CS-184: Computer Graphics

Lecture #10: Clipping and Hidden Surfaces

Prof. James O'Brien
University of California, Berkeley

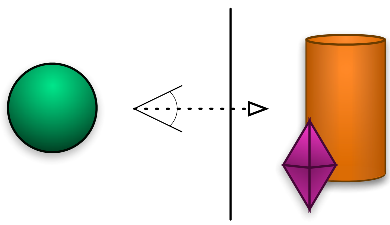
V2006-F-10-1.0

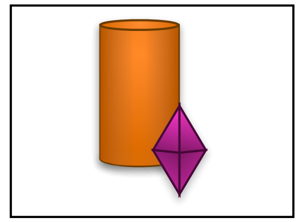
Today

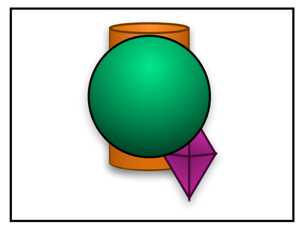
- Clipping
  - Clipping to view volume
  - Clipping arbitrary polygons
- Hidden Surface Removal
  - Z-Buffer
  - BSP Trees
  - Others

Clipping

- Stuff outside view volume should not be drawn
  - Too close: obscures view







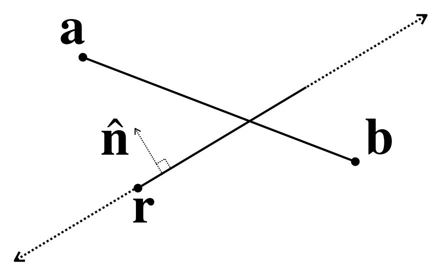
Clipping

- Stuff outside view volume should not be drawn
  - Too close: obscures view
  - Too far:
    - Complexity
    - Z-buffer problems
  - Too high/low/right/left:
    - Memory errors
    - Broken algorithms
    - Complexity

Clipping Line to Line/Plane

Line segment to be clipped

$$\mathbf{x}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$$



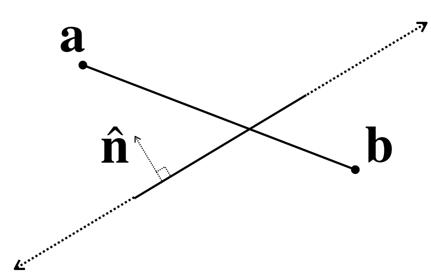
Line/plane that clips it

$$\hat{\mathbf{n}} \cdot \mathbf{x} - \hat{\mathbf{n}} \cdot \mathbf{r} = 0$$

Clipping Line to Line/Plane

Line segment to be clipped

$$\mathbf{x}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$$



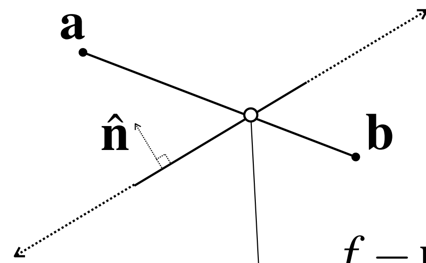
Line/plane that clips it

$$\hat{\mathbf{n}} \cdot \mathbf{x} - f = 0$$

Clipping Line to Line/Plane

Line segment to be clipped

$$\mathbf{x}(t) = \underbrace{\mathbf{a} + t(\mathbf{b} - \mathbf{a})}$$



$$t = \frac{f - \hat{\mathbf{n}} \cdot \mathbf{a}}{\hat{\mathbf{n}} \cdot \mathbf{d}}$$

Line/plane that clips it

$$\hat{\mathbf{n}} \cdot \mathbf{x} - f = 0$$

$$\hat{\mathbf{n}} \cdot (\mathbf{a} + t(\mathbf{b} - \mathbf{a})) - f = 0$$

$$\hat{\mathbf{n}} \cdot \mathbf{a} + t(\hat{\mathbf{n}} \cdot (\mathbf{b} - \mathbf{a})) - f = 0$$

