Hidden Surface Removal

which polygons are visible, which are hidden?
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

paint distant objects before near objects
sort polygons in a scene by depth and draw in that order
- still draws invisible parts
uses “depth ordering”
- Example: note parts of meadow are nearer than distant trees - but ordering is based on occlusion
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

**problem:**
triangle intersection

who’s in front of whom?
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

problem:
occlusion cycle

also, sorting primitives by depth is slow
both of these problems can be resolved by cutting triangles
Use a **z-buffer** for hidden surface removal

test depth on a pixel by pixel basis

red drawn last

<table>
<thead>
<tr>
<th>without z-buffer</th>
<th>with z-buffer</th>
</tr>
</thead>
</table>

- assume both spheres of the same size, red drawn last
Use a *z-buffer* for hidden surface removal

at each pixel, record distance to the closest object that has been drawn in a *depth buffer*
Use a **z-buffer** for hidden surface removal

- Each fragment must carry a depth
- Usually used fixed precision depth buffers - can get errors due to roundoff
Use a z-buffer for hidden surface removal

- fragment has z value and color value
- compare z value to old z value at that pixel
- if new value is nearer replace both color value and z value

http://www.beyond3d.com/content/articles/41/
Backface culling: another way to eliminate hidden geometry

this is only okay for closed surfaces
Hidden Surface Removal in OpenGL

```c
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

glEnable(GL_DEPTH_TEST);

glEnable(GL_CULL_FACE);
```

For a perspective transformation, there is more precision in the depth buffer for z-values closer to the near plane.
Transformation Matrices
<whiteboard>