CMSC427
Finishing basic 3D rendering

Credit: slides 9+ from Prof. Zwicker
Quick ideas

- What we don’t see: culling 3D polygons
  - Backface culling
  - Clipping to frustrum or viewport
  - Z-buffer
- Texture mapping
  - Image plus texture coordinates
When is a triangle visible? It is ...
Culling polygons

- When is a triangle visible? It is ...
  - Facing the camera
  - Within the camera frustum or viewport
  - In front of other triangles

- **Terminology:**
  - Facing camera: *Backface culling*
  - Within viewport: *Clipping*
  - In front: *Z-buffering*
Backface culling

- Discard polygons facing away from camera
- How compute?
Backface culling

- Discard polygons facing away from camera

- How compute?
  - Angle between normal and view direction < 90
  - So N • VD > 0
  - Do not need to normalize

- Convention is to wind front face CCW so right hand rule faces out

- OpenGL has flag to cull back, front or neither
• Discard polygons facing away from camera

• How compute?
  • Angle between normal and view direction < 90
  • So $N \cdot VD > 0$
  • Do not need to normalize

• Convention is to wind front face CCW so right hand rule rule faces out

• OpenGL has flag to cull back, front or neither
Clipping

• To frustrum (in 3D)

• To viewport (in 2D)

• Note: triangle clipped can become quad
Z-buffering


• Store “depth” at each pixel
  • Store $1/w$ because we compute it for rasterization already

• Depth test
  • During rasterization, compare stored value to new value
  • Update pixel only if new $1/w$ value is larger

```
setpixel(int x, int y, color c, float w)
if((1/w)>zbuffer(x,y)) then
  zbuffer(x,y) = (1/w)
  color(x,y) = c
```

• In graphics hardware, z-buffer is dedicated memory reserved for GPU (graphics memory)

• Depth test is performed by GPU
Z-buffer
• Basic shading – constant material objects

• Basic shading plus texture mapping – color varies over object

• How do?
Texture mapping – quick version

- Basic shading – constant material objects
- Basic shading plus texture mapping – color varies over object
- How do?
Texture mapping – texture coordinates

- Each vertex mapped to location in image
- Location interpolated inside polygon/triangle
Texture mapping – can be complicated…

Models and texture maps from Poser, a product for creating and rendering human characters.
Texture mapping – can also be simple

• Cube

• Cylinder
Textures – in Processing

• Load image
  \[
  \text{Pimage tex = loadImage("berlin-1.jpg");}
  \]

• Set texture image
  \[
  \text{texture(tex);}\]

• Give texture coordinates per vertex (last two)
  \[
  \text{vertex(-1, -1, 1, 0, 0);}\]

• Texture coordinates can be in image coordinates (0 to w, 0 to h) or in normalized coordinates (0 to 1, 0 to 1)

• Examples: TextureCube and TextureCylinder
• For polygon mesh vertices need:
• Location x,y,z
• Normal nx,ny,nz
• Texture coordinates u,v

• For cylinder?
Implications for OpenGL

• Backface
  • OpenGL lets you turn it off and on, and set front facing winding direction

• Clipping
  • Built into rasterization stage and fixed

• Z-buffering
  • OpenGL lets you turn it off and on
  • A consideration in setting near and far plane (too far apart, you get precision errors in z)

• Texture mapping
  • Add to meshes texture coordinates and texture buffers
• Backface
  • OpenGL lets you turn it off and on, and set frant facing winding direction

• Clipping
  • Built into rasterization stage and fixed

• Z-buffering
  • OpenGL lets you turn it off and on
  • A consideration in setting near and far plane (too far apart, you get precision errors in z)

• Texture mapping
  • Add to meshes texture coordinates and texture buffers