Hidden Surfaces and z-Buffer

CS 417 Lecture 22

Hidden surface elimination

- At each pixel we want to see only the closest thing
  - physical basis: light does not pass through most things
- This was a major topic in graphics some time ago
  - but now brute force has won out for the object-order case

Clip coordinates

- Transformation sequence
  - refine a bit to expose
  - one more coordinate system
  - projection was a 3x4 going
  - direct to 2D projective space
  - now separate into projection
  - and homogeneous division

Clip coordinates

- In clip coordinates all visibility questions are reduced to the orthographic case
  - view direction is the z axis
  - still in homogeneous coordinates, though
**Back face culling**

- For closed shapes you will never see the inside
  - therefore only draw surfaces that face the camera
  - implement by checking $\mathbf{n} \cdot \mathbf{v}$

**Painter’s algorithm**

- Simplest way to do hidden surfaces
- Draw from back to front, use overwriting in framebuffer
 Painter's algorithm

- Simplest way to do hidden surfaces
- Draw from back to front, use overwriting in framebuffer

---

 Painter's algorithm

- Simplest way to do hidden surfaces
- Draw from back to front, use overwriting in framebuffer

---

 Painter's algorithm

- Simplest way to do hidden surfaces
- Draw from back to front, use overwriting in framebuffer

---

 Painter's algorithm

- Amounts to a topological sort of the graph of occlusions
  - that is, an edge from A to B means A sometimes occludes B
  - any sort is valid
    - ABCDEF
    - BADCFE
  - if there are cycles there is no sort

---

 Painter's algorithm

- Amounts to a topological sort of the graph of occlusions
  - that is, an edge from A to B means A sometimes occludes B
  - any sort is valid
    - ABCDEF
    - BADCFE
  - if there are cycles there is no sort
**Painter’s algorithm**

- Useful when a valid order is easy to come by
- Compatible with alpha blending

**The z-buffer**

- In many (most) applications maintaining a z sort is too expensive
  - changes all the time as the view changes
  - many data structures exist, but complex
- Solution: draw in any order, keep track of closest
  - allocate extra channel per pixel to keep track of closest depth so far
  - when drawing, compare object’s depth to current closest depth and discard if greater
  - this works just like any other compositing operation

**The z-buffer**

- another example of a memory-intensive brute force approach that works and has become the standard

**Precision in z buffer**

- The precision is distributed between the near and far clipping planes
  - this is why these planes have to exist
  - also why you can’t always just set them to very small and very large distances
- Most systems actually store and compare homogeneous z (that is, z/w)
  - avoids division at this stage
  - concentrates precision near the view point