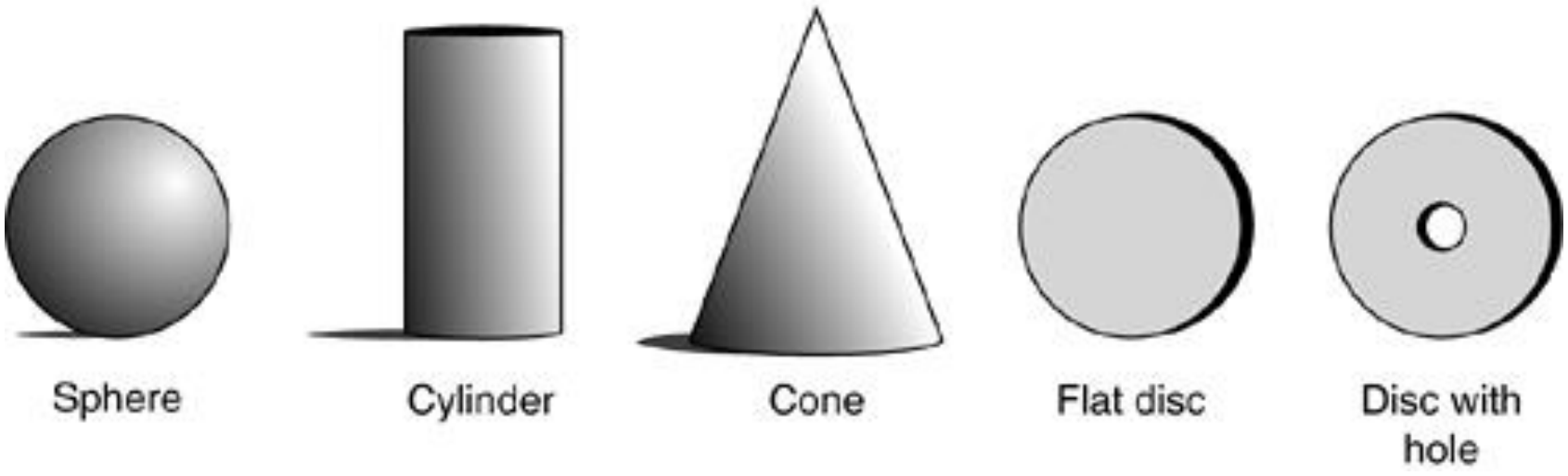
- 3D graphics is little more than a computerized version of connect-the-dots.
- Vertices are laid out in 3D space and connected by flat primitives
- Smooth curves and surfaces are approximated using flat polygons and shading tricks.
- The more polygons used, usually the more smooth and curved a surface may appear

3D Graphics

- OpenGL does provide some additional support, however, that makes the task of constructing more complex surfaces a bit easier.
- The easiest to use some GLU functions that render spheres, cylinders, cones, and flat, round disks, optionally with holes in them.
- OpenGL also provides top-notch support for complex surfaces that may be difficult to model with a simple mathematical equation: Bezier and NURB curves and surfaces

Built-in Surfaces

- The OpenGL Utility Library (GLU) contains a number of functions that render 3 quadratic surfaces: spheres, cylinders and disks.



These quadric objects can be arranged to create more complex models

Setting Quadric States

- The quadric surfaces can be drawn with some flexibility as to whether normals, texture coordinates, and so on are specified.
- Quadric functions use an object-oriented model.

To create an empty quadric object and delete it:

```
GLUquadricObj *pObj;

//Create and initialize Quadric
pObj = gluNewQuadric();

// Set Quadric rendering Parameters
// Draw Quadric surfaces

gluDeleteQuadric(pObj); //Free Quadric object
```

Quadric Draw Style

- void gluQuadricDrawStyle(
  GLUquadricObj * obj,
  GLenum drawStyle);
  - drawStyle:
    - GLU_FILL
    - GLU_LINE
    - GLU_POINT

Quadric surface normals

- void gluQuadricNormals(
  GLUquadricObj *obj,
  GLenum normals);
  - normals:
    - GLU_NONE
    - GLU_SMOOTH
    - GLU_FLAT