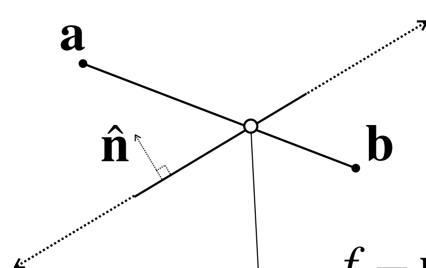
Clipping Line to Line/Plane

- Segment may be on one side

  $$t \notin [0 \ldots 1]$$

- Lines may be parallel

  $$\hat{\mathbf{n}} \cdot \mathbf{d} = 0$$



$$t = \frac{f - \hat{\mathbf{n}} \cdot \mathbf{a}}{\hat{\mathbf{n}} \cdot \mathbf{d}}$$

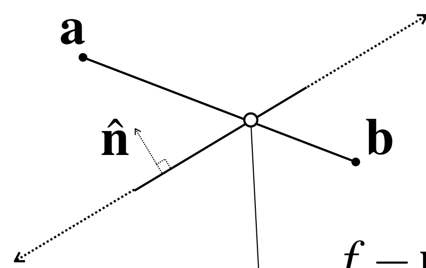
Clipping Line to Line/Plane

- Segment may be on one side

  $t \notin [0 \ldots 1]$

- Lines may be parallel

  $\hat{\mathbf{n}} \cdot \mathbf{d} = 0$



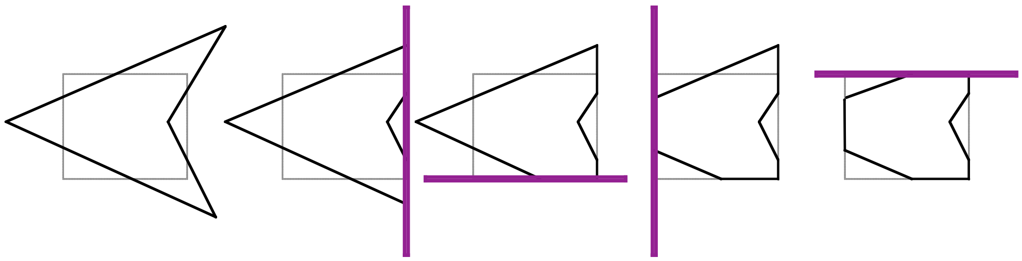
$$t = \frac{f - \hat{\mathbf{n}} \cdot \mathbf{a}}{\hat{\mathbf{n}} \cdot \mathbf{d}}$$

$\boxed{\left|\hat{\mathbf{n}} \cdot \mathbf{d}\right| \leq \varepsilon \quad \text{(Recall comments about numerical issues)}}$

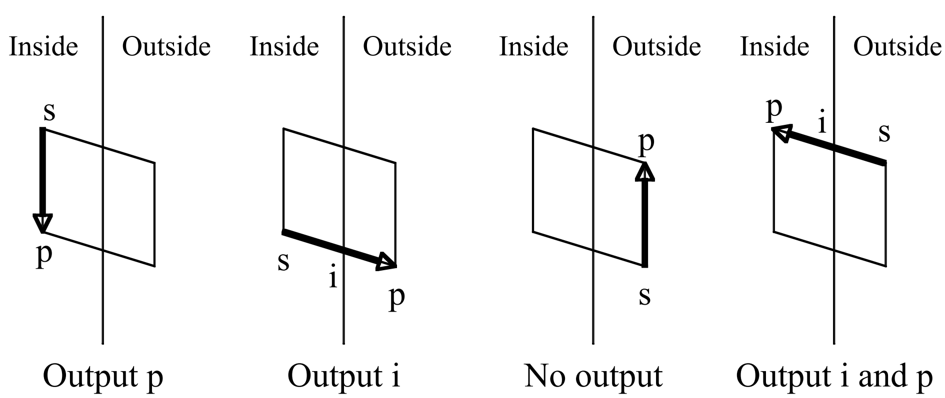
Polygon Clip to Convex Domain

- Convex domain defined by collection of planes (or lines or hyper-planes)
- Planes have outward pointing normals
- Clip against each plane in turn
- Check for early/trivial rejection

