11. Quadrics in OpenGL

GLU provides routines to model and render tessellated, polygonal approximations of spheres, cylinders, disks and parts of disks.

These quadric “surfaces” are described by the following general quadratic equation:

\[ a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5yz + a_6zx + a_7x + a_8y + a_9z + a_{10} = 0 \]  

(11.1)

Steps to create a quadric surface using OpenGL:

1. Create an quadric object using `gluNewQuadric()`

2. Specify the rendering attributes for the object
   - Use `gluQuadricOrientation()` to differentiate the interior from the exterior
   - Use `gluQuadricDrawStyle()` to render as points, lines or filled polygons
   - Use `gluQuadricNormal()` to specify how surface normals should be generated
   - Use `gluQuadricTexture()` to generate texture coordinates

3. If desired, register an error-handling routine

4. Invoke `gluSphere()`, `gluCylinder()`, `gluDisk()`, or `gluPartialDisk()` to render the object

5. When finished delete the quadric object with `gluDeleteQuadric()`
• **Managing Quadric Objects**

New quadric objects are created using

```c
GLUquadricObj* gluNewQuadric(void);
```

where

- The returned value is a pointer to the new quadric object
- null is returned if an object could not be created

Quadric objects are deleted using

```c
void gluDeleteQuadric(GLUquadricObj* qobj);
```

where

- `qobj` is the object to be deleted

• **Controlling Quadrics Rendering Attributes**

The drawing style of a specified object is set using

```c
void gluQuadricDrawStyle(GLUquadricObj* qobj
                        GLenum drawStyle);
```

where `drawStyle` may be

- GLU_POINTS, GLU_LINE, GLU_SILHOUETTE, or GLU_POLYGON
The direction in which generated normals point is controlled using

```c
void gluQuadricOrientation(GLUquadricObj* qobj
                           GLenum orientation);
```

where `orientation` may be
- `GLU_OUTSIDE`, vectors point in
- `GLU_INSIDE`, vectors point out
- for disks, the positive z side of the disk is considered the outside

The number of generated normals and subsequently the appearance of a quadric object are controlled using

```c
void gluQuadricNormals(GLUquadricObj* qobj
                        GLenum normals);
```

where `normals` may be
- `GLU_NONE`, object will not be lit
- `GLU_FLAT`, one normal for each facet
- `GLU_SMOOTH`, one normal for each vertex
Generation of texture coordinates is turned on and off using

```c
void gluQuadricTexture(GLUquadricObj* qobj
                      GLBoolean textureCoords);
```

where `textureCoords` may be

- GLU_TRUE, coordinates are generated
- GLU_FALSE, no coordinates generated
• Generating Quadric Surfaces

Vertices and other quadric data are generated using one of the following depending on the type of surface to be rendered

♦ Spheres

A quadric sphere is generated using

```c
void gluSphere(GLUquadricObj* qobj,
               GLdouble radius,
               GLint slices,
               GLint stacks);
```

where

- `radius` is the sphere radius
- `slices` specifies the number of lines of longitude (vertical, around the z axis)
- `stacks` specifies the number of lines of latitude (horizontal, along the z axis)
Cylinders

A quadric cylinder oriented along the z axis is generated using

```c
void gluCylinder(GLUquadricObj* qobj
    GLsizei height,
    GLsizei baseRadius,
    GLsizei topRadius,
    GLsizei slices,
    GLsizei stacks);
```

where

- `topRadius` is the radius of the cylinder at z = height (zero to create a cone)
- `baseRadius` is the radius of the cylinder at z = 0
- `slices` specifies the number of subdivisions around the z axis
- `stacks` specifies the number of subdivisions along the z axis
Disks

A disk and partial disks in the plane $z = 0$ are generated using

```c
void gluDisk(GLUquadricObj* qobj,
    GLdouble innerRadius,
    GLdouble outerRadius,
    GLint slices,
    GLint rings);
```

and

```c
void gluPartialDisk(GLUquadricObj* qobj
    GLdouble innerRadius,
    GLdouble outerRadius,
    GLint slices,  GLint rings
    GLdouble startAngle,
    GLdouble sweepAngle);
```

where

- $innerRadius$ is the radius or the inner circle (zero to get a full disk)
- $BaseRadius$ is the outer radius of the disk
- $slices$ specifies the number of subdivisions around the $z$ axis (pizza slices)
- $rings$ specifies the number of subdivisions about the $z$ axis (concentric rings)
- $startAngle$ and $sweepAngle$ are in degrees relative to the $+y$ axis
Example 11.1

Creating Quadric Objects

GLUquadricObj *qobj;

// Create 4 display lists, each with a different quadric object.
// Different draw styles and surface normal specifications are demonstrated.

startList = glGenLists( 4 );
qobj = gluNewQuadric();

 gluQuadricDrawStyle( qobj, GLU_FILL );
gluQuadricNormals( qobj, GLU_SMOOTH ); /* smooth shaded */
gNewList( startList, GL_COMPILE );
    gluSphere( qobj, 0.75, 15, 10 );
gEndList();

gluQuadricDrawStyle( qobj, GLU_FILL );
gluQuadricNormals( qobj, GLU_FLAT ); /* flat shaded */
gNewList( startList+1, GL_COMPILE );
    gluCylinder( qobj, 0.5, 0.3, 1.0, 15, 5 );
gEndList();

gluQuadricDrawStyle( qobj, GLU_LINE ); /* all polygons wireframe */
gluQuadricNormals( qobj, GLU_NONE );
gNewList( startList+2, GL_COMPILE );
    gluDisk( qobj, 0.25, 1.0, 20, 4 );
gEndList();

gluQuadricDrawStyle( qobj, GLU_SILHOUETTE ); /* boundary only */
gluQuadricNormals( qobj, GLU_NONE );
gNewList( startList+3, GL_COMPILE );
    gluPartialDisk( qobj, 0.0, 1.0, 20, 4, 0.0, 225.0 );
gEndList();

gluDeleteQuadric(qobj);