Lecture 9:
Shading 2 (Texture Mapping)
Announcements

• Lingqi’s office hour
  - Moved to Wednesday at 2PM (only for this week)

• Assignment 4 is out
  - Implementing Blinn-Phong Reflectance Model!

• Midterm next Thursday (Feb 13)
  - 72 minutes in class
  - 60 multiple choice questions for 60 points
  - Scantrons will be provided, but bring your own pencil
  - A (short) sample will be provided by this Friday
Last Lecture

• Shading 1
  - Blinn-Phong reflectance model
    - Diffuse
    - Specular
    - Ambient
  - At a specific shading point
Today

• Finishing up shading 1
  - Shading frequencies
  - Graphics pipeline

• Shading 2 (Texture Mapping)
  - Texture mapping
  - Barycentric coordinates
Shading Frequencies
Shading Frequencies

What caused the shading difference?
Shade each triangle (flat shading)

**Flat shading**

- Triangle face is flat — one normal vector
- Not good for smooth surfaces
Shade each vertex (Gouraud shading)

**Gouraud** shading

- **Interpolate** colors from vertices across triangle
- Each vertex has a normal vector (how?)
Shade each pixel (Phong shading)

**Phong shading**

- Interpolate normal vectors across each triangle
- Compute full shading model at each pixel
  - Not the Blinn-Phong Reflectance Model
Shading Frequency: Face, Vertex or Pixel

Shading freq.: Face, Vertex or Pixel

Shading type: Flat, Gouraud, Phong

Num Vertices

Defining Per-Vertex Normal Vectors

Best to get vertex normals from the underlying geometry
  • e.g. consider a sphere

Otherwise have to infer vertex normals from triangle faces
  • Simple scheme: average surrounding face normals

\[ N_v = \frac{\sum_i N_i}{\| \sum_i N_i \|} \]
Defining Per-Pixel Normal Vectors

Barycentric interpolation (introducing soon) of vertex normals

Don’t forget to normalize the interpolated directions
Graphics (Rendering) Pipeline
The Graphics Pipeline:

1. **Application**
   - **Vertex Processing**
     - **Vertex Stream**
     - **Triangle Processing**
     - **Rasterization**
     - **Fragment Processing**
     - **Framebuffer Operations**

2. **Display**
   - **Input: vertices in 3D space**
   - **Vertices positioned in screen space**
   - **Triangles positioned in screen space**
   - **Fragments (one per covered sample)**
   - **Shaded fragments**
   - **Output: image (pixels)**

Slide courtesy of Prof. Ren Ng, UC Berkeley
Graphics Pipeline

Model, View, Projection transforms

Slide courtesy of Prof. Ren Ng, UC Berkeley
Graphics Pipeline

- Application
  - Vertex Processing
    - Vertex Stream
  - Triangle Processing
    - Triangle Stream
  - Rasterization
    - Fragment Stream
  - Fragment Processing
    - Shaded Fragments
  - Framebuffer Operations
    - Display

Sampling triangle coverage

Slide courtesy of Prof. Ren Ng, UC Berkeley
Rasterization Pipeline

Application

Vertex Processing
  - Vertex Stream

Triangle Processing
  - Triangle Stream

Rasterization
  - Fragment Stream

Fragment Processing
  - Shaded Fragments

Framebuffer Operations

Display

Z-Buffer Visibility Tests

Slide courtesy of Prof. Ren Ng, UC Berkeley
Graphics Pipeline

- **Application**
  - **Vertex Processing**
    - Vertex Stream
  - **Triangle Processing**
    - Triangle Stream
  - **Rasterization**
    - Fragment Stream
  - **Fragment Processing**
    - Shaded Fragments
  - **Framebuffer Operations**

- **Display**

**Shading**

- Ambient + Diffuse = Blinn-Phong Reflectance Model
- + Specular

Slide courtesy of Prof. Ren Ng, UC Berkeley
Graphics Pipeline

Slide courtesy of Prof. Ren Ng, UC Berkeley
Shader Programs

• Program vertex and fragment processing stages
• Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```glsl
uniform sampler2D myTexture;
uniform vec3 lightDir;
varying vec2 uv;
varying vec3 norm;

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);
  kd *= clamp(dot(-lightDir, norm), 0.0, 1.0);
  gl_FragColor = vec4(kd, 1.0);
}
```

• Shader function executes once per fragment.
• Outputs color of surface at the current fragment’s screen sample position.
• This shader performs a texture lookup to obtain the surface’s material color at this point, then performs a diffuse lighting calculation.
Shader Programs

• Program vertex and fragment processing stages
• Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```glsl
uniform sampler2D myTexture; // program parameter
uniform vec3 lightDir; // program parameter
varying vec2 uv; // per fragment value (interp. by rasterizer)
varying vec3 norm; // per fragment value (interp. by rasterizer)

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv); // material color from texture
  kd *= clamp(dot(-lightDir, norm), 0.0, 1.0); // Lambertian shading model
  gl_FragColor = vec4(kd, 1.0); // output fragment color
}
```
Snail Shader Program

Inigo Quilez
Snail Shader Program

Goal: Highly Complex 3D Scenes in Realtime

- 100’s of thousands to millions of triangles in a scene
- Complex vertex and fragment shader computations
- High resolution (2-4 megapixel + supersampling)
- 30-60 frames per second (even higher for VR)
Graphics Pipeline Implementation: GPUs

Specialized processors for executing graphics pipeline computations

Discrete GPU Card
(NVIDIA GeForce Titan X)

Integrated GPU:
(Part of Intel CPU die)

Slide courtesy of Prof. Ren Ng, UC Berkeley
Modern GPUs offer ~2-4 Tera-FLOPs of performance for executing vertex and fragment shader programs.
Texture Mapping
Different Colors at Different Places?

L_d = k_d * (I / r^2) * (n dot l)

Pattern on ball
Wood grain on floor
Surfaces are 2D

Surface lives in 3D world space

Every 3D surface point also has a place where it goes in the 2D image (texture).
Texture Applied to Surface

Rendering without texture  Rendering with texture  Texture

Each triangle “copies” a piece of the texture image to the surface.

Slide courtesy of Prof. Ren Ng, UC Berkeley
Visualization of Texture Coordinates

Each triangle vertex is assigned a texture coordinate \((u,v)\)
Texture Applied to Surface

Rendered result

Triangle vertices in texture space

Slide courtesy of Prof. Ren Ng, UC Berkeley
Textures applied to surfaces
Visualization of texture coordinates
Textures can be used multiple times!

example textures used / tiled
Thank you!