Hidden Surfaces

Dr. Scott Schaefer
Hidden Surfaces
Hidden Surfaces
Hidden Surfaces
Backface Culling
Backface Culling

view direction v
Backface Culling
Backface Culling

\[ n \cdot v < 0, \text{ draw polygon} \]
Backface Culling

\[ n \cdot v \geq 0, \text{ cull polygon} \]
Backface Culling
Backface Culling

counter clock-wise orientation, draw polygon
Backface Culling

clock-wise orientation, cull polygon
Backface Culling

- Advantages
  - Improves rendering speed by removing roughly half of polygons from scan conversion

- Disadvantages
  - Assumes closed surface with consistently oriented polygons
  - NOT a true hidden surface algorithm!!!
Backface Culling

- Is this all we have to do?
Backface Culling

- Is this all we have to do? No!
  - Can still have 2 (or more) front faces that map to the same screen pixel
Backface Culling

- Is this all we have to do? No!
  - Can still have 2 (or more) front faces that map to the same screen pixel
  - Which actually gets drawn?
Painter’s Algorithm

- Sort polygons according to distance from viewer
- Draw from back to front

- How do we sort polygons?
Painter’s Example

Sort by depth:
Green rect
Red circle
Blue tri
Painter’s Algorithm
Painter’s Algorithm

- Sometimes there is NO ordering that produces correct results!!!
Painter’s Algorithm

1. Sort all objects’ $z_{\text{min}}$ and $z_{\text{max}}$
Painter’s Algorithm

1. Sort all objects’ $z_{\text{min}}$ and $z_{\text{max}}$

2. If an object is uninterrupted (its $z_{\text{min}}$ and $z_{\text{max}}$ are adjacent in the sorted list), it is fine
Painter’s Algorithm

1. Sort all objects’ $z_{\text{min}}$ and $z_{\text{max}}$
2. If an object is uninterrupted (its $z_{\text{min}}$ and $z_{\text{max}}$ are adjacent in the sorted list), it is fine
3. If 2 objects DO overlap
   3.1 Check if they overlap in x
      - If not, they are fine
   3.2 Check if they overlap in y
      - If not, they are fine
      - If yes, need to split one
Painter’s Algorithm

- The splitting step is the tough one
  - Need to find a plane to split one polygon by so that each new polygon is entirely in front of or entirely behind the other
  - Polygons may actually intersect, so then need to split each polygon by the other
Painter’s Algorithm

- The splitting step is the tough one
  - Need to find a plane to split one polygon by so that each new polygon is entirely in front of or entirely behind the other
  - Polygons may actually intersect, so then need to split each polygon by the other
- After splitting, you can resort the list and should be fine
Painter’s Algorithm-Summary

- **Advantages**
  - Simple algorithm for ordering polygons

- **Disadvantages**
  - Sorting criteria difficult to produce
  - Redraws same pixel many times
  - Sorting can also be expensive
Depth ("Z") Buffer

- Simple modification to scan-conversion
- Maintain a separate buffer storing the closest "z" value for each pixel
- Only draw pixel if depth value is closer than stored "z" value
  - Update buffer with closest depth value
Depth ("Z") Buffer

- Advantages
  - Simple to implement
  - Allows for a streaming approach to polygon drawing

- Disadvantages
  - Requires extra storage space
  - Still lots of overdraw
Binary Space Partitioning Trees

- BSP tree: organize all of space (hence *partition*) into a binary tree
  - *Preprocess*: overlay a binary tree on objects in the scene
  - *Runtime*: correctly traversing this tree enumerates objects from back to front
  - Idea: divide space recursively into half-spaces by choosing *splitting planes*
    - Splitting planes can be arbitrarily oriented
BSP Trees: Objects
BSP Trees: Objects
BSP Trees: Objects

Put front objects in the left branch
BSP Trees: Objects

Put front objects in the left branch
BSP Trees: Objects

Put front objects in the left branch
BSP Trees: Objects

Put front objects in the left branch
BSP Trees: Objects

Put front objects in the left branch

When to stop the recursion?
Object Splitting

- No bunnies were harmed in my example
- But what if a splitting plane passes through an object?
  - Split the object; give half to each node:
  - Worst case: can create up to $O(n^3)$ objects!
Building a BSP Tree

- Choose a splitting polygon
- Sort all other polygons as
  - Front
  - Behind
  - Crossing
  - On
- Add “front” polygons to front child, “behind” to back child
- Split “crossing” polygons with infinite plane
- Add “on” polygons to root
- Recur
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree
Building a BSP Tree

Diagram of a BSP Tree structure with nodes labeled 1, 2, 3, and 4, and sub-regions 7-2, 5-2, 7-1, 5-1.
Rendering with a BSP Tree

- If eye is in front of plane
  - Draw “back” polygons
  - Draw “on” polygons
  - Draw “front” polygons
- If eye is behind plane
  - Draw “front” polygons
  - Draw “on” polygons
  - Draw “back” polygons
- Else eye is on plane
  - Draw “front” polygons
  - Draw “back” polygons
BSP Trees: Objects

Correctly traversing this tree enumerates objects from back to front

Traversal order?
BSP Trees: Objects

Correctly traversing this tree enumerates objects from back to front

Traversal order:
8 -> 9 -> 7 -> 6 -> 5 -> 3 -> 4 -> 2 -> 1
Building a BSP Tree

Traversal order:
Building a BSP Tree

Traversal order:
6->(5-2)->(7-2)->3->(5-1)->4->(7-1)->2->1
Building a BSP Tree

Traversing order:
Building a BSP Tree

Traversal order:
1->2->(7-1)->4->(5-1)->3->(7-2)->(5-2)->6
Building a BSP Tree

Traversal order?
Rendering with a BSP Tree

- Advantages
  - No depth comparisons needed
  - Polygons split and ordered automatically

- Disadvantages
  - Computationally intense preprocess stage restricts algorithm to static scenes
  - Splitting increases polygon count
  - Redraws same pixel many times
  - Choosing splitting plane not an exact science
Improved BSP Rendering

- Take advantage of view direction to cull away polygons behind viewer
Improved BSP Rendering

- Take advantage of view direction to cull away polygons behind viewer
Improved BSP Rendering

- Take advantage of view direction to cull away polygons behind viewer
OpenGL and Hidden Surfaces

```gl
glEnable(GL_DEPTH_TEST);
glEnable(GL_CULL_FACE);

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glCullFace ( GL_BACK );
```
Scan Line Algorithm

- Assume for each line of screen, we have scan-lines for all polygons intersecting that line
- For each polygon, keep track of extents of scan line
- Whenever the x-extents of two scan lines overlap, determine ordering of two polygons
Scan Line Algorithm
Scan Line Algorithm
Scan Line Algorithm
Scan Line Algorithm

- Advantages
  - Takes advantage of coherence resulting in fast algorithm
  - Does not require as much storage as depth buffer

- Disadvantages
  - More complex algorithm
  - Requires all polygons sent to GPU before drawing