CMP205: Computer Graphics

Lecture 6: Hidden Surface Removal

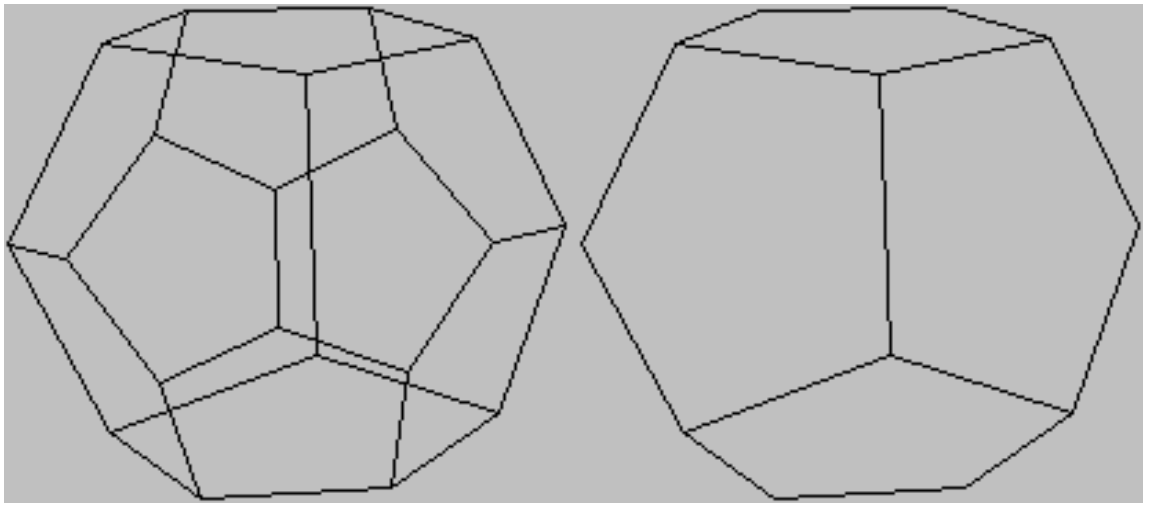
Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Fall 2012

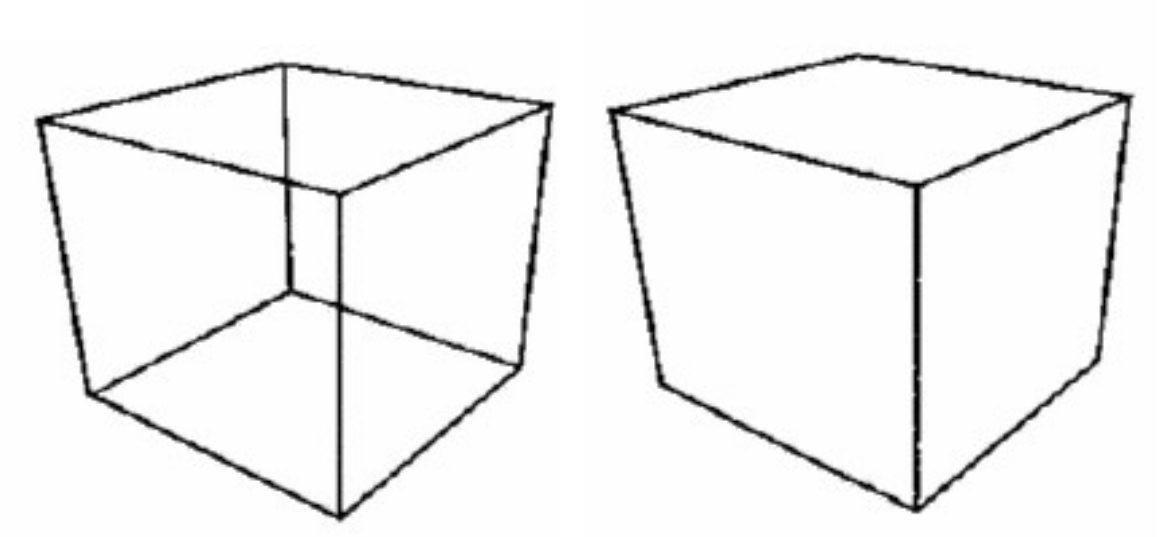
Agenda

- Z-Buffer
- Painter's Algorithm
- BSP Trees

Acknowledgment: Some slides adapted from Steve Marschner and Maneesh Agrawala

Hidden Surface Elimination

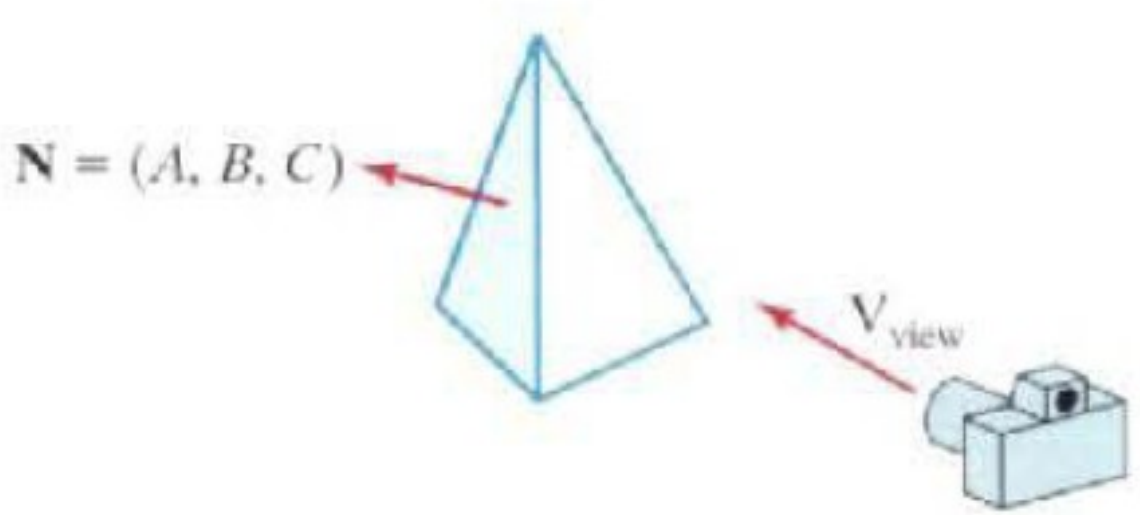




Back Face Elimination

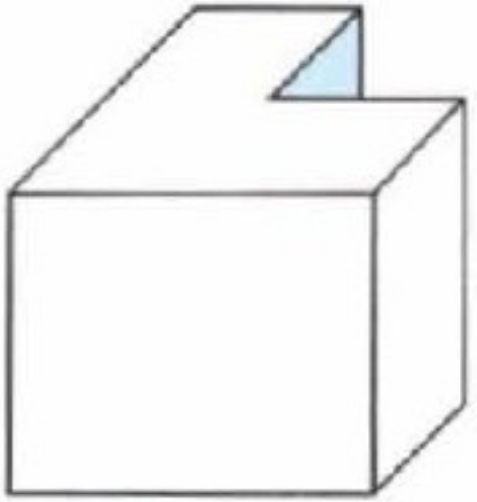
- Object Space
  - Works in the scene and compares objects
  - e.g. Back Face Culling
- Image Space
  - Works on the projected pixels
  - e.g. Z-Buffer and BSP Trees

Back Face Culling



$$\text{If } \boldsymbol{N} \cdot \boldsymbol{V}_{view} > 0 \;\rightarrow\; \text{Back Face}$$

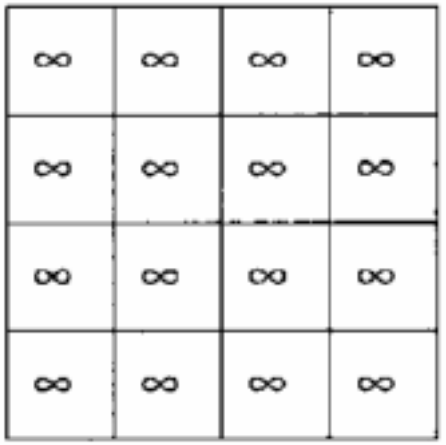
Is this enough?



Z-Buffer

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = M_{vp} M_{orth} P M_{cam} M_m \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

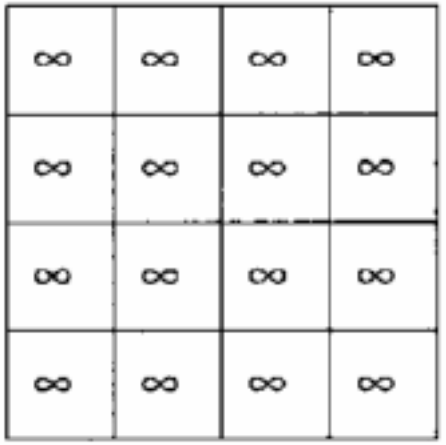
From the transformations pipeline, we get z_c



Keep a z value for every pixel on the screen !

