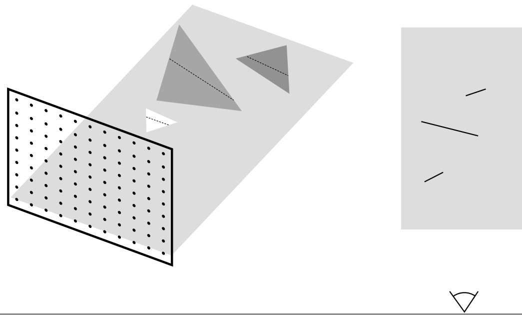
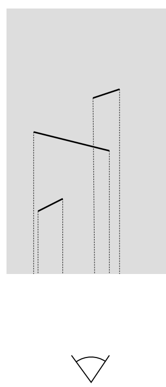
VSA: Scanline Alg

- scanline at a time
- reduce dimension: 3D->2D



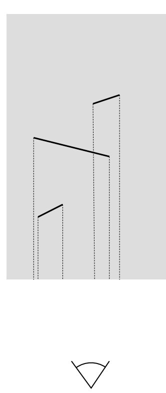
Scanline Alg

- rely on depth & scan-line coherence
- keep list of active edges
- sort in x
- do visibility test at each vertex
- can combine with z-buffer



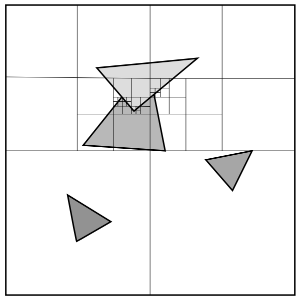
Scanline Alg

- Advantages:
  - no memory for frame buffer
  - fast for small numbers of polygons (originally used in flight simulators)
- Disadvantages
  - memory for polygon edges
  - image space



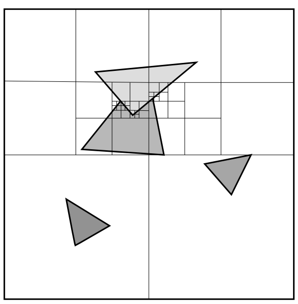
VSA: Warnock's Alg

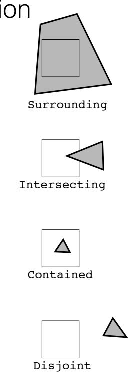
- recursively subdivide screen
- stop when "simple" or at pixel
- at pixel draw closest object



Warnock's Alg

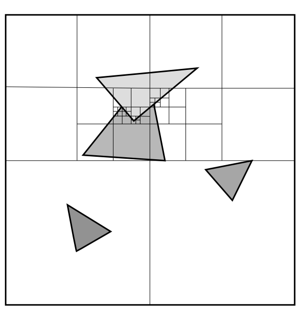
- classify polygon in list w.r.t each region





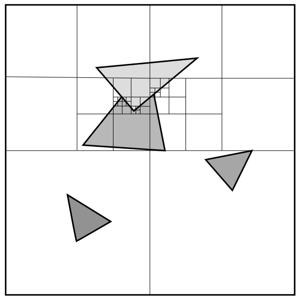
Warnock's Alg

- all polygons disjoint
  -> draw background
- 1 intersecting or contained
  -> draw background, polygon
- polygon surrounding
  -> draw polygon
- 1 surrounding in front of other polygons (any type)
  -> draw surrounding poly



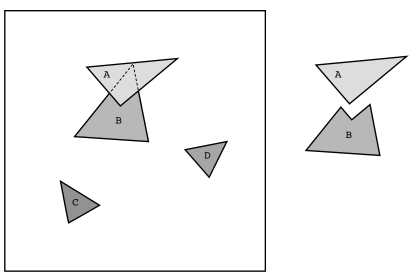
Warnock's Alg

- Advantages
  - relatively simple
- Disadvantages
  - memory for polygons
  - memory for frame buffer



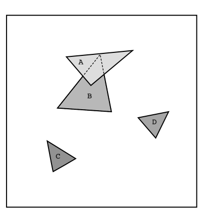
VSA: Weiler-Atherton

- subdivide at polygon boundaries
- need powerful clipping algorithm



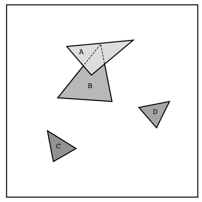
Weiler-Atherton

- fast sort of polygons by z
- select "closest" polygon
- use it to clip the rest
- if any poly inside clipping poly closer -> initial sort wrong
  - use it as clipping poly first
- otherwise discard those inside
- draw clipping poly



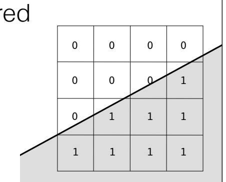
Weiler-Atherton

- Advantages
  - object space
- Disadvantages
  - memory for polygons
  - clipping