Specify whether the normals point out of the surface or inward

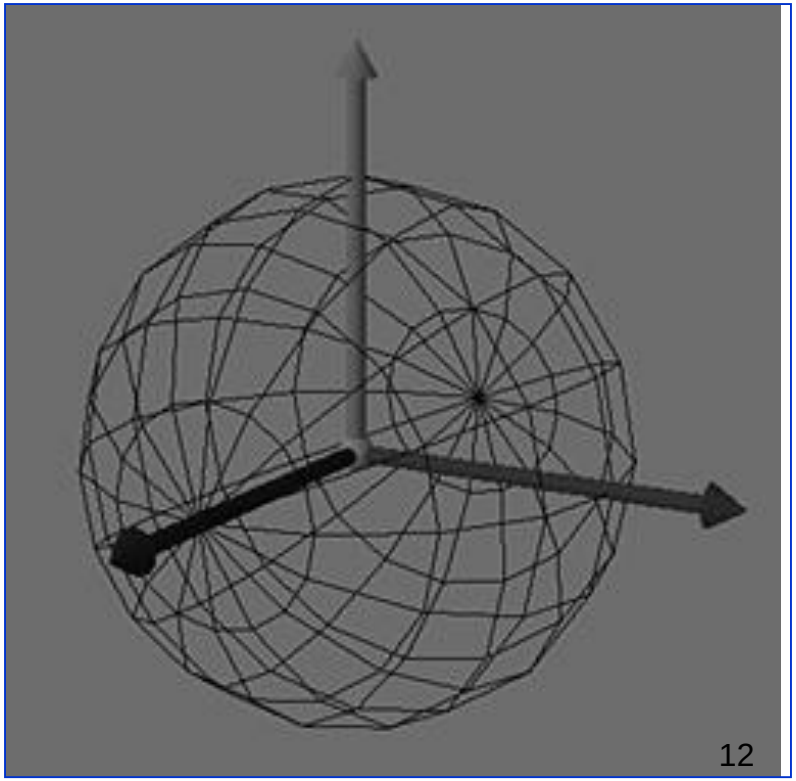
- void gluQuadricOrientation(
  GLUquadricObj * obj,
  GLenum orientation);
  - orientation:
    - GLU_OUTSIDE
    - GLU_INSIDE
  - By default, quadric surfaces are wound counterclockwise, with the front faces facing the outside of the surfaces.

Texture coordinates for quadric surfaces

- void gluQuadricTexture(
  GLUquadricObj *obj,
  GLenum textureCoords);
  - textureCoords:
    - GL_TRUE
    - GL_FALSE

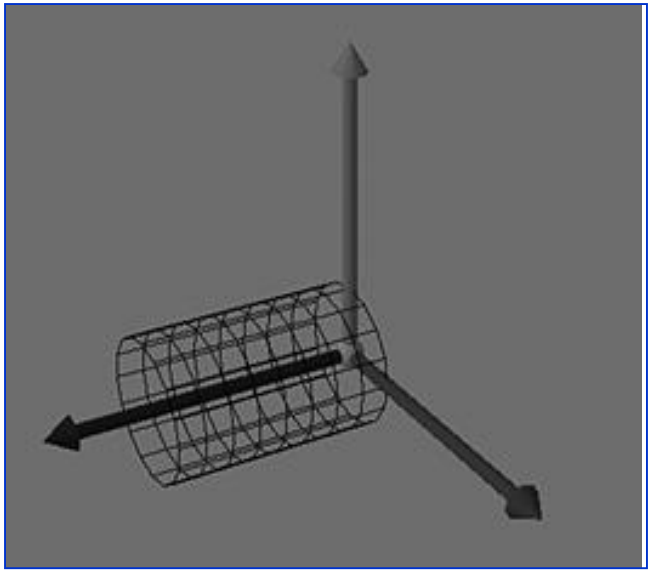
Drawing Quadrics

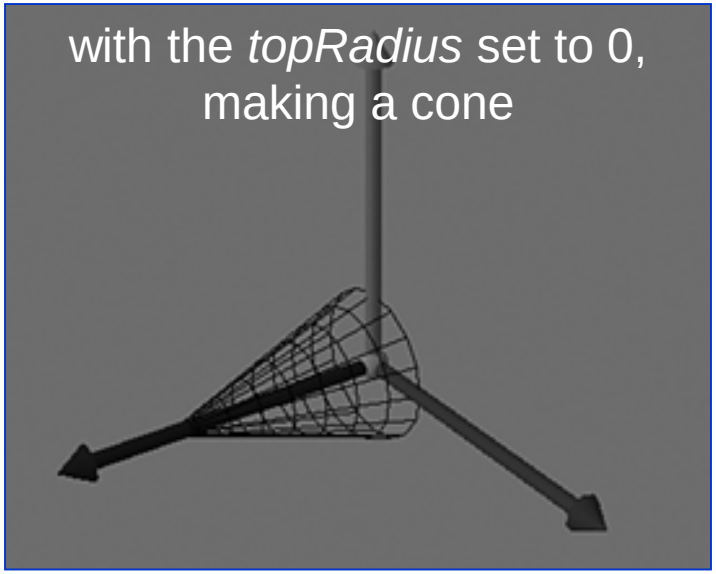
- void gluSphere(
  GLUquadricObj * obj,
  GLdouble radius,
  GLint slices,
  GLint stacks);