Z-Buffer

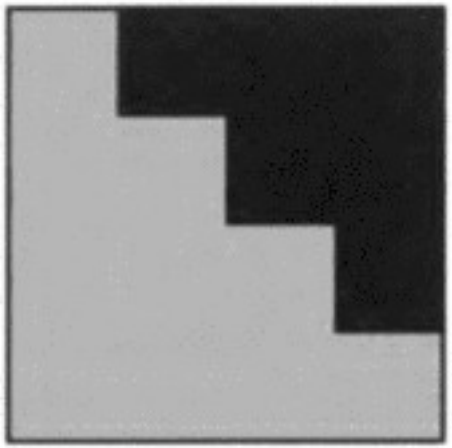
```
function SetPixel(i, j, color, z)
  if (z < z_buffer(i,j) then
    z_buffer(i,j) = z
    screen(i,j) = color
```

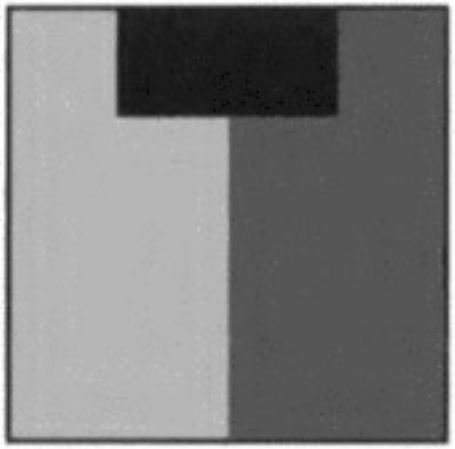


Z-Buffer

1	∞	∞	∞
1	3	∞	∞
1	3	5	∞
1	3	5	7

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

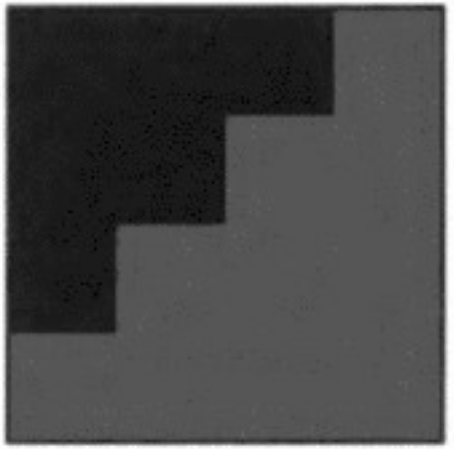


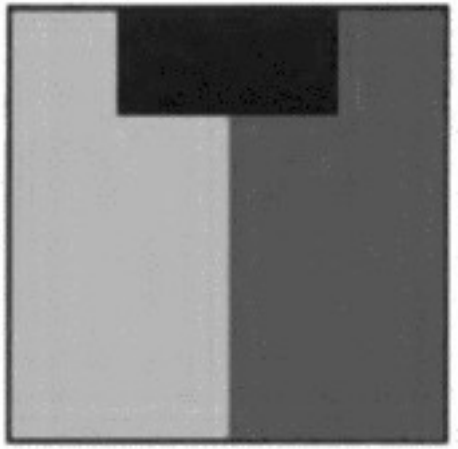


Left then right triangle

∞	∞	∞	1
∞	∞	3	1
∞	5	3	1
7	5	3	1

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1



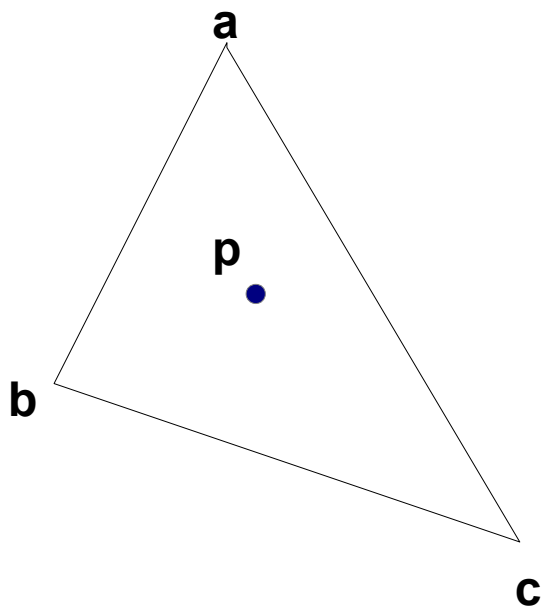


Right then left triangle

Z-Buffer independent of rendering order !

Z-Buffer

How do we get z values for pixels on a triangle ?!

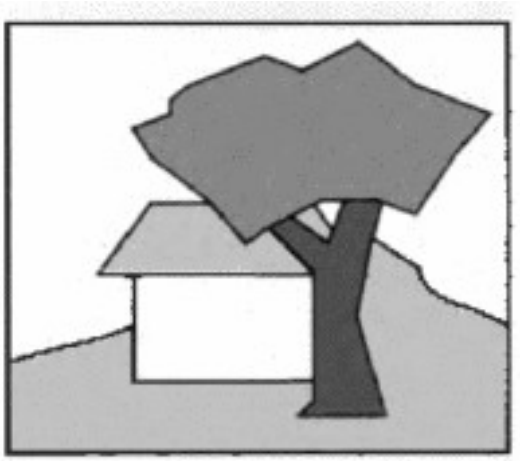


Barycentric Coordinates !

$$p(\alpha,\beta,\gamma)=\alpha a+\beta b+\gamma c$$

$$s.t.\quad \alpha+\beta+\gamma=1$$

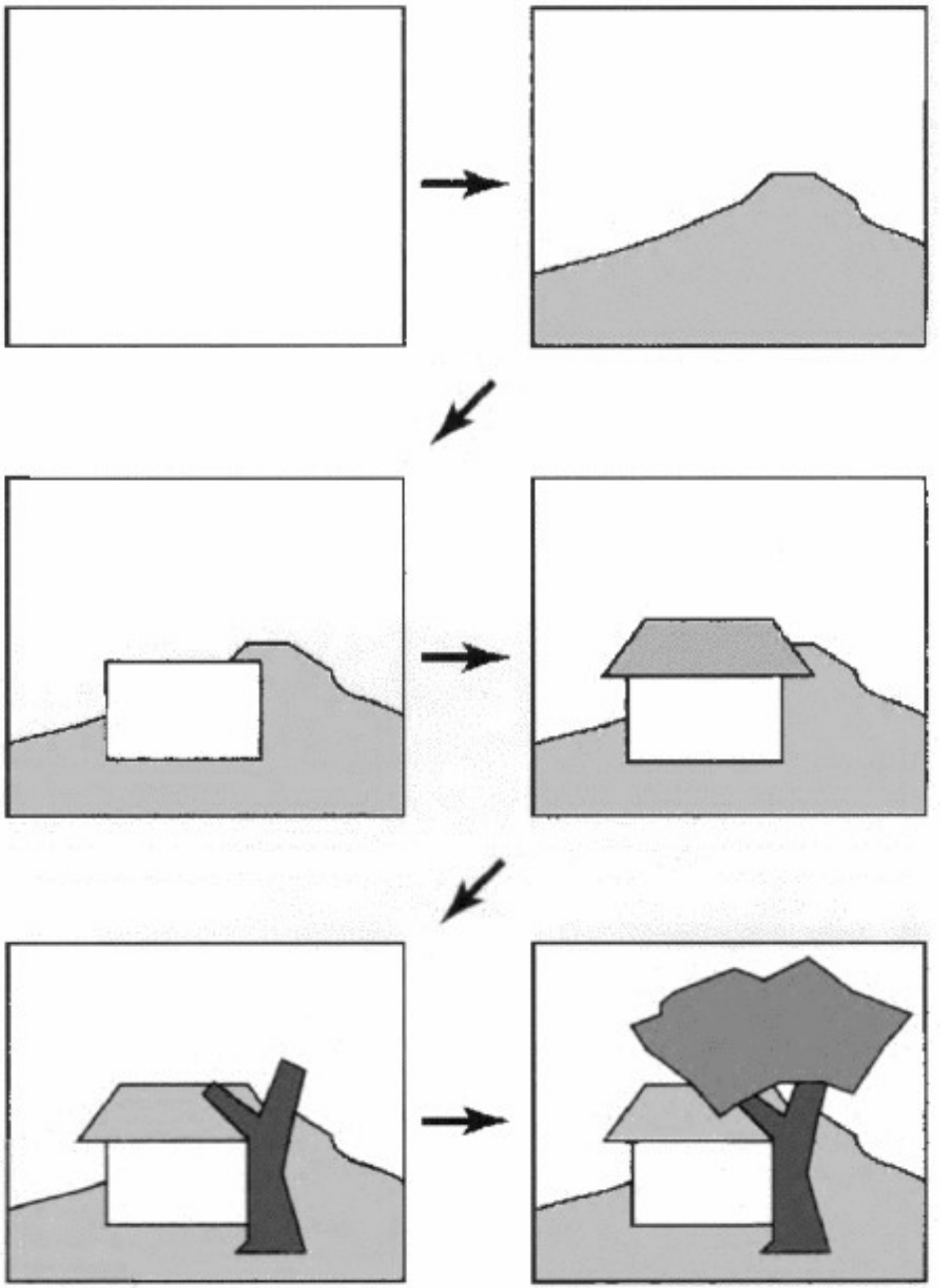
Painter's Algorithm



How should we draw these objects?