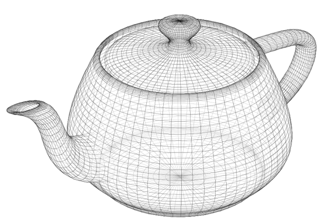


Sub-Pixel Area Subdivision
(Antialiasing)

- continue Warnock's Alg sub-pixel
- Weiler-Atherton at each pixel
- A-Buffer
  - 1 depth, 1 lighting calculation per pixel
  - use bitmap to approximate area covered



Curved Surfaces



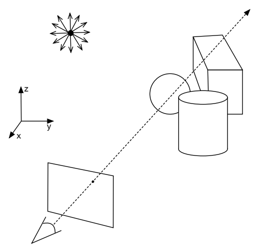
- valiant attempts
- z-buffer
- recursive subdivision- stop when "flat" or "small"

VSA: Ray Casting

(Ray Tracing)

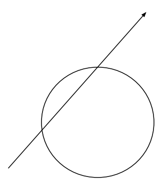
- shoot ray ray from eye through screen into world
- intersect objects with ray
- find closest intersection
- do shading/lighting calculation

- very floating-point intensive



Computing Intersections

- ray: $p_t = o + t \cdot \vec{d}$
- sphere: $x^2 + y^2 + z^2 = 1$



$$(o_x + t \cdot \vec{d}_x)^2 + (o_y + t \cdot \vec{d}_y)^2 + (o_z + t \cdot \vec{d}_z)^2 - 1 = 0$$

$$(o_x^2 + 2 \cdot o_x \cdot t \cdot \vec{d}_x + t^2 \cdot \vec{d}_x^2) + (o_y^2 + 2 \cdot o_y \cdot t \cdot \vec{d}_y + t^2 \cdot \vec{d}_y^2) + (o_z^2 + 2 \cdot o_z \cdot t \cdot \vec{d}_z + t^2 \cdot \vec{d}_z^2) - 1 = 0$$

$$(\vec{d}_x^2 + \vec{d}_y^2 + \vec{d}_z^2)t^2 + 2(o_x \cdot \vec{d}_x + o_y \cdot \vec{d}_y + o_z \cdot \vec{d}_z)t + (o_x^2 + o_y^2 + o_z^2) - 1 = 0$$

$$dot(d,d)t^2 + 2dot(o,d)t + dot(o,o) - 1 = 0$$

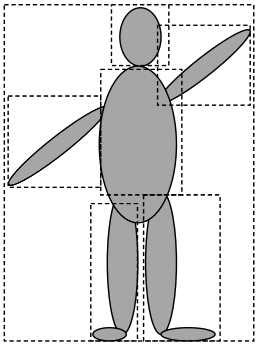
$$at^2 + bt + c = 0$$

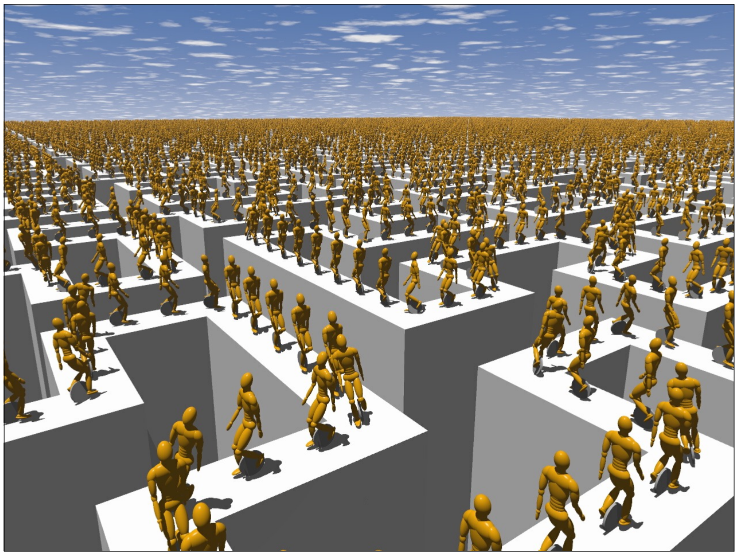
Ray Casting Efficiency

- 1K * 1K * 1K objects = one billion intersection calc 😞
- improvements
  - hierarchical bounding volumes
  - spatial partitioning

Hierarchical Bounding Volumes

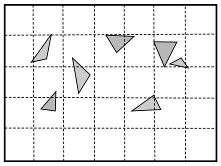
- Explicit creation of hierarchy
- Automatic creation
  - list of objects -> tree
  - intersection ∝ surface area





Spatial Partitoning

- 2-D



- 3-D
  - subdivide space
  - traverse grid one *voxel* at a time
  - uniform vs octree