Drawing Quadrics

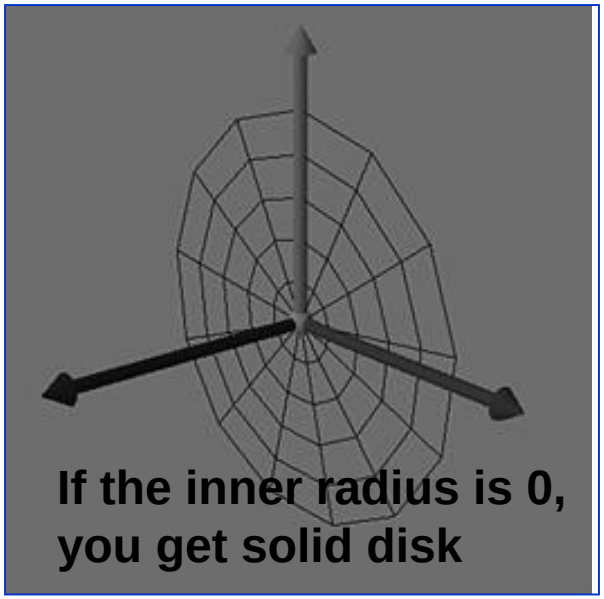
- void gluCylinder (
  GLUquadricObje *obj,
  GLdouble baseRadius,
  GLdouble topRadius,
  GLdouble height,
  GLint slices, GLint stacks);