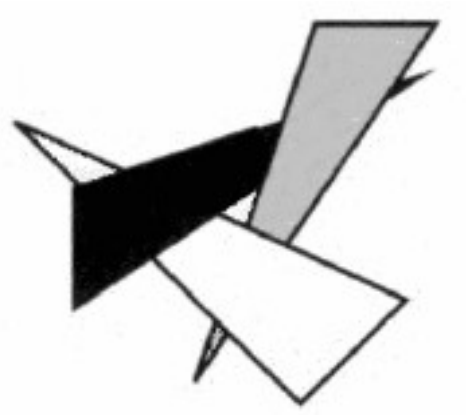
Painter's Algorithm

```
Sort objects back to front
For each object do
   draw object
```



Problem ?

Painter's Algorithm

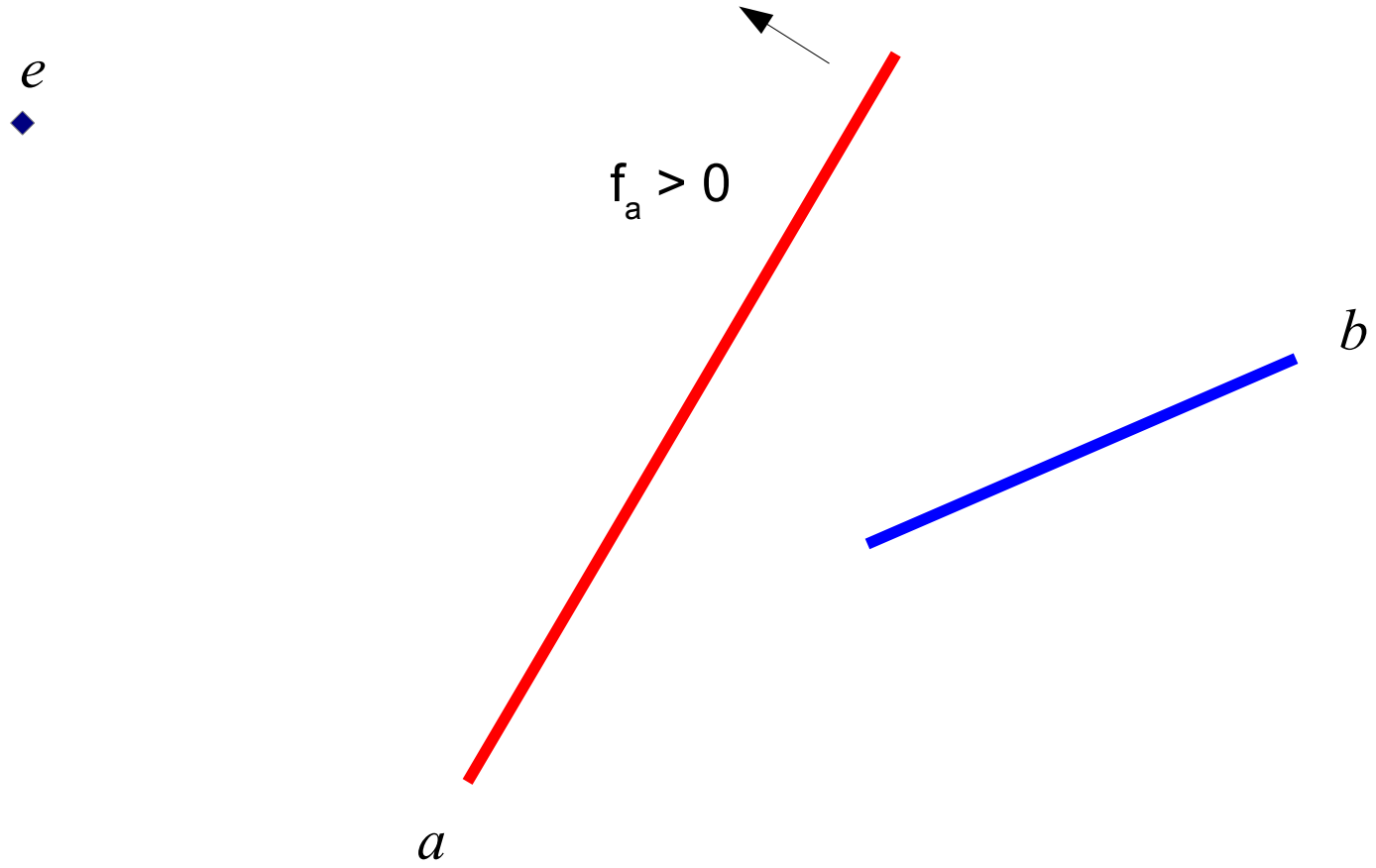


Cycles → No global order !

Binary Space Partitioning (BSP) Trees

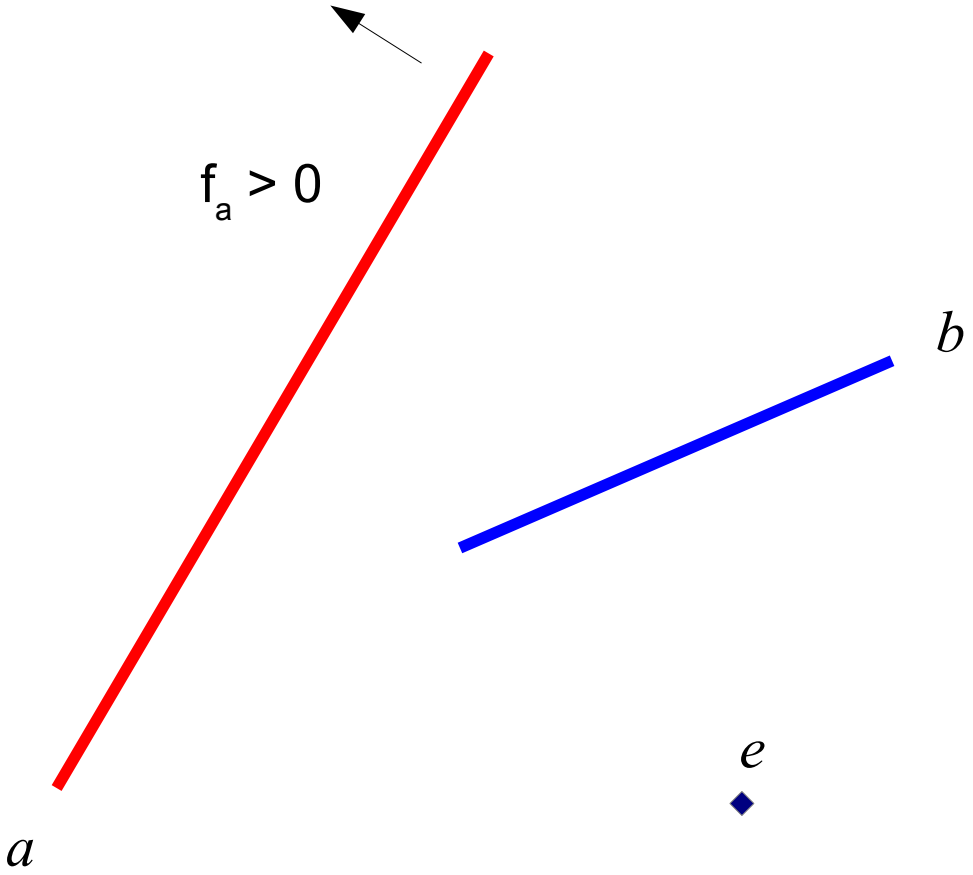
- A data structure built from the scene
- Built only once
- Can be used for any viewpoint
- Helps eliminate hidden surfaces

BSP Trees



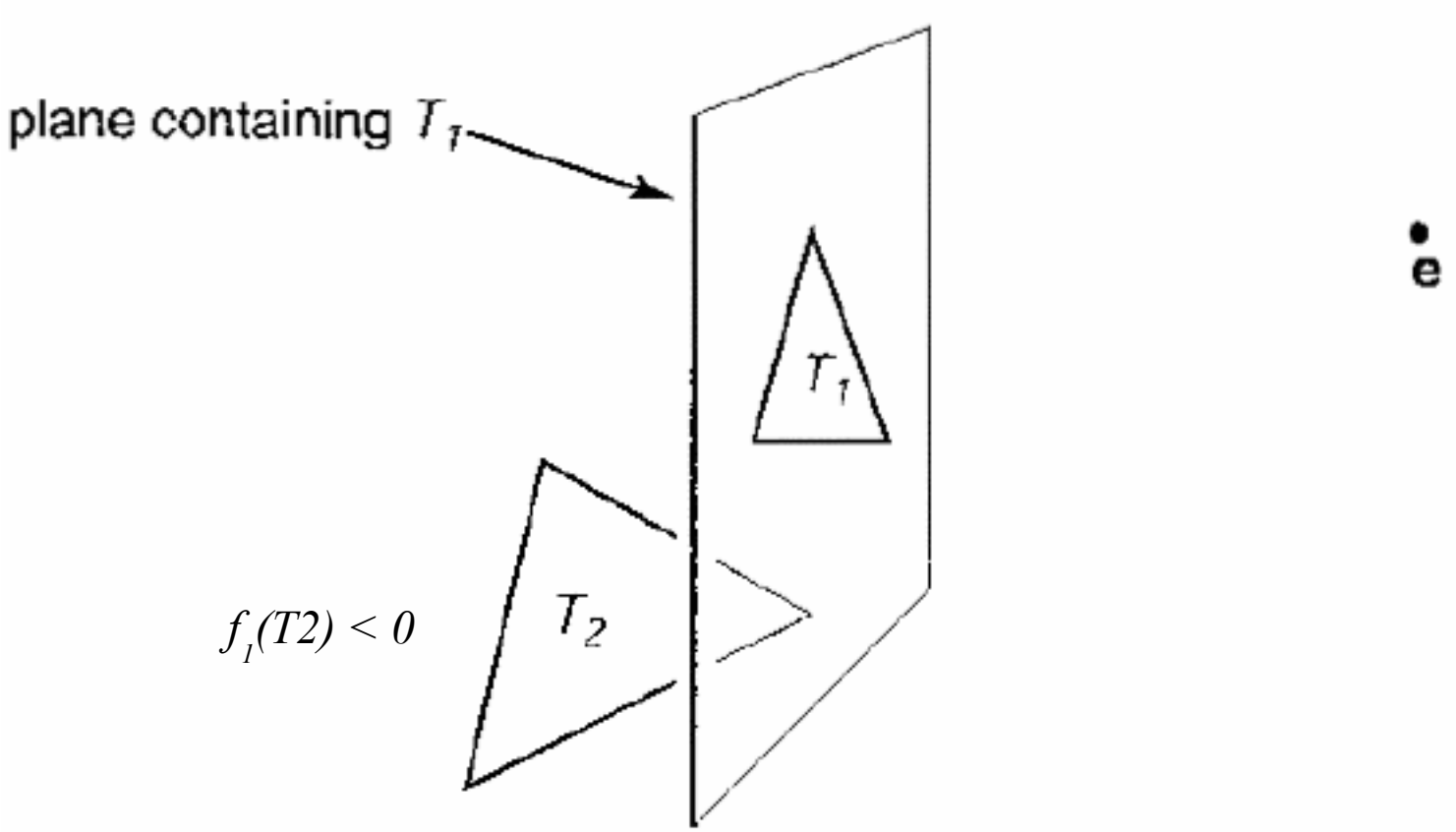
In which order should we draw *a* and *b* ?