Polygon Clip to Convex Domain



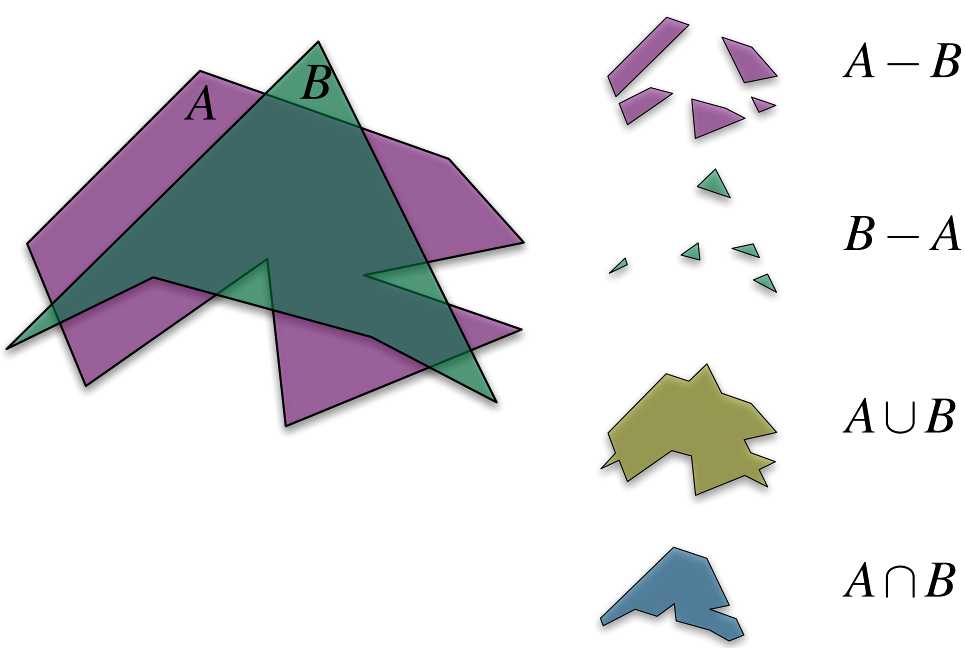
Polygon Clip to Convex Domain



Polygon Clip to Convex Domain

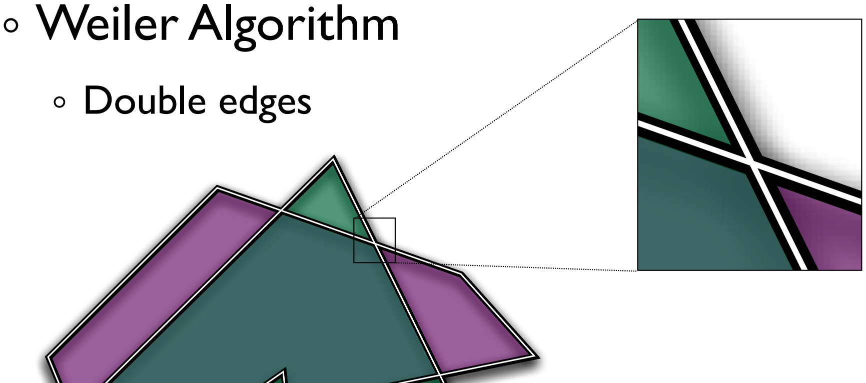
- Sutherland-Hodgman algorithm
  - Basically edge walking
- Clipping done often... should be efficient
  - Liang-Barsky parametric space algorithm
  - See text for clipping in 4D homogenized coordinates

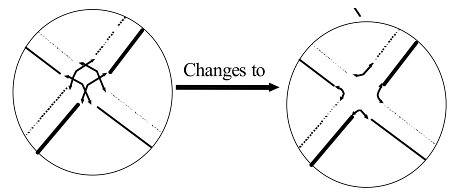
General Polygon Clipping



General Polygon Clipping

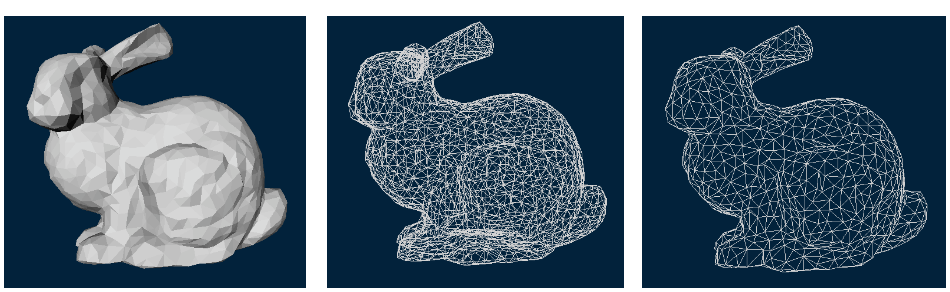
- Weiler Algorithm
  - Double edges





Hidden Surface Removal

- True 3D to 2D projection would put every thing overlapping into the view plane.
- We need to determine what's in front and display only that.