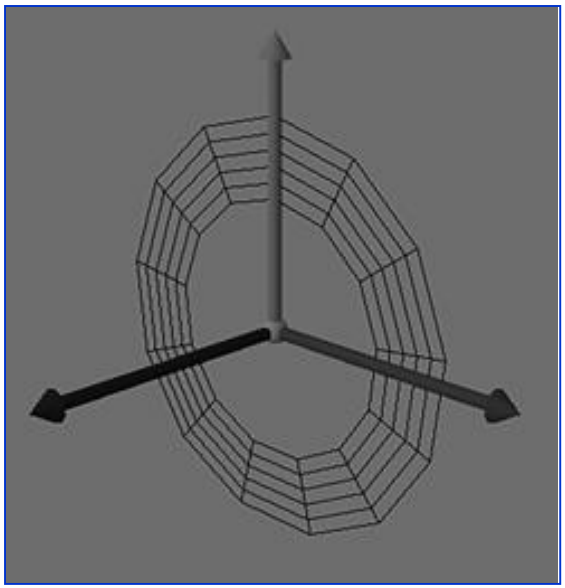




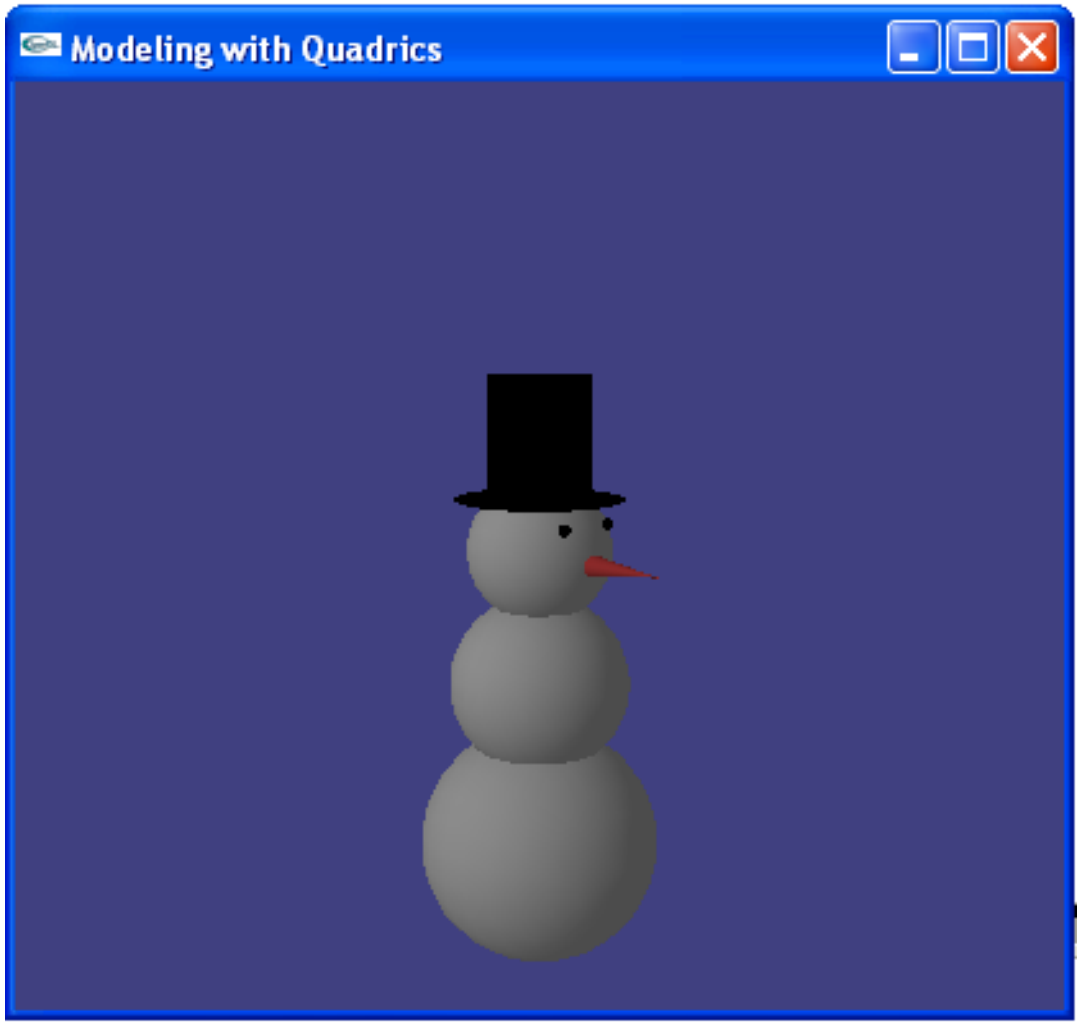
Drawing Quadrics

- void gluDisk (
  GLUquadricObj * obj,
  GLdouble innerRadius,
  GLdouble outerRadius,
  GLint slices, GLint loops)





Rendering code for the Snowman



```
void RenderScene(void)
{
GLUquadricObj *pObj; // Quadric Object
// Clear the window with current clearing color
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Save the matrix state and do the rotations
glPushMatrix();
// Move object back and do in place rotation
glTranslatef(0.0f, -1.0f, -5.0f);
glRotatef(xRot, 1.0f, 0.0f, 0.0f);
glRotatef(yRot, 0.0f, 1.0f, 0.0f);
// Draw something
pObj = gluNewQuadric();
gluQuadricNormals(pObj, GLU_SMOOTH);
```

```
// Main Body
glPushMatrix();
    glColor3f(1.0f, 1.0f, 1.0f);
    gluSphere(pObj, .40f, 26, 13); // Bottom
    glTranslatef(0.0f, .550f, 0.0f); // Mid section
    gluSphere(pObj, .3f, 26, 13);
    glTranslatef(0.0f, 0.45f, 0.0f); // Head
    gluSphere(pObj, 0.24f, 26, 13);

    // Eyes
    glColor3f(0.0f, 0.0f, 0.0f);
    glTranslatef(0.1f, 0.1f, 0.21f);
    gluSphere(pObj, 0.02f, 26, 13);
    glTranslatef(-0.2f, 0.0f, 0.0f);
    gluSphere(pObj, 0.02f, 26, 13);

    // Nose
    glColor3f(1.0f, 0.3f, 0.3f);
    glTranslatef(0.1f, -0.12f, 0.0f);
    gluCylinder(pObj, 0.04f, 0.0f, 0.3f, 26, 13);
glPopMatrix();
```

```
// Hat
glPushMatrix();
	glColor3f(0.0f, 0.0f, 0.0f);
	glTranslatef(0.0f, 1.17f, 0.0f);
	glRotatef(-90.0f, 1.0f, 0.0f, 0.0f);
	gluCylinder(pObj, 0.17f, 0.17f, 0.4f, 26, 13);

	// Hat brim
	glDisable(GL_CULL_FACE);
	gluDisk(pObj, 0.17f, 0.28f, 26, 13);
	glEnable(GL_CULL_FACE);
	glTranslatef(0.0f, 0.0f, 0.40f);
	gluDisk(pObj, 0.0f, 0.17f, 26, 13);
	glPopMatrix();
	// Restore the matrix state
glPopMatrix();

glutSwapBuffers();
}
```