Interpolation and Approximation of functions

Dinesh K. Pai

Partly from Textbook Chapter 9

Today

- Reminders:
  - Assignment 4 is out
  - Midterm 2 coming soon (March 21)
  - Final exam: APR 26, 03:30 PM

- Wrap up texture mapping for now

- New topic: Interpolation and approximation
  - Foundation of a lot of topics (Chapter 9), including geometric modeling (Chapter 22), animation (Chapter 23) and dealing with images (Chapters 16-18)
Assignment 4

- Available now, due March 25 (by popular demand)
- However: please make sure you do at least the first 3 parts before March 21 (midterm)
  - These parts will help you understand the topics covered in class and textbook Chapter 15, which will be on the midterm
- Use the extra time after midterm for “Creative Licence”. Texture mapping is particularly great for this… many of you will find this a lot of fun!

C³ Review: Texture mapping

- Which texture mapping technique would you use to put the smiley face in the scene?
  a) Basic texturing
  b) Projector texture mapping
  c) Environment cube maps
  d) All of the above
  e) None of the above
C³ Review: Texture mapping

- In which of the following will the resulting surface color appear to be static when the eye (camera) is moving?
  a) Environment cube mapping  
  b) Projector texture mapping  
  c) Basic texture mapping  
  d) B & C  
  e) All of the above

C³ Review: Texture mapping

- Can you use texture mapping along with other lighting/shading techniques?
  a) Yes. Texture mapping should be done in the vertex shader, and lighting/shading should be done in the fragment shader so they don’t overlap.  
  b) Yes. The lighting/shading effect can be added to the texture map on each fragment.  
  c) No. The normal from the texture is in conflict with the normal from the model  
  d) No. The final colour of each pixel can only be determined by one technique.
Geometry of Projector Textures

Projector texture mapping

- The slide projector is modeled using 4 by 4, modelview and projection matrices, $M_s$ and $P_s$

\[
\begin{bmatrix}
x_t w_t \\
y_t w_t \\
\_ \\
w_t
\end{bmatrix} = P_s M_s \begin{bmatrix}
x_o \\
y_o \\
z_o \\
1
\end{bmatrix}
\]
Projector texture mapping

- With the texture coordinates defined as
  \[ x_t = \frac{x_t W_t}{w_t} \quad \text{and} \quad y_t = \frac{y_t W_t}{w_t} \]

- To color a point on a triangle with object coordinates \([x_o, y_o, z_o, 1]^T\), we fetch the texture data stored at location \([x_t, y_t]^T\).

Projector vertex shader

```glsl
#version 330

uniform mat4 uModelViewMatrix;
uniform mat4 uProjMatrix;
uniform mat4 uSProjMatrix;
uniform mat4 uSModelViewMatrix;
in vec4 aVertex;
out vec4 vTexCoord;

void main(){
    vTexCoord = uSProjMatrix * uSModelViewMatrix * aVertex;
    gl_Position = uProjMatrix * uModelViewMatrix * aVertex;
}
```

Vertex shader generates texture coordinates!
But not normalized
Projector texture mapping

- Projector fragment shader

```cpp
#version 330

uniform sampler2D vTexUnit0;

in vec4 aTexCoord;
out vec4 fragColor;

void main(){
    vec2 tex2;
    tex2.x = aTexCoord.x/aTexCoord.w;
    tex2.y = aTexCoord.y/aTexCoord.w;
    vec4 texColor0 = texture2D(vTexUnit0, tex2);
    fragColor = texColor0;
}
```
Interpolation & Approximation

**Motivation:**

Digital/Discrete representation of continuous (smooth) functions (in any dimension)

We will focus on 1D

- easier to explain
- generalization to higher dim are conceptually straightforward
- lots of applications
  - e.g. functions of time

\[ f(t) \]

```
Computer animation
```

```
Audio
```

```
key frame animation
```

```
Intermediate frames done by computer
```

```
Sampling
```

**Key Questions**

Discrete \( \rightarrow \) Reconstruction \( \rightarrow \) Continuous

```
Interpolation is one way to do this
```

```
```
8 Constant (interpolation)

Obvious limitations
not continuous
may want smoother shapes

Note: this is what your monitor does!
when you display an image.

8 Linear Interpolation (piecewise)

Focus on this interval

\[ c(t) = a \cdot t + b \]

Pixel boundaries

\[ C_0 = a \cdot 0 + b \Rightarrow b = C_0 \]
\[ C_1 = a \cdot 1 + C_0 \Rightarrow a = C_1 - C_0 \]
\[ c(t) = (C_1 - C_0) \cdot t + C_0 \]
# Key step. Reorder this
\[ C(t) = c_0 (1-t) + c_1 t \]

- Works separately data
- from the "blending weights" that are independent of data. Just depend on the type of interpolation.

\[ C_i \] are called "control values"

In general, \( C_i \) can be any dimensional vector

E.g., \( C_i \) could be the coordinates of a 3D point

In common usage, \( C_i \) are called "control points"

Next class: Generalize to higher smoothness

Bézier curves, etc.