11/19/2020

There will be a quiz on Tuesday 11/24 during class.
It will include material covered after the midterm and before
The Texture mapping. Mainly related to E1 and shading
Including S3 material understanding concepts and the assignments material.
Solution to S3 will be provided.

Extra Credit deadline on 12/03/2020 at 5:00 pm

1) **Instructor evaluation proof of submission**
    a. **Email from IT**
    b. **Take a screenshot of the confirmation**
    c. **You can still offer qualitative answers**
2) **Class evaluation**

Two alternatives choose the one you prefer. Can do both but get extra credit only for 1.
Please comply with the requirements for 2 (in case you choose 2)

Curves and Surfaces

Leaving the "Flatland"

Geometry primitives

Flat primitives - Points, lines, polygons (triangles, quads)
Used to represent any object regardless of complexity
|➔ May need to use a huge (inhibitory large) set of primitives
Used for plotting of functions (curves and surfaces) assumed
that the function is given
glVertex

Raster primitives (texture mapping)

Place images on top of Geometry
glTexture

Generating "arbitrary" curves and surfaces

The function is not know (given) cannot be accurately specified
Our primitives are polygons with vertices given through a parametric equation approximating (and or interpolating) functions using glTexcoord().
We **Model** (trial and error) a curve e.g., for Computer Aided design of objects with curves (e.g. the curvature of a car body)
Rather than location (x, y, z) it is specified using the u, v parameters of the parametric representation of functions

The screen is a raster (discrete) screen the output is pixels (points)

35:2 function representation for plotting

Modeling curves (assuming that a curve to be modeled is given e.g., made by an artist)

Modeling ➔ Interpolate or approximate the curve 35:4

The programmer supplies **control points** and requirements to OGL (interpolate, approximate, or use the derivative of curve for modeling).

Open GL is using these control points to provide a parametric function that complies with the requirement. The programmer might divide the curve into segments and provide (4) control points per segment.

"Good" Model 35:16 division to segments
Implicit requirement
    Continuity of segments
    Continuity of derivatives
Explicit requirements35:5

Function representation 35: 6 to 14 please review

In general, we have freedom in selecting the way to model out of several families of functions.
We model using parametric representation (freedom WRT the family)

Parametric p(u), q(u) u is between 0 and 1. P(1) == q(0) for continuity.

Our preference is to use Polynomial as the family of functions used for modeling.
The programmer supply control points and conditions OGL identifies a polynomial parametric function that complies with the control points and their requirements.

Why polynomials? See 35:17

Slide 16 – general representation using polynomials

C's are coefficients N, M, L are degrees
Make N=M=L ➔ We (OGL) have to find 3(n+1) coefficients.
35:17 no continuity in derivatives.

Restricting to cubic polynomials 35:18 35:19
|➔ From your 4-control point OGL has to figure 12 parameters.

Due to symmetry WRT to X, Y, Z we now concentrate on the X axis.

In 35:18 looking for:

$x(u) = c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3$
$y(u) = c_{0y} * u^0 + c_{1y} * u^1 + c_{2y} * u^{2+} c_{3y} * u^3$
$z(u) = c_{0z} * u^0 + c_{1z} * u^1 + c_{2z} * u^{2+} c_{3z} * u^3$

Need to identify all the 'C' coefficients.

The conditions are a mixture of continuity requirements at the joint points and conditions for fitting the data to the curve.

Curves of Interest
- Interpolating Curves
  - interpolate all the control points
- Hermite Curves
  - Interpolate the end-points and satisfy a constraint on derivatives
- Bezier curve
  - Interpolate the end-points and satisfy a constraint on approximating the derivatives
- B-Splines
  - Extension of Bezier

(NURBS)

OGL has only functions for generating Bezier curves.

Bezier Curves
4-control point P_0, P_1, P_2, P_3 (per segments)
P_0 and P_3 should interpolate the curve P_1 and P_2 are used to approximate derivatives
Nice features in next slides.

How to interpolate and get -B-splines. Using model view transformations 38:15 from set of slides 38:15
Every curve is a Bezier

In Open GL

39:4 we have to define evaluators.
Setting them up

38:5 6

Define the control points (array)
Glmap1f()
Type GL_MAP_VERTEX_3
U_min ==0
Umax == 1
Stride how are the points organized X0, Y0 Z0, X1, Y1, Z1 (stride of 3)
Or X0, X1, X2, X3, Y0, Y1, Y2, Y3, Z0.. Z3 (stride of 1)

Order 4 for cubic
Pointer to the array

The result ➔ OGL identifies the Bezier polynomial suiting the control points

$x(u) = c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3$
$y(u) = c_{0y} * u^0 + c_{1y} * u^1 + c_{2y} * u^{2+} c_{3y} * u^3$
$z(u) = c_{0z} * u^0 + c_{1z} * u^1 + c_{2z} * u^{2+} c_{3z} * u^3$

To generate the curve generate vertices defined by values of U.

Two ways to define these vertices.

glEvalCoord1f(u) you supply the u (slide 8), does not have to be uniformly divided between 0 to 1

A shortcut in 39:9 is to replace the two functions in slide 8

For uniform division you can use 39:9

By glMapGrid(100, 0.0, 1.0);

glEvalMesh1(GL_LINE, 0, 99);

Example for Non uniform

```
glBegin(GL_LINE_STRIP)
        glEvalCoord1f(0);
        glEvalCoord1f( 1/3);   / equivalent to glVertex specifying the x, y z, via a value of u O
        OGL figures XYZ.

        glEvalCoord1f( 7/8);
        glEvalCoord1f( 1);

glEnd();
```