BSP Trees



In which order should we draw *a* and *b* ?

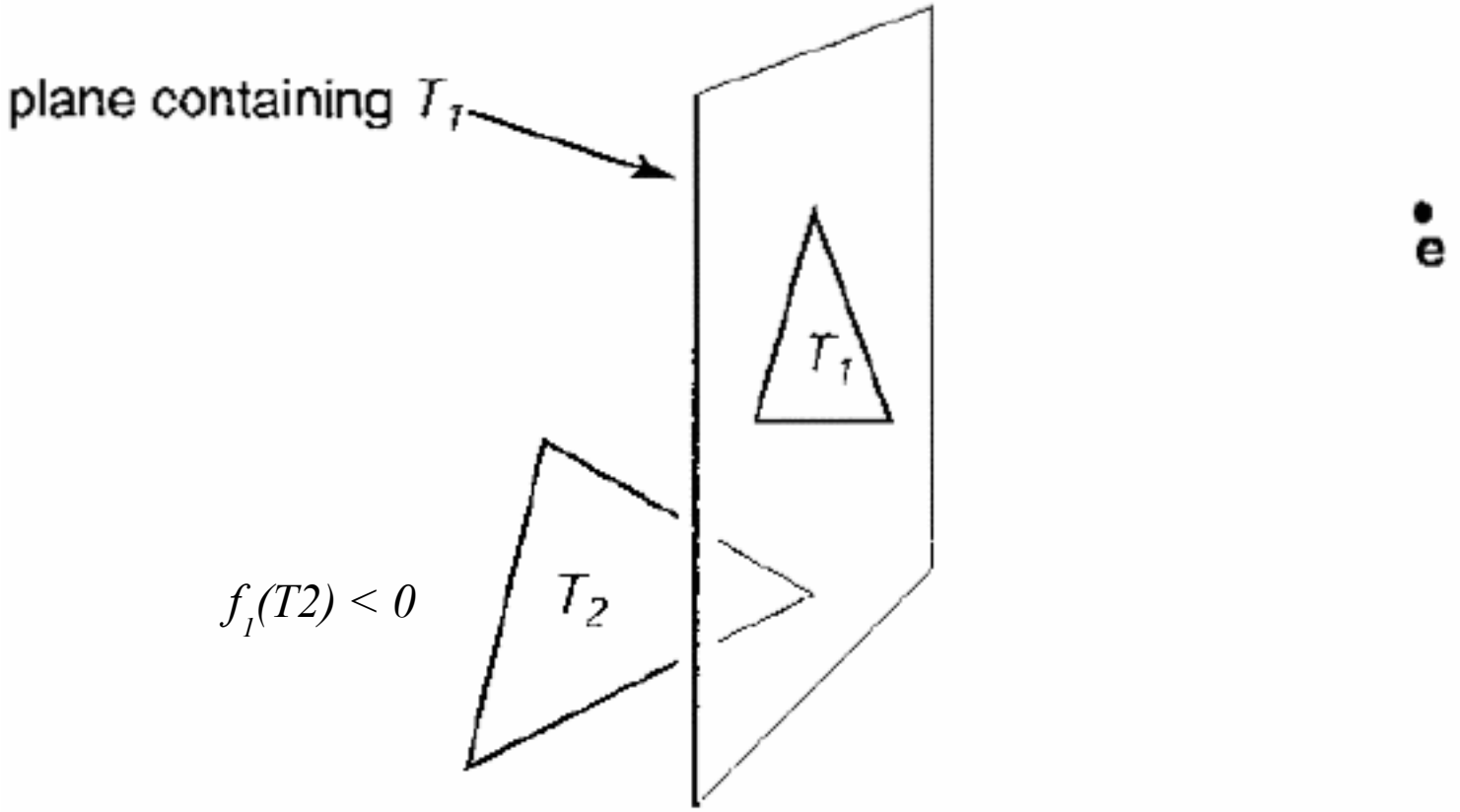
BSP Trees



Recall: $f_1(x) = 0$ is the implicit equation for plane T1

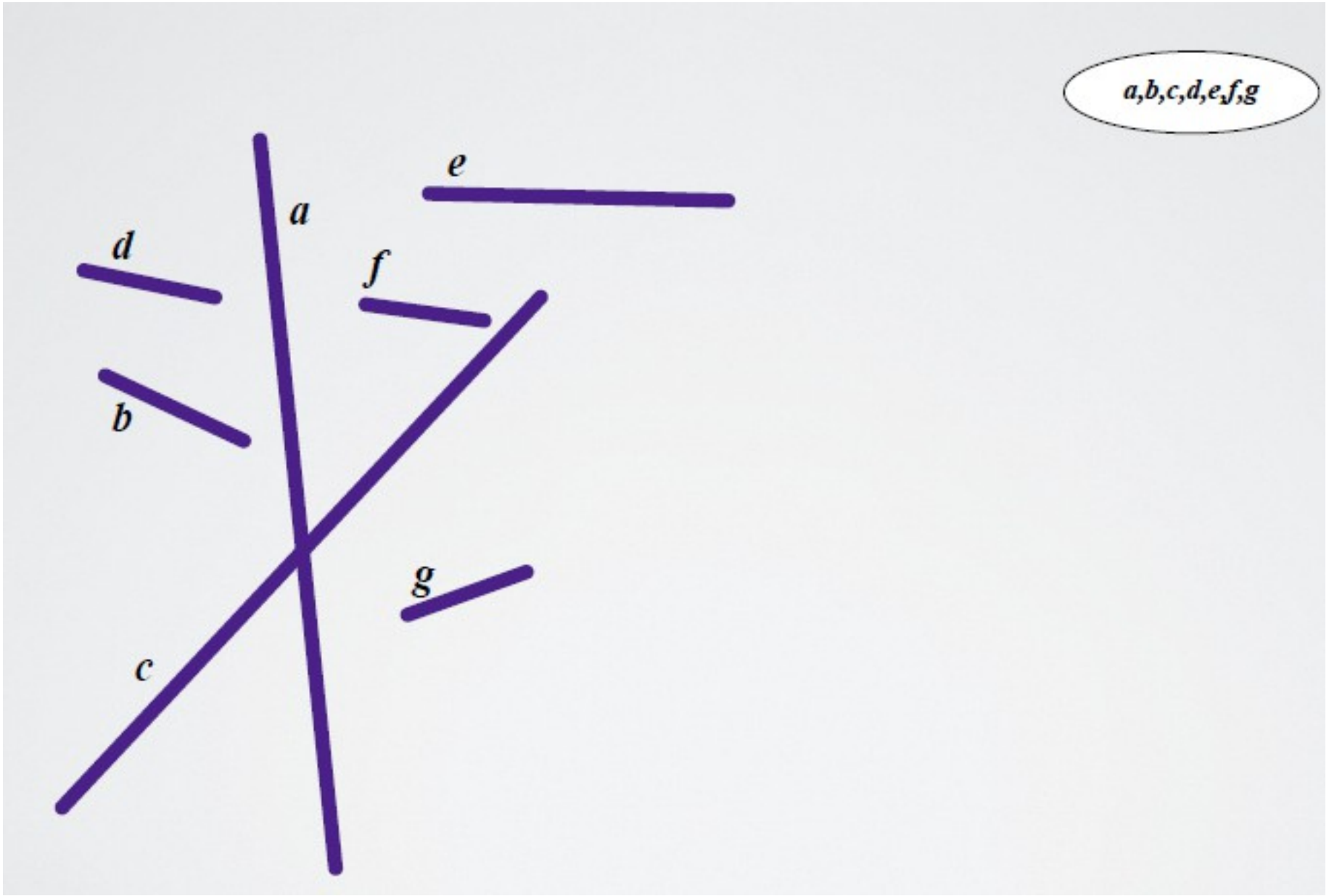
In which order should we draw T1 and T2?