Z-Buffers

- Add extra depth channel to image
- Write Z values when writing pixels
- Test Z values before writing

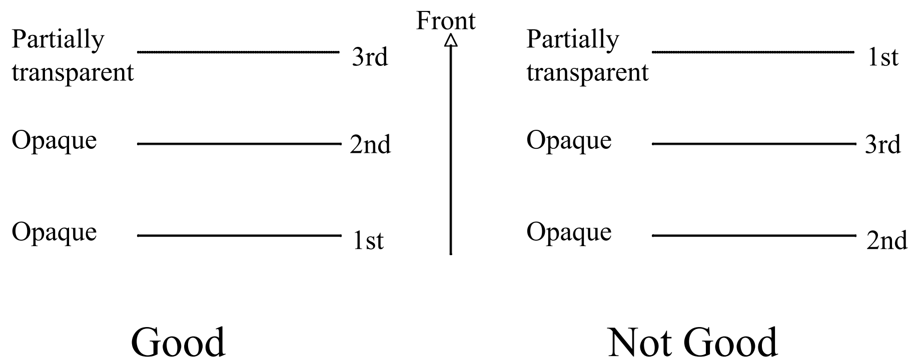
Images from Okan Arikan

Z-Buffers

- Benefits
  - Easy to implement
  - Works for most any geometric primitive
  - Parallel operation in hardware
- Limitations
  - Quantization and aliasing artifacts
  - Overfill
  - Transparency does not work well

Z-Buffers

- Transparency requires partial sorting:



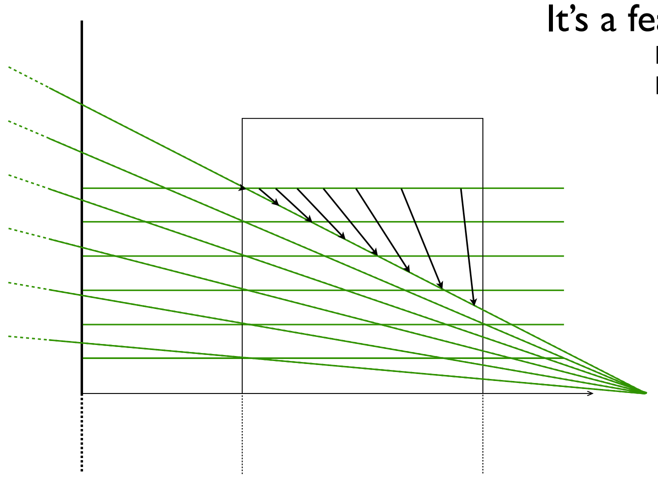
Z-Buffers

Recall depth-value distortions.

It's a feature...

More resolution near viewer

Best use of limited precision



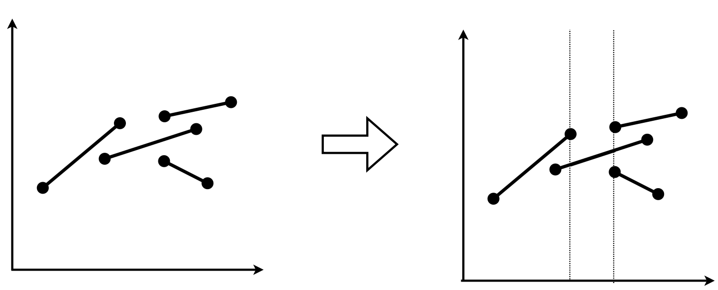
A-Buffers

- Store sorted list of "fragments" at each pixel
- Draw all opaque stuff first then transparent
- Stuff behind full opacity gets ignored

- Nice for antialiasing...

