Shapes and Scenes

Geometric Modeling Techniques for Shapes

- Non-Smooth Surfaces (Fractals, Polygon Soup)
- Interactive and Editable free-form surfaces (Subdivision Splines, A-splines, NURBS)
- Shell surfaces
- Boolean Set (CSG) operations on Solids
- Physically Based Procedural Modeling (Diffusion Modeling, Particle systems, Elasto-dynamics),

Scenes

Games, Movies, Advertisements, Scientific Discovery,

- Natural and Artificial Terrains
- Simulated Environments
- Nano-worlds to Cosmo- Worlds

Curves and Surfaces Programming using OpenGL and GLU

Quadrics support in GLU

Define a quadric object.

```
GLUquadricObj*p;
p=gluNewQuadric();
```

Specify a rendering Style of Quadric. Example as a wireframe.

```
gluQuadricDrawStyle(p,GLU_LINE);
```

Example a cylinder with its length along the y-axis

```
gluCylinder(p,BASE_RADIUS,BASE_RADIUS,BASE_HEIGHT,sample_circle,sample_height)
```

```
sample_circle = number of pieces of the base
```

```
sample_height = number of height pieces
```

Bézier Curves and Surfaces

Support is available through 1D, 2D, 3D, 4D *evaluators* to compute values for the polynomials used in Bézier and NURBS.

```
glMaplf(type,u_min,u_max,stride,order,point_array)

type = 3D points, 4D points, RGBA colors, normals, indexed colors,
1D to 4D texture coordinates

u_min <= parameter u  <=  u_max

stride = number of parameter values between curve segments

order = degree of polynomial + 1

control polygon = defined by point_array
```

Example an *evaluator* for a 3D cubic Bézier curve defined over (0,1) with a stride of 3 and order 4

```
point  data[]={...}
glMaplf{GL_MAP_VERTEX_3,0.0,1.0, 3,4,data};
```

Multiple evaluators can be active at the same time, and can be used to evaluate curves, normals, colors etc at the same time

To render the Bézier Curve over (0,1) with 100 line segments

```
glEnable{GL_MAP_VERTEX_3};
glBegin(GL_LINE_STRIP)
    for(i=0; i<100; i++) glEvalCoord1f((float) i/100.0);
glEnd();
```

See example OpenGL/GLU program on pg 618 Chap11 (also pg 522, Chap 10,3rd ed.) for displaying a teapot using Bézier functions.

For lighting / shading using a NURBS surface, when additionally needs surface normals. These could be generated automatically, using

```
glEnable(GL_AUTO_NORMAL)
```

NURBS functions in GLU library

gluNewNurbsRenderer() - create a pointer to a NURBS object

gluNurbsProperty() - choose rendering values such as size of lines, polygons. Also enables a mode where the tesselated geometry can be retrieved through the callback interface

gluNurbsCallBack() - register the functions to call to retreive the tesselated geometric data or if you wish notification when an error is encountered

gluNurbsCurve() gluNurbsSurface() - to generate and render -specify control points, knot sequence, order, and/or normals, texture coordinates

Reading Assignment and News

Chapter 10, 467-568 and chapter 11 pages 600 - 622, of Recommended Text.

Please also track the News section of the Course Web Pages for the most recent Announcements related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics25/cs354/)