BSP Trees

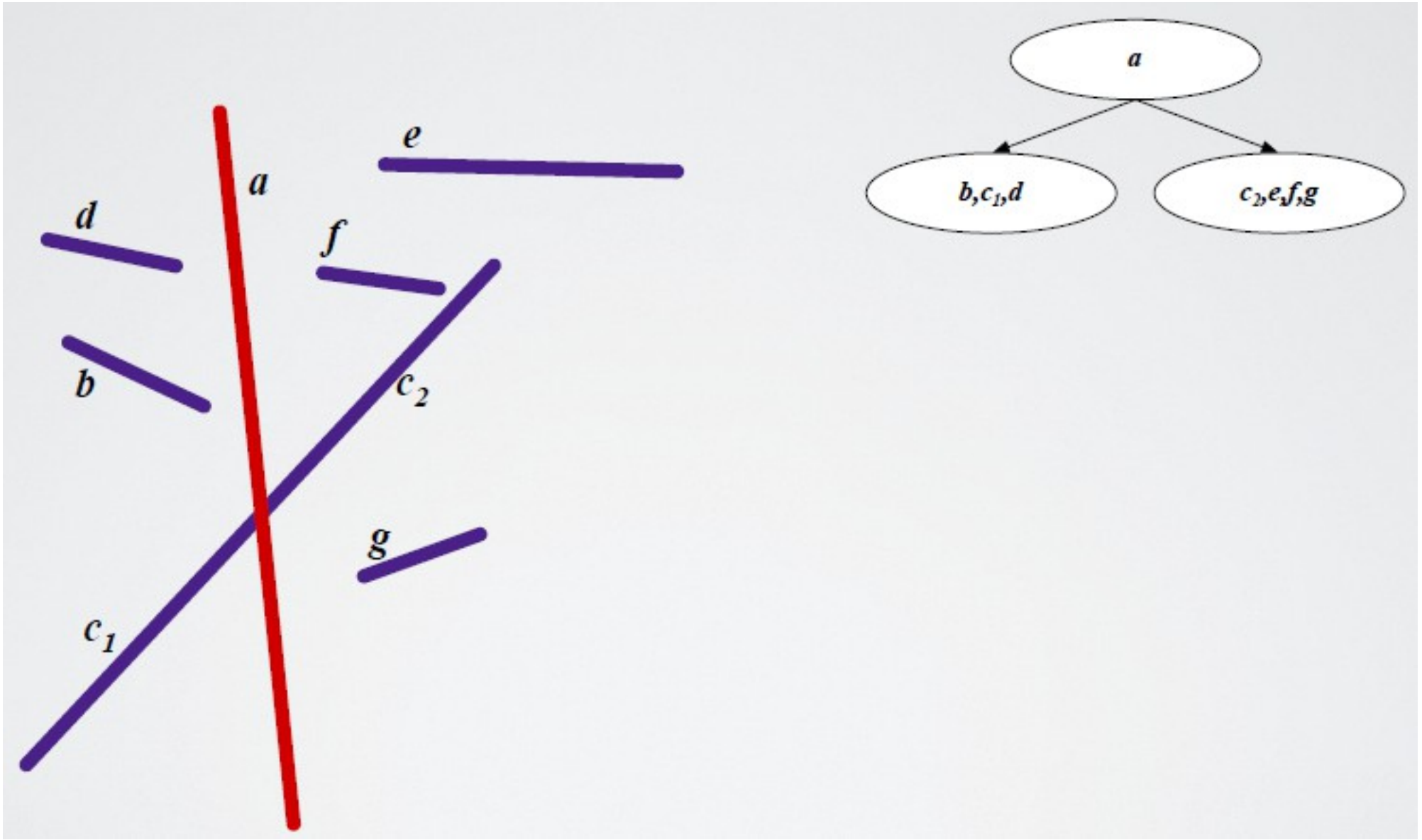



```
if (f1(e) < 0) then
   draw T1
   draw T2
else
   draw T2
   draw T1
```