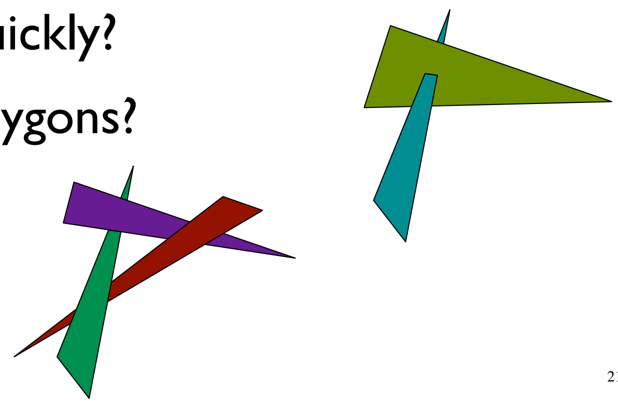
Scan-line Algorithm

- Assume polygons don't intersect
- Each time an edge is crossed determine who's on top



Painter's Algorithm

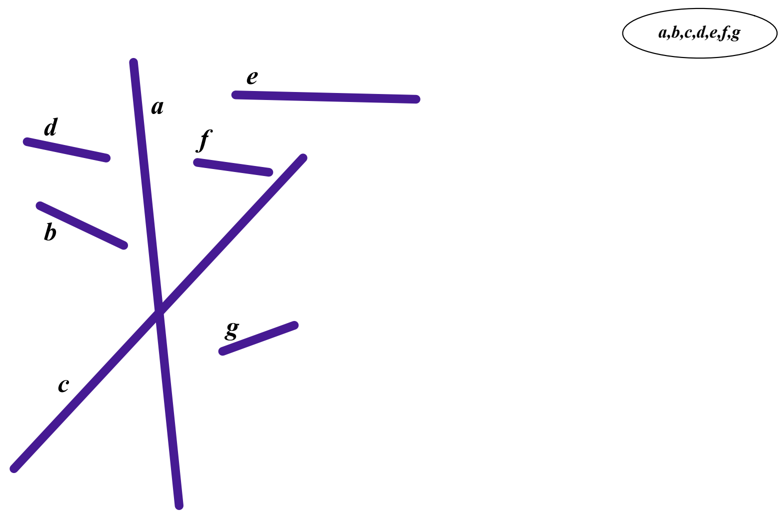
- Sort Polygons Front-to-Back
  - Draw in order
  - Back-to-Front works also, but wasteful
- How to sort quickly?
- Intersecting polygons?
- Cycles?



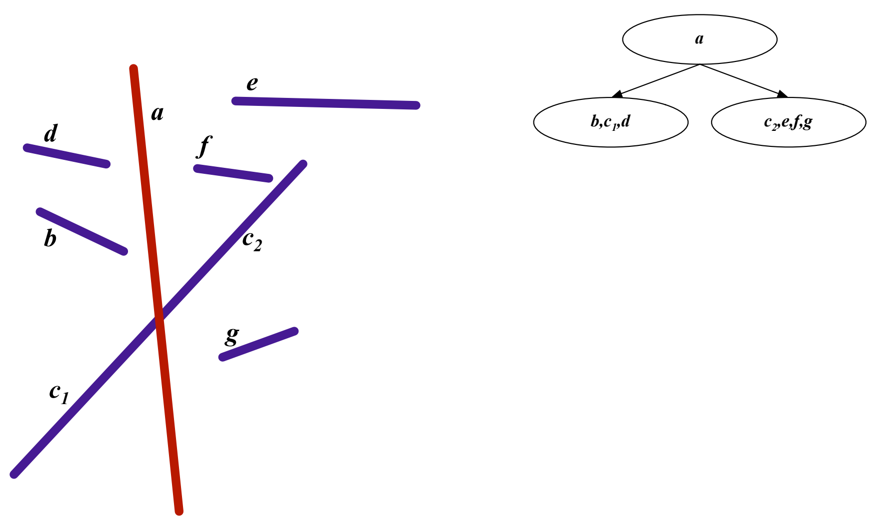
BSP-Trees

- Binary Space Partition Trees
  - Split space along planes
  - Allows fast queries of some spatial relations
- Simple construction algorithm
  - Select a plane as sub-tree root
  - Everything on one side to one child
  - Everything on the other side to other child
  - Use random polygon for splitting plane

