12/01//2020

1. I am extending the deadline for this assignment to December 8, 2020 @11:55pm and designating it as an extra credit assignment. With the following policy:

   a. I will either count your best 4 ‘S’-assignments (out of the 5 ‘S’-assignments)
   b. or give you 2% extra credit towards your final grade; whichever is higher.
   c. The questions in this assignment (S5) are highly relevant to questions in the final.

On Thursday 12/03/2020 I will administrate a quiz in class. It will cover material related to Shading, and Shadowing theory and application (as in assignment S3).

   d. Given this the final will only cover material related to buffers, texture mapping, curves and surfaces,
      additionally, it will include material on perspective projection and hidden surface removal to be covered this week.

The Final will be a take home exam with open material including, notes, written material, and material available online.

   a. It will cover material listed in section 2.a above.
   b. It will be posted on TRACS on the early morning hours of December 10, 2020. And will be due on December 10, 2020 @11:55 pm.
   c. You will have to provide an extremely detailed description of your solution, showing that you completely understand your own solution.

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 (written) 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)

<u>Curves and Surfaces</u>

Last time concentrated on the general principals and OGL implementation.
Today, complete the OGL part.

Next, we will study how to use the conditions (control points etc.) to develop
The parametric equations.

Given 4 control points find the cubic polynomial that interpolate (passing through)
The points) ➔ 4 equations in 4 unknown per axis,

In a way we find a general set of equations and we solve them

The parametric equations sought

Based on the control points and restrictions (conditions) we obtain:

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

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

We know that the curve goes through the points
|→ The points must comply with the equations above

P0, P1, P2, P3

For x(u) P0x,P1x, P2x, P3x, comply with the equations

$x(u)$ @p0 = $(c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3)$ @P0
$x(u)$ @p1 = $(c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3)$ @P1
$x(u)$ @p2 = $(c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3)$ @P2
$x(u)$ @p3 = $(c_{0x} * u^0 + c_{1x} * u^1 + c_{2x} * u^{2+} c_{3x} * u^3)$ @P3

Assume that the points are evenly placed between u=0 and u=1

P0 @u=0
P2 @u =1/3
P3 @ u =2/3
P3 @ u= 1

Substitute ➔ 4 equations in 4 unknowns.

$x(u)$ @p0 = (c_{0x}) @P0
$x(u)$ @p1 = $(c_{0x} * + c_{1x}*(1/3)^1 + c_{2x}*(1/3)^{2+} c_{3x}*(1/3)^3)$ @P1
$x(u)$ @p2 = $(c_{0x} + c_{1x}*(2/3)^1 + c_{2x}*(2/3)^{2+} c_{3x}*(2/3)^3)$ @P2
$x(u)$ @p3 = $(c_{0x} + c_{1x} + c_{2x}{}^+ c_{3x})$ @P3

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

How will you solve these equations to find the C's

Substitutions?

Use matrix form.

P=Ac slide 5

Slide 5.

Find A^{-1}
Multiply by the 2 sides

$A^{-1}p= A^{-1}*A*C = C$

$C = A^{-1*}P$

A does not depend on C ➔ only one Matrix A

What would glMap1f() do given the control points?

Answer
$C = A^{-1}*P$ to get the C's
 |➔ x(u), y(u), z(U)

After OGL is getting the coefficients C

What is the next thing
glEvalCoord1f(u)

given a specific U

We want to place U into the following equation

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

so for **glEvalCoord1f(0.9)**
OGL will calculate
$x(u) = (c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)$ @u=0.9

glMapGrid and glEvalMesh are function that call glMap1f() and glEvalCoor()

Point-cloud to Mesh
Kinect ➔ Point cloud ➔ Mesh
In 6 we have the Mi = A^(-1).

Skipping 8 - 13

Hermite proposes that the 4 conditions for finding the 4 equations would be
Condition 1 P0 = p(0)
Condition 2 is about the derivative of the curve at 0 p'(0) (P1)

Condition 3 is about the derivative of the curve at 1 p'(1) (P2)

Condition 4 P3 = P(1)

Now we substitute these conditions into the 4 equations.

$x(u)$ @p(0) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)$
$x(u)$ @p'(0) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)'$ (' mean obtain the derivative and set u to 0)
$x(u)$ @p'(1) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)'$ (' mean obtain the derivative and set u to 1)
$x(u)$ @p(1) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)$

Stays the same
$x(u)$ @p(0) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)$
$x(u)$ @p(1) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2+} c_{3x}*u^3)$

What is $x'(u)$

$x'(u) = (c_{1x} + 2*c_{2x}*u^2 + 3*c_{3x}*u^3)$

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

$x'(u)$ @u=0 = (c_{1x})
$x'(u)$ u=1 = $(c_{1x} + 2*c_{2x} + 3*c_{3x})$

See slide 14 (4 equations in 4 unknowns

Slide 16, matrix and inverse matrix

A problem is that generally the derivatives are not known and hard to estimate.

Bezier is providing a way to approximate the derivatives.

36-3 a vector form of the above.

Back to open GL

Find (solve) the C's
OGL is solving these C's for the Bezier Curve

How do we figure 4 unknowns (C's)?

Consider an interpolating curve.
We supply 4 points P_0, P_1, P_2, P_3.

We want to get a curve that "goes through" these 4 points.
The curve is expected to be represented by a cubic polynomial.
Assume that it starts at P_0 and ends at P_3.

OGL has to figure the 4 C's for x (12 for x, y, and z in 3-D)

What can we do to get these 4 unknowns

We have to figure out 4 equations , upon solving the equations we obtain the C's

In OGL the first part is:

Given 4 points P_0, P_1, P_2, P_3
OGL figures the 4 equation in 4 unknowns
Solves and get the 3 parametric functions.

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

For the Bezier curve.

Done using the function glMap1f()

Next OpenGL plots the 3 functions

```
Internal loop
    $u$ goes from 0 to 1 in increments of $du$ ($du$ is based on your specifications)
        glBegin(shape)
            Produce a vertex at $[x(u), y(u), z(u)),1]^T$ (using glEvalCoord1f())
        glEnd()
```

The shape can be any GL 2.x shape (point, line, line strip, quad strip).

To Produce a vertex at [x(u), y(u), z(u)),1]^T (e.g. @ u=0.25)
Using glEvalCoord1f((float) 0.25);
Instead of glVertex() for the actual x, y, and z for u=0.25

Each call to glEvalCoord1f((float) d); with a given value d is instantiating a vertex

```
glBegin(shape)
glEvalCoord1f( (float) 0);
glEvalCoord1f( (float) 0.25);
glEvalCoord1f( (float) 0.4);
glEvalCoord1f( (float) 0.6);
glEvalCoord1f( (float) 1.0);
glEnd()
```

Not necessarily uniformly spaced between 0 and 1

Instead of the two OGL instructions
We can generate a plot where the point are uniformly spaced using

```
glMapGrid(100, 0.0, 1.0);
glEvalMesh1(GL_LINE, 0, 99);
```

OGL can only produce the 3 functions

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

For a Bezier curve

How to interpolate or B-Spline

38:15 left most: what you “get” from the 4 points in GL
How to get interpolation ho to get a B-spline
|➔ use model view transformations.

38:13 how to get the Transformation
38:14 provides the transformation

For S5 you have to load the matrices to the Model View Matrix to get the right effect.