CSE 167: Introduction to Computer Graphics
Lecture #14: Occlusion Detection

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2020
Announcements

- **Tomorrow at 1pm:**
  - Discussion Project 3

- **Sunday, November 22^{nd} at 11:59pm:**
  - Homework Project 3 due

- **Tip: Making of Horizon Zero Dawn**
Occlusion

- At each pixel, we need to determine which triangle is visible
Painter’s Algorithm

- Paint from back to front
- Need to sort geometry according to depth
- Every new pixel always paints over previous pixel in frame buffer
- May need to split triangles if they intersect

- Intuitive, but slow algorithm
- Still used today to render translucent geometry
Z-Buffering

- Z-buffer stores depth (z-) value for each pixel
- Z-buffer is dedicated memory in GPU

Algorithm:
- Create z-buffer with as many entries as pixels in render window
- Initialize z-buffer with farthest z value
- During rasterization, compare stored value to new value
- Update pixel only if new value is smaller

```c
setpixel(int x, int y, color c, float z)
if(z < zbuffer(x,y)) then
  { zbuffer(x,y) = z; color(x,y) = c }
```

- Depth test is performed by GPU ➔ very fast
# Z-Buffer Example

<table>
<thead>
<tr>
<th>Z-Buffer Example</th>
<th>+</th>
<th>=</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Z-Buffer Example" /></td>
<td><img src="image" alt="Z-Buffer Example" /></td>
<td><img src="image" alt="Z-Buffer Example" /></td>
</tr>
</tbody>
</table>
Displaying the Z-Buffer

- Interpret z-buffer values as luminance values
- `gl_FragCoord` in fragment shader contains depth value
- Output this depth value as a color:

  ```cpp
  void main() { FragColor = vec4(vec3(gl_FragCoord.z), 1.0); }
  ```
In OpenGL applications:

- Ask for a depth buffer when you create your GLFW window.
  - `glfwOpenWindow(512, 512, 8, 8, 8, 0, 16, 0, GLFW_WINDOW)`
- Place a call to `glEnable(GL_DEPTH_TEST)` in your program's initialization routine.
- Set `zNear` and `zFar` clipping planes (`glm::perspective(fovy, aspect, zNear, zFar)`) to optimize depth buffer precision: near plane as far away as possible, far plane as close as possible without cutting into scene.
- Add `GL_DEPTH_BUFFER_BIT` parameter to `glClear`:
  - `glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);`
- Z-buffer is non-linear: uses smaller depth bins in foreground for greater depth resolution near viewer.
Z-Buffer Fighting

- Problem: polygons close together don’t get rendered correctly. Errors change with camera perspective → flicker
- Cause: differently colored fragments from different polygons being rasterized to same pixel and depth → not clear which is in front
- Solutions:
  - Move surfaces farther apart, so that fragments rasterize into different depth bins
  - Bring near and far planes closer together
  - Use a higher precision depth buffer. Note that OpenGL often defaults to 16 bit even if your graphics card supports 24 bit or 32 bit depth buffers
Translucent Geometry

- Need to depth sort translucent geometry and render with Painter’s Algorithm (back to front)
- Problem: incorrect blending with cyclically overlapping geometry

Solutions:

- Back to front rendering of translucent geometry (Painter’s Algorithm), after rendering opaque geometry
  - Binning: every piece of geometry is put in either the opaque or the translucent bin
- Theoretically: need to store multiple depth and color values per pixel (not practical in real-time graphics)