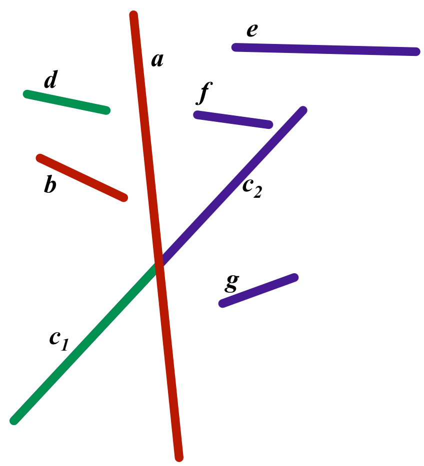
BSP-Trees

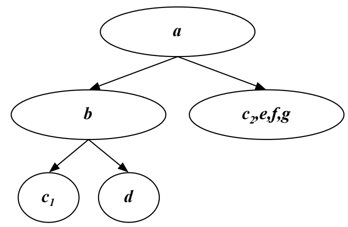


BSP-Trees

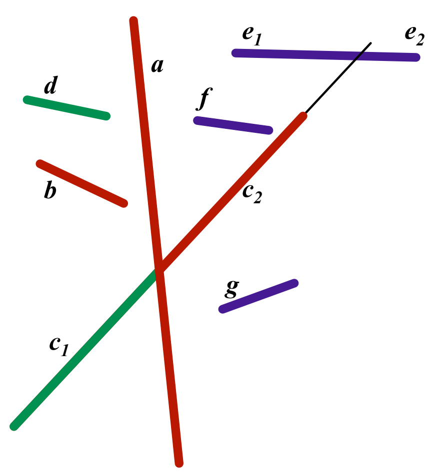


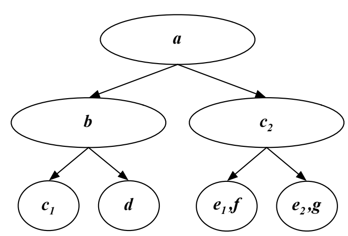
BSP-Trees



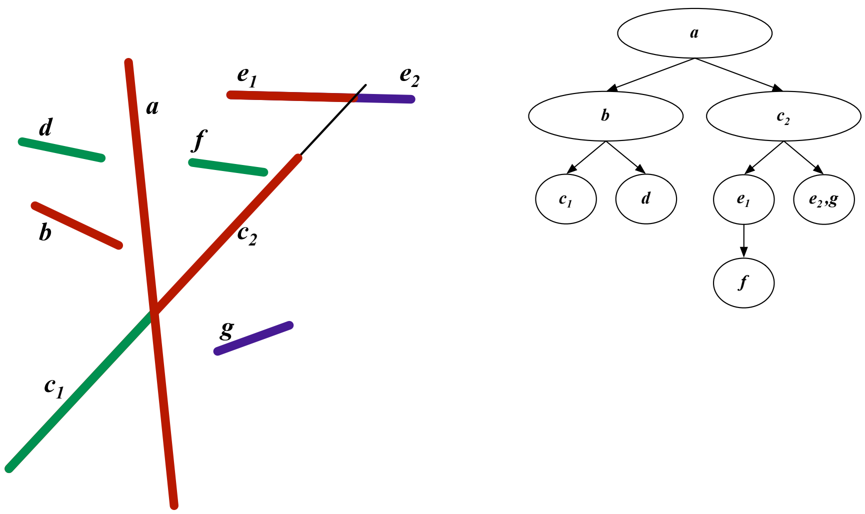


BSP-Trees

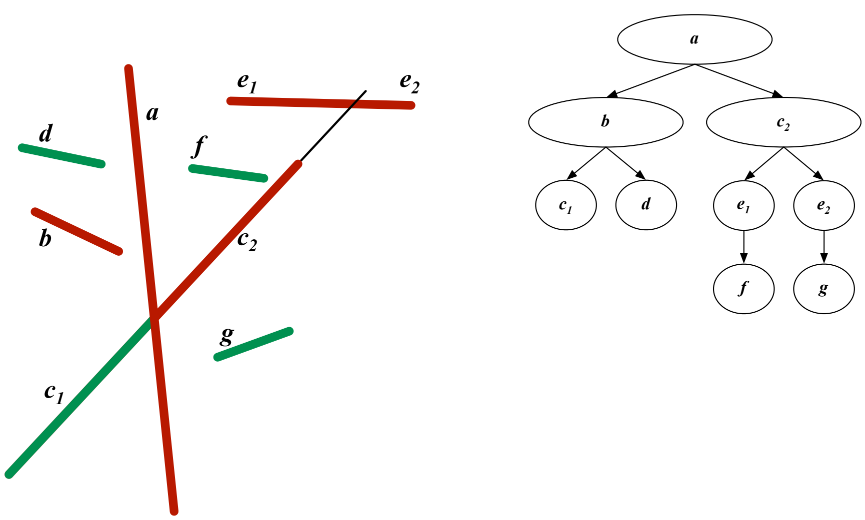




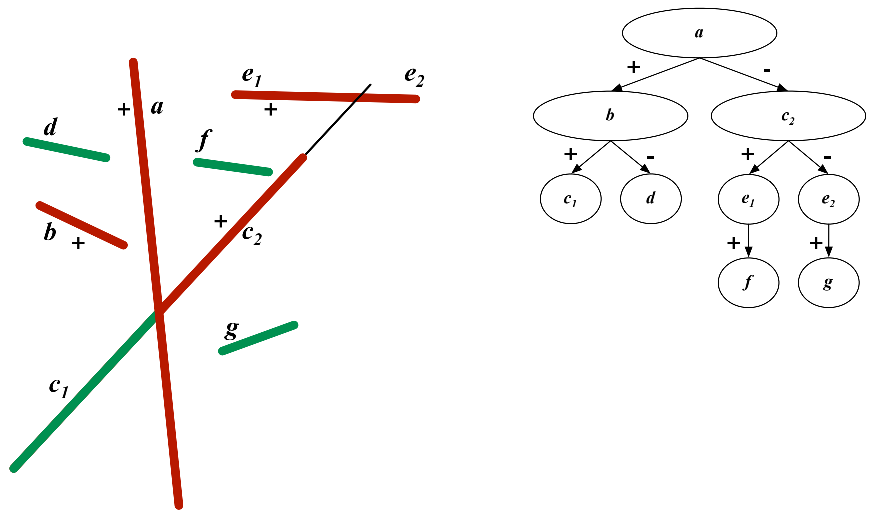
BSP-Trees



BSP-Trees



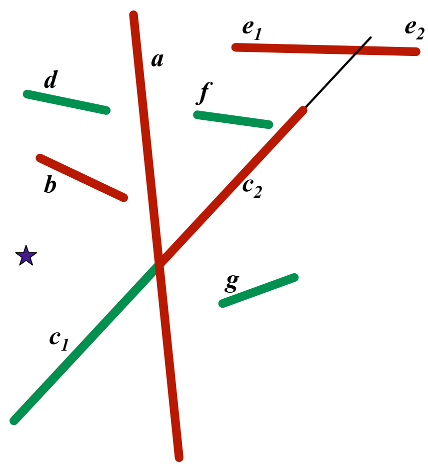
BSP-Trees

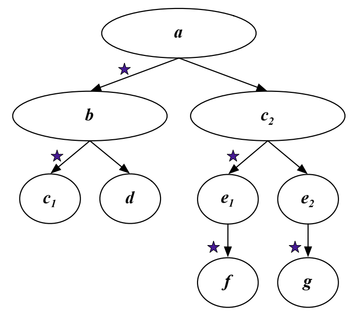


BSP-Trees

- Visibility Traversal
  - Variation of in-order-traversal
    - Child one
    - Sub-tree root
    - Child two
  - Select "child one" based on location of viewpoint
    - Child one on same side of sub-tree root as viewpoint

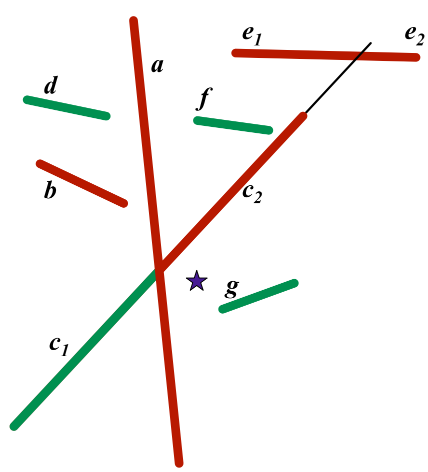
BSP-Trees

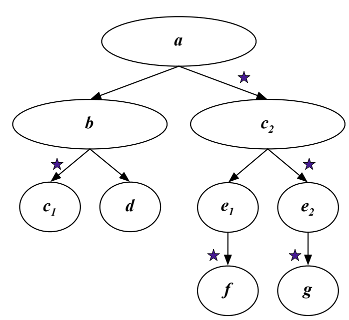




c_1:b:d:a:f:e_1:c_2:g:e_2

BSP-Trees





g:e_2:c_2:f:e_1:a:c_1:b:d