Visibility

- What is visible?
  - Which objects are visible?
  - Which pixels/fragments to render?
- Why check for visibility?
  - Efficiency
  - Correctness
  - Disambiguation

The Double Eagle Tanker: 4GB of data, 82 M Triangles
From: http://www.cs.unc.edu/~geom/hardware/#Vis

Simple question

- The art gallery problem:
  Given a planar art gallery, what is the minimum number of guards that need to be placed at the corners (but inside the gallery) so that every part of the gallery is visible to at least one guard.

Visibility

- The Image Space problem formulation
  for (each pixel in the rendered image)
  1. determine the object closest to the viewer that is intercepted by the projector (ray) through the pixel;
  2. draw the pixel in the appropriate color;
- Worst case complexity: \(np\)
  \(n\) : number of objects, \(p\) : number of pixels

Visibility

- The Object Space problem formulation
  for (every object in the world)
  1. determine those parts of the object whose view is obstructed by other parts of itself or any other object;
  2. draw those parts in the appropriate color;
- Worst case complexity: \(n^2\)
  \(n\) : number of objects
Visibility

• Types of visibility computation we have seen:
  - Clipping – 2D and 3D
  - View-frustum clipping/culling
  - Backface culling

Backface Culling

Simple Idea:
Discard surface patches that face away from the camera.

Backface Culling is not enough

Backface Culling does not remove all occluded patches (it is a conservative algorithm as are many visibility algorithms) – the example shown here is a case of self-occlusion.

Z-Buffer and Scan Conversion

• Initialize the z-buffer to the max Z value.
• glClear, glDepthRange

Z-Buffer and Scan Conversion

The example shown here is a case of self-occlusion.
Visibility

- Z-Buffer Algorithm
  - Advantages:
    - Simple, accurate (modulo non-linear z mapping).
    - Independent of order of drawing polygons.
  - Disadvantages:
    - Memory (not an issue these days).
    - Wasted computation when overwriting distant points.
  - Complexity:
    - Space: \( O(nm \cdot b) \) – \( nm \) pixels, \( b \) bytes precision per pixel.
    - Time: \( O(nm \cdot k) \) – \( nm \) pixels, \( k \) polygons.
  
  Space: \( \text{depth may be the same (this is very unlikely).} \)

Visibility

- Z-Buffer Algorithm
  - Initialize
    - \( zbuf[I, J] = \text{MAX\_DEPTH} \)
    - \( cbuf[I, J] = \text{BACKGROUND\_COLOR} \)
  - For each scan converted polygon:
    - Find pseudodepth, \( z \), of polygon at pixel \((x, y)\) with color \( c \).
    - \( (z = zbuf[I, J]; \; \text{if } z < zbuf[I, J], zbuf[I, J] = z; \; cbuf[I, J] = c) \)

Visibility

- Z-Buffer Algorithm and Scan Conversion
  - Construct the active edge list, \( AEL \), for every scanline.
  - Interpolate the pseudodepth for each active edge.
  - Note: The color at a pixel is also interpolated along the scanline like the pseudodepth is.
Visibility

- Z-Buffer Algorithm and Scan Conversion
  - Compute the active edge list.
  - Interpolate the pseudodepth for each active edge.

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.

Visibility

- Painter’s Algorithm
  - Sort polygons in increasing order of depth.
  - Draw the sorted list of polygons from back to front, i.e., from greatest depth to lesser depths.
  - What happens when a polygon has vertices at different depths?
  - Sort according to depth of farthest vertex.
Visibility

- Binary Space Partitioning (BSP) Trees

  - Observe the correct order of drawing polygons as the eye moves.

  - If e and T₁ are on the same side of T₂:
    - Draw T₁ and then draw T₂.
  - If e and T₂ are on different sides of T₁:
    - Draw T₂ and then draw T₁.

- BSP Tree is an efficient data structure for quickly determining the inside/outside relation between polygons and the camera position.

  - Two Phases
    - Preprocessing: BSP Tree construction (done once for a given scene)
    - Rendering: BSP Tree traversal (done whenever the eye position changes)
Visibility

- BSP Tree construction

1. BSP Tree traversal

- If e is outside (or in front of) a face i:
  - Draw everything behind i, draw everything in front of i

- If e is inside a (or behind) face i:
  - Draw everything in front of i, draw everything behind i

What is the traversal order now?