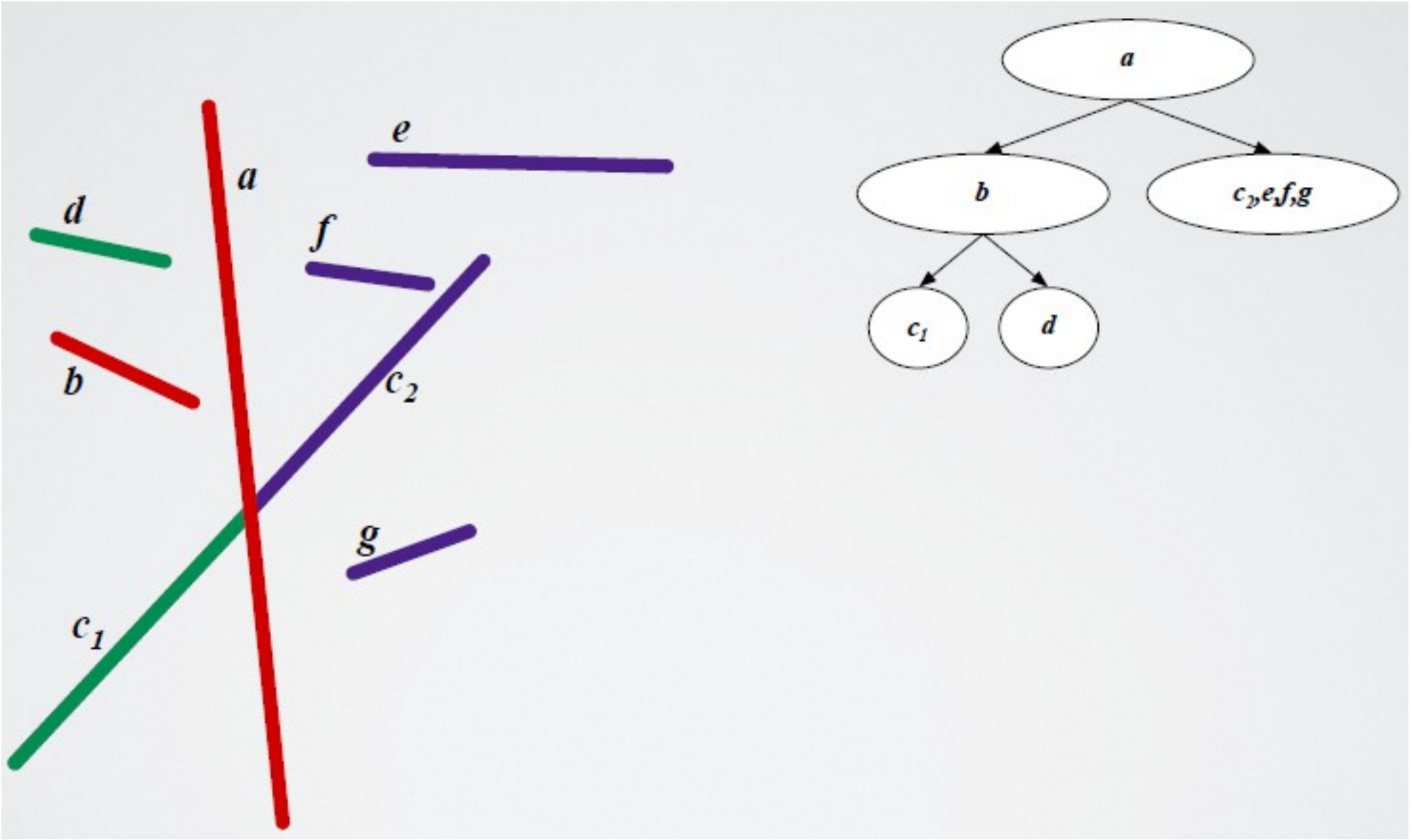
BSP Trees



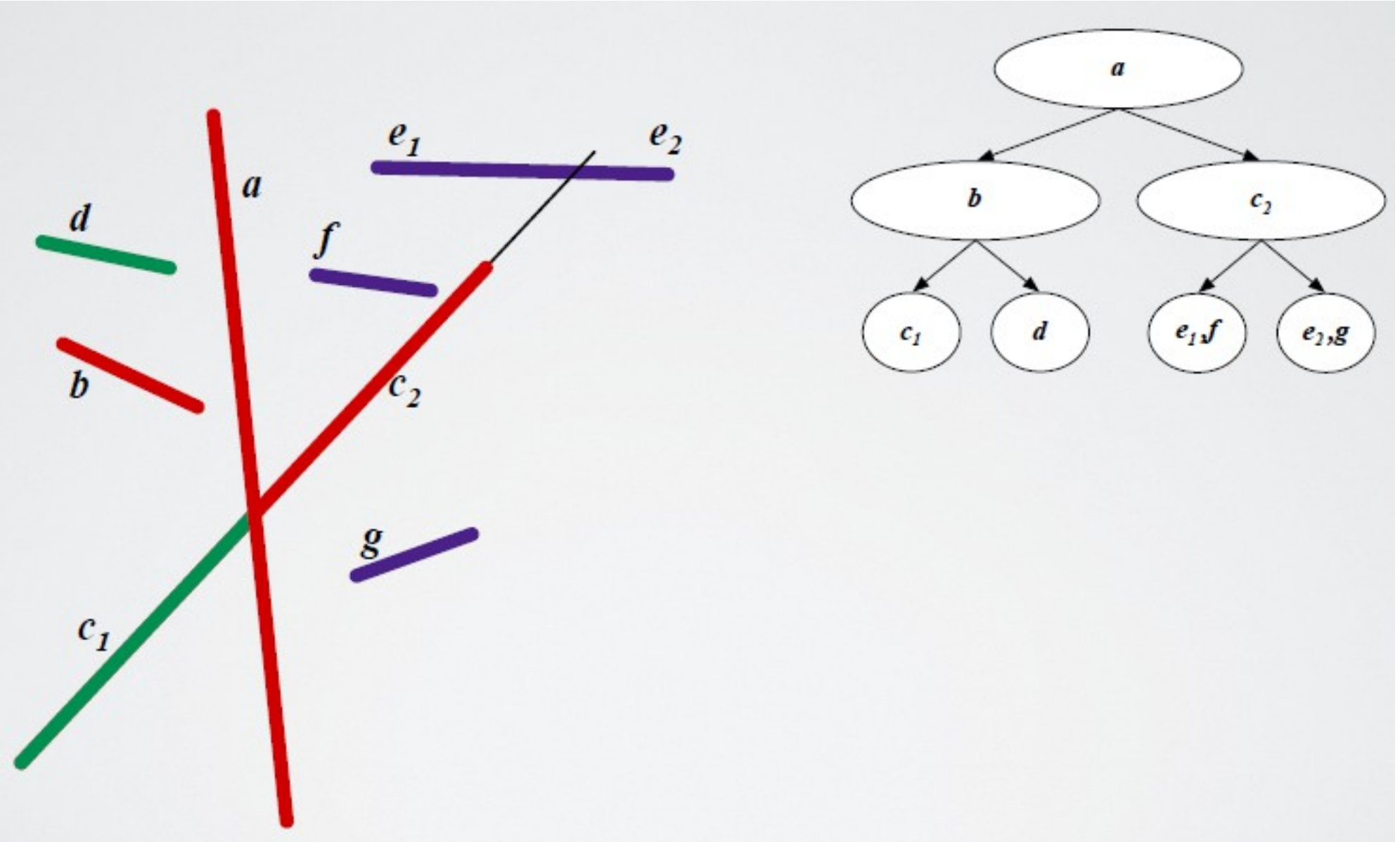
BSP Trees



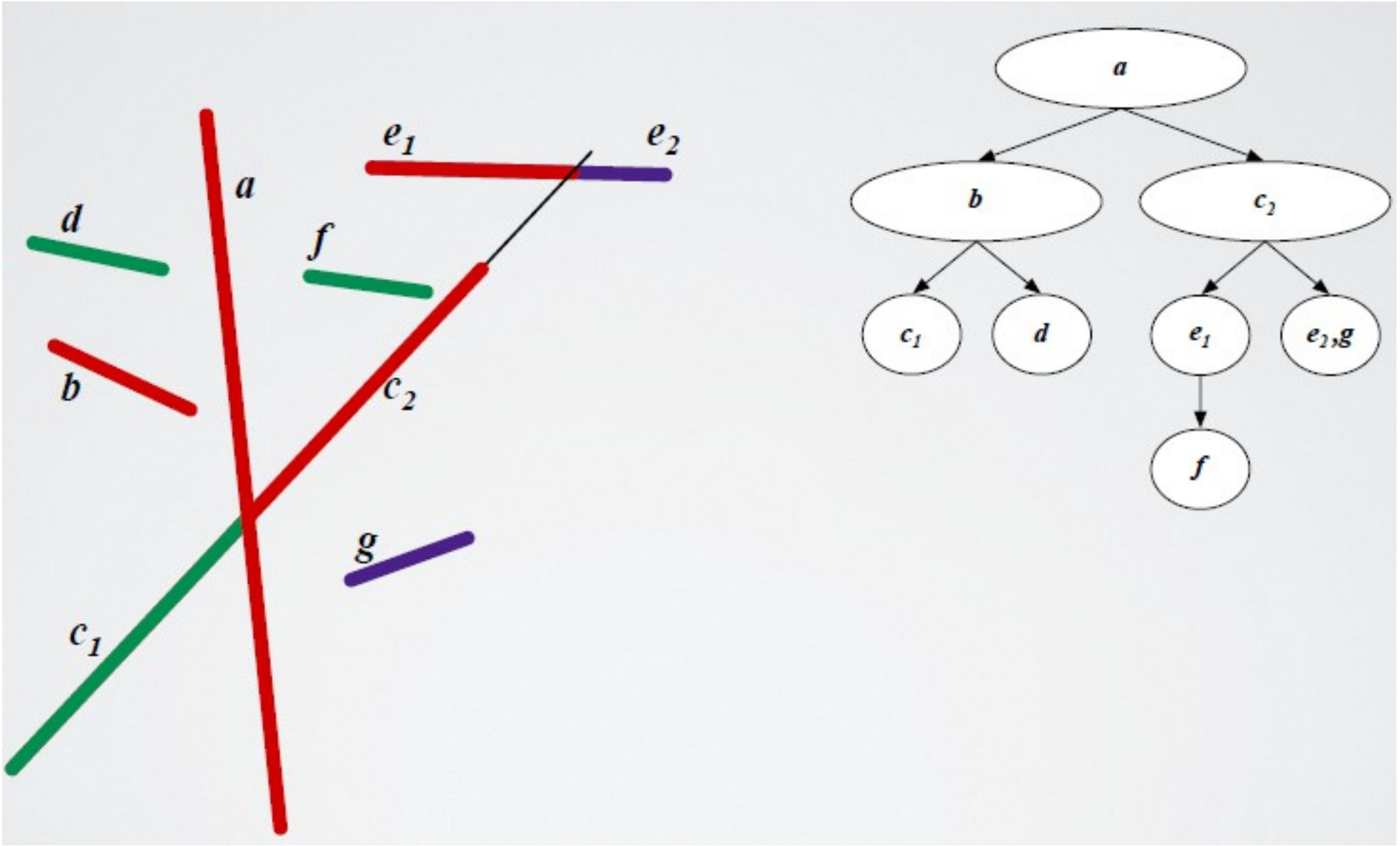
BSP Trees



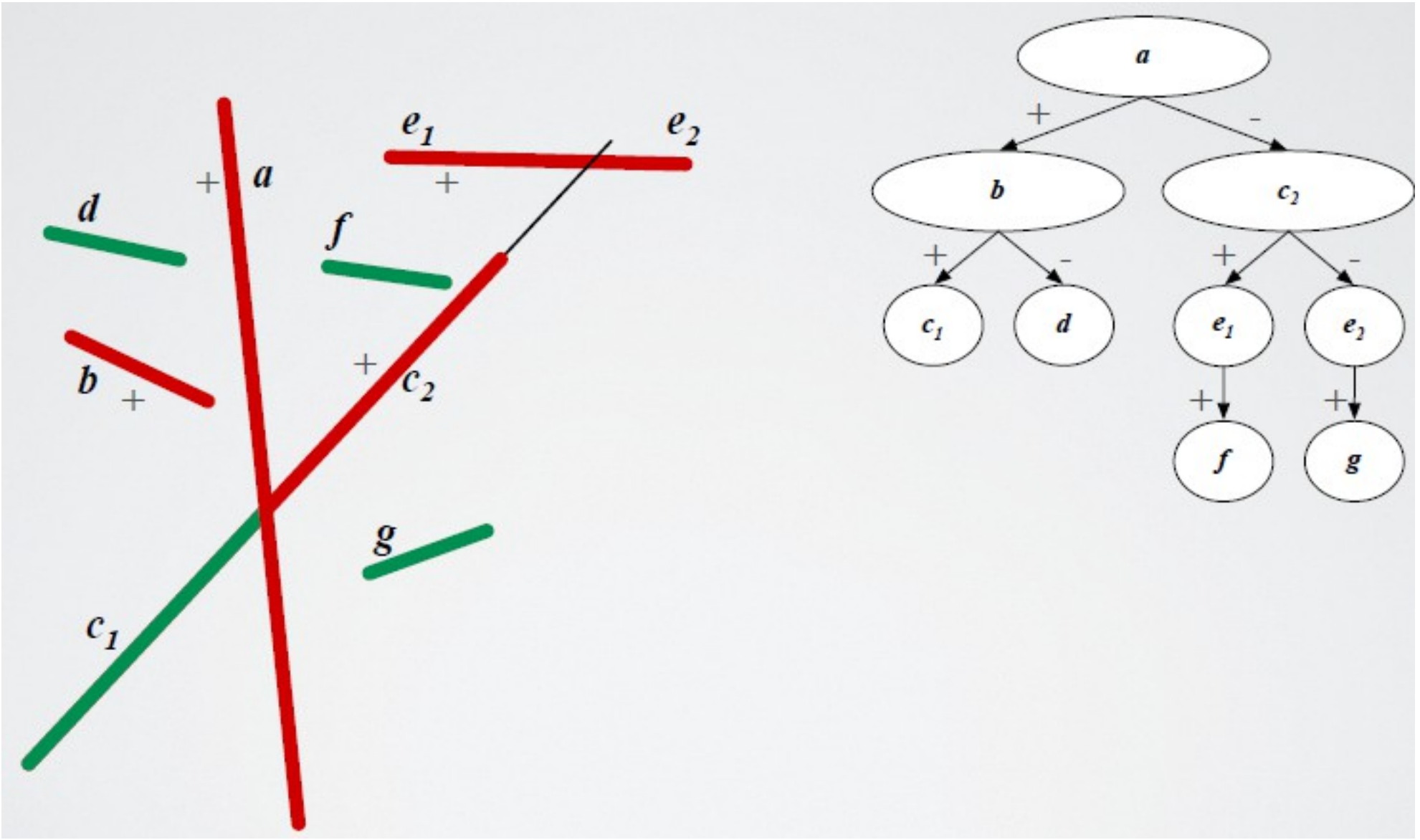
BSP Trees



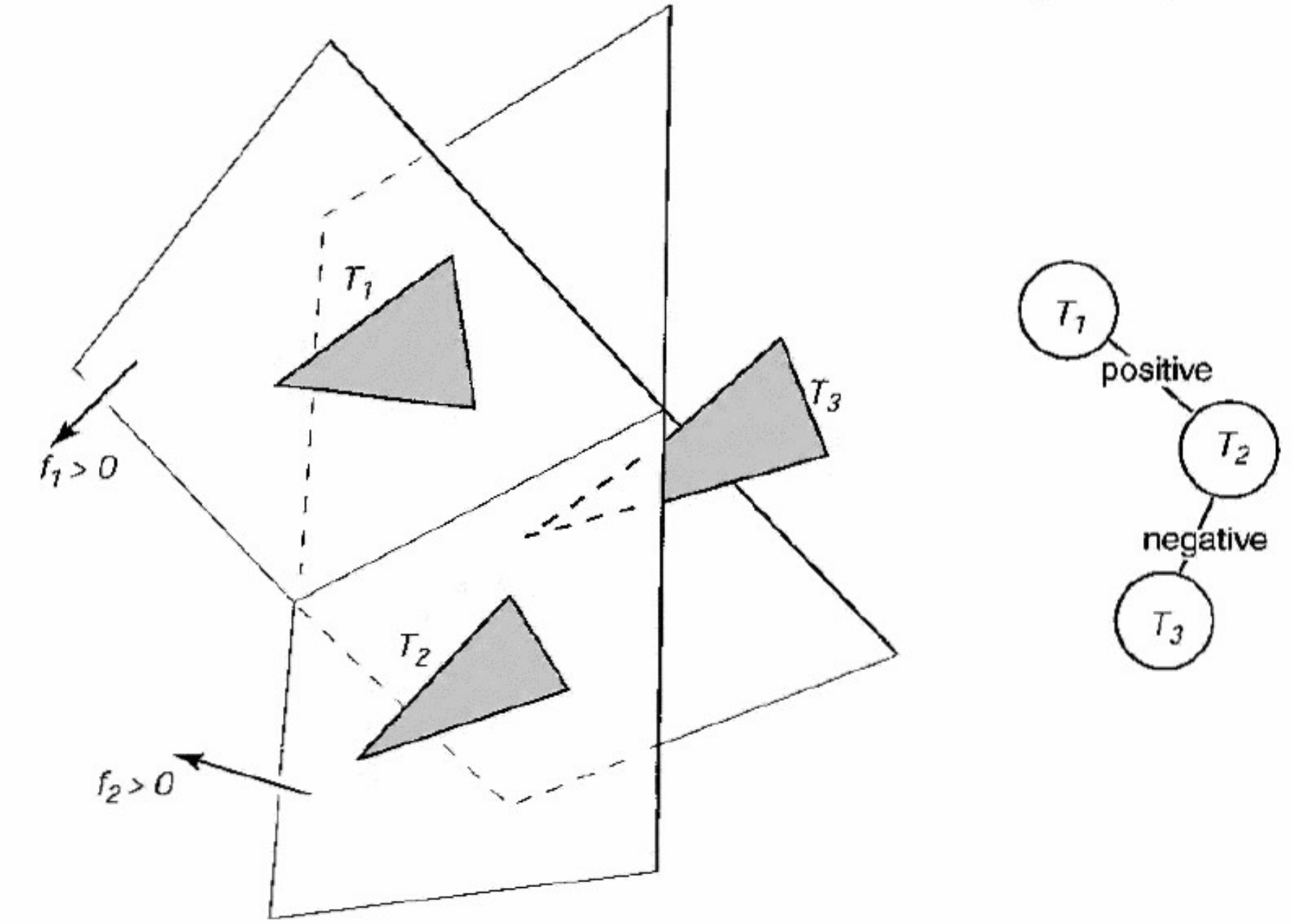
BSP Trees



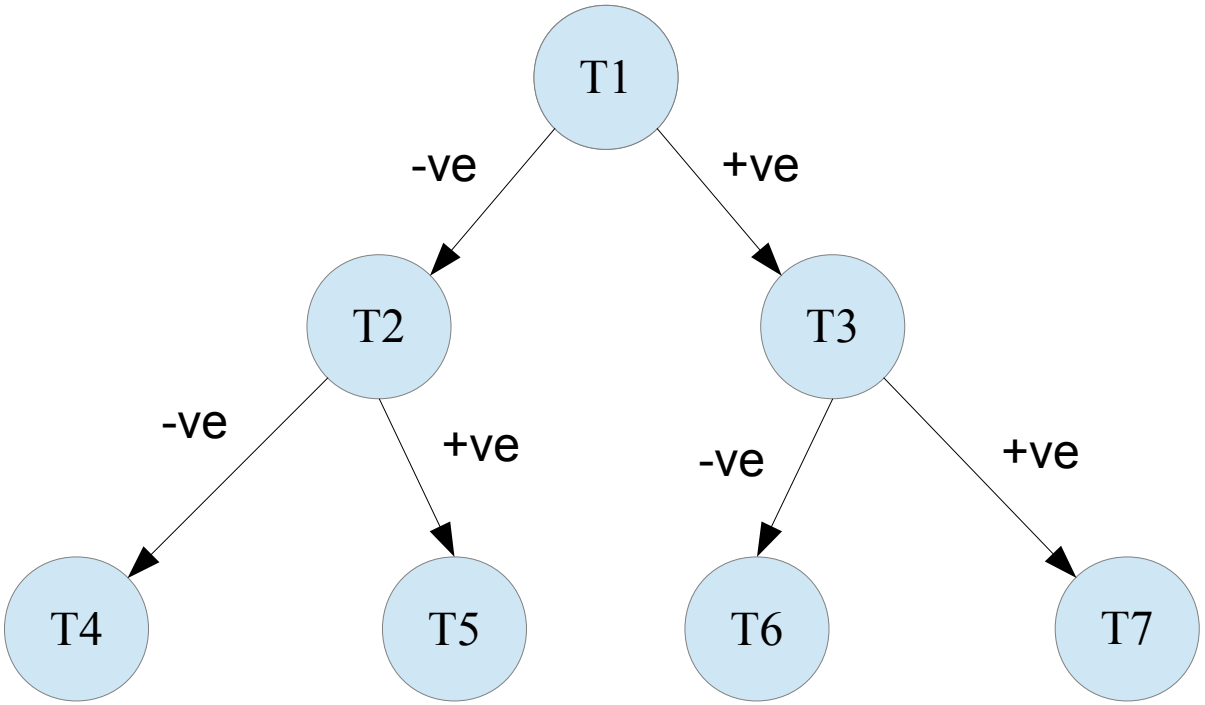
BSP Trees



BSP Trees

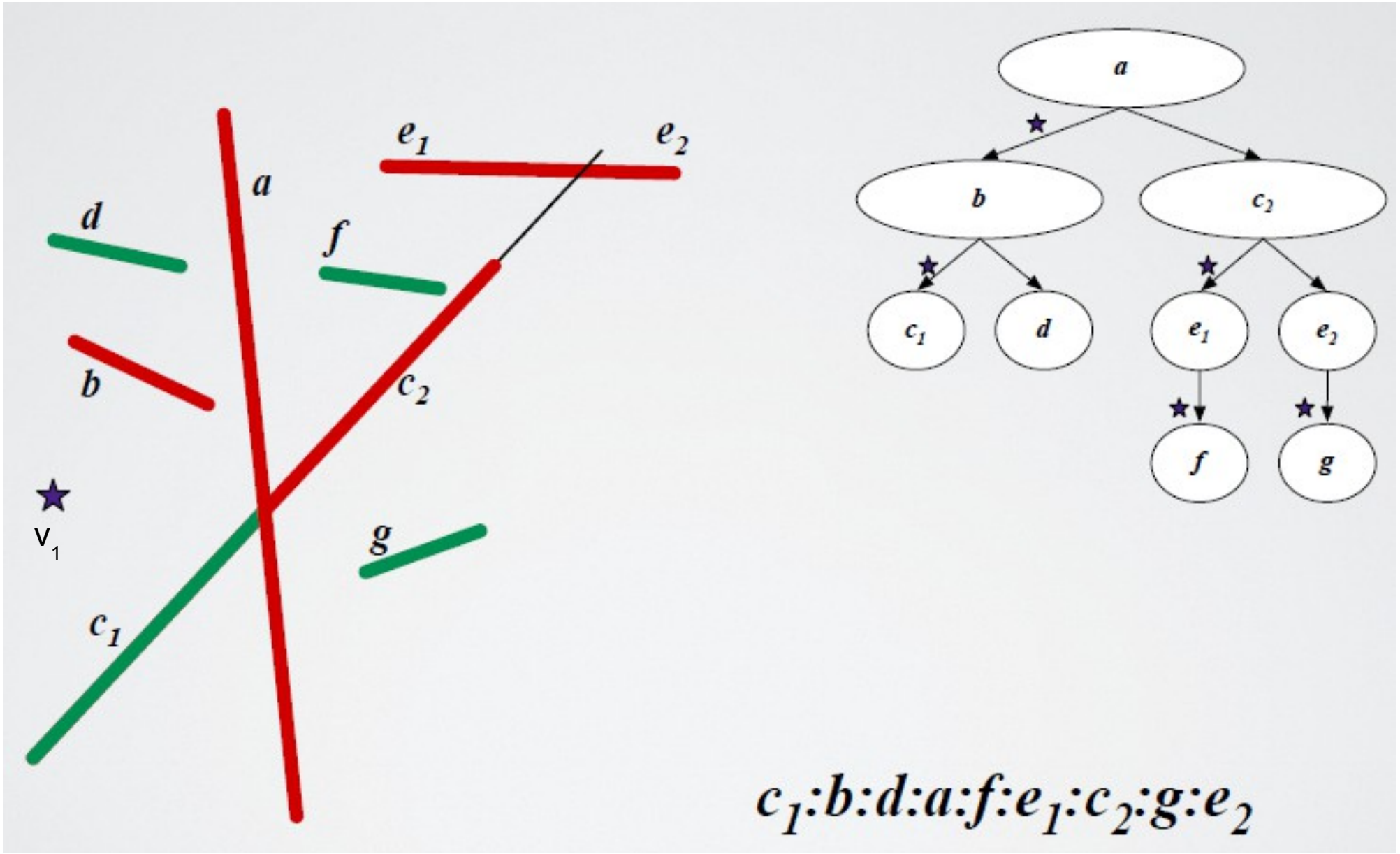


BSP Trees



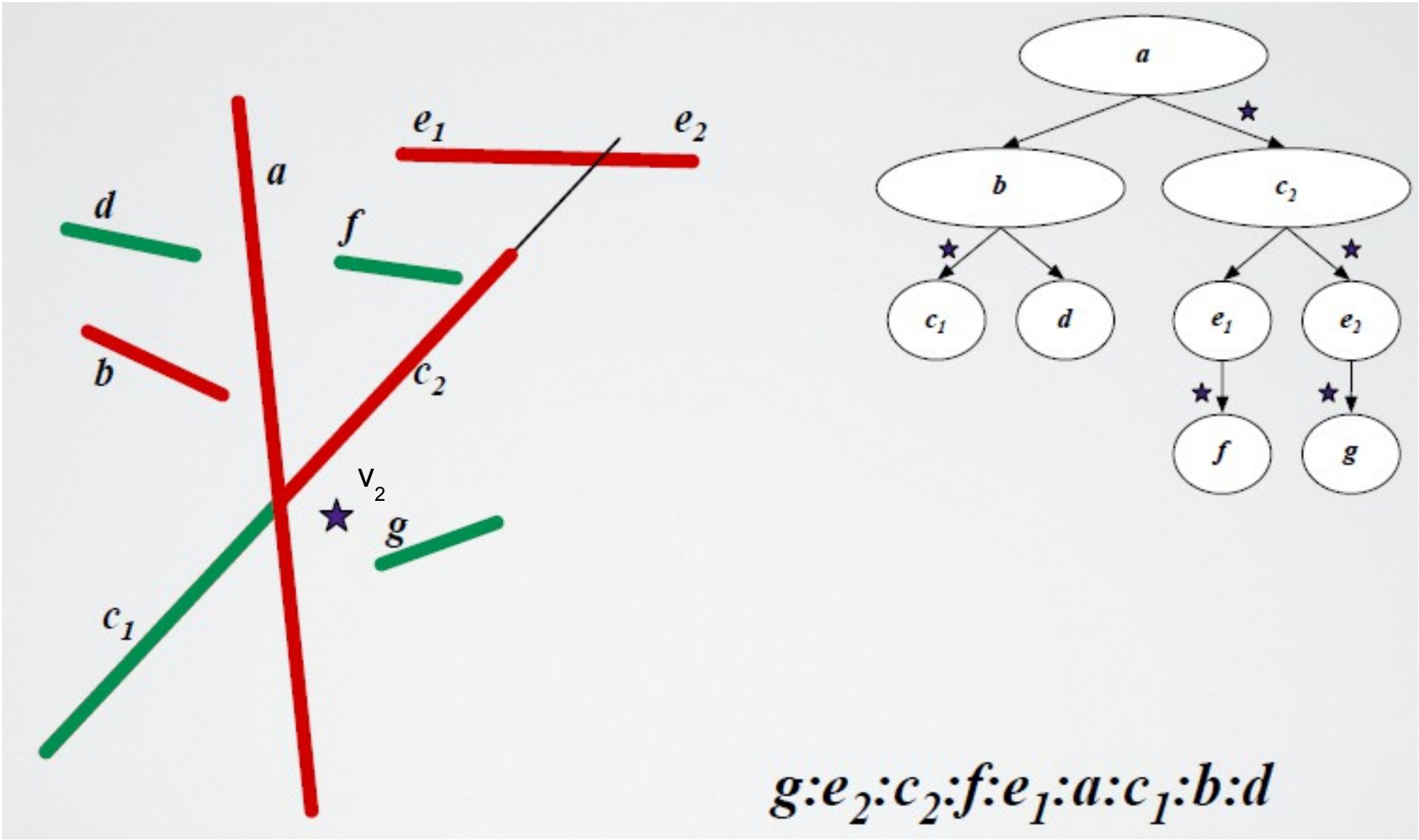
A BSP built from 7 triangles

BSP Trees



Viewing from v_1

BSP Trees



Viewing from v_2

BSP Trees

```
function draw(tree, e)
  if (tree.empty)
    return
  if (f_tree.root(e) < 0)
    draw(tree.plus, e)
    rasterize tree.root
    draw(tree.minus, e)
  else
    draw(tree.minus, e)
    rasterize tree.root
    draw(tree.plus, e)
  endif
```

Works for any viewpoint e !

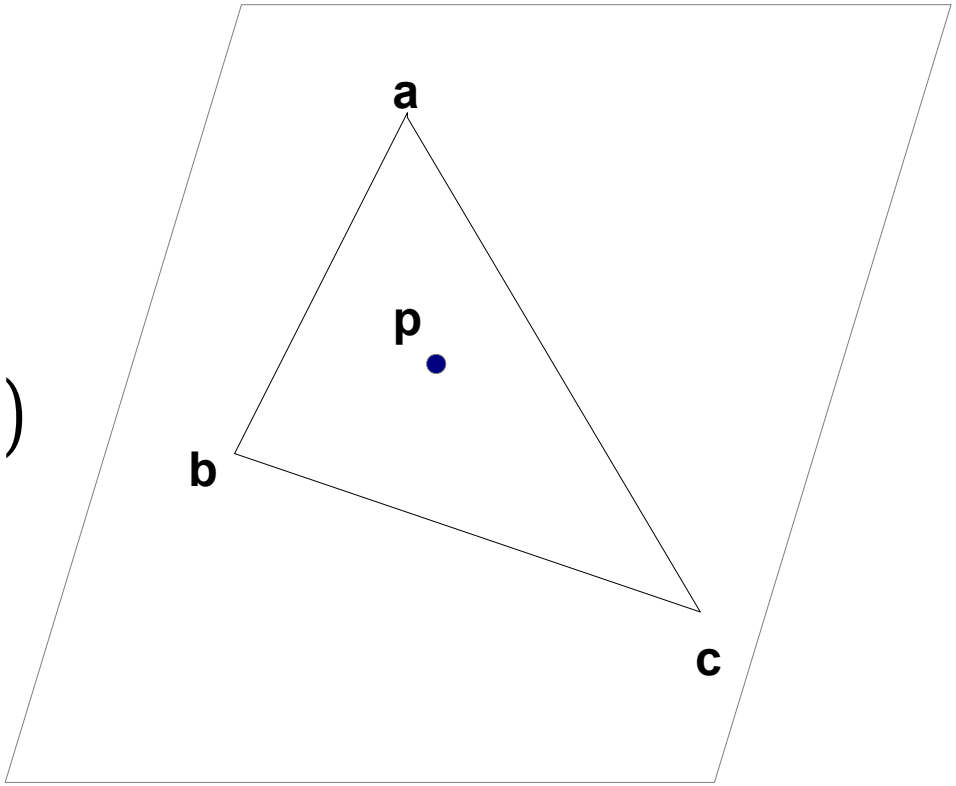
Where do we get $f_{tree.root}$?

BSP Trees

Implicit function

$$f(\boldsymbol{p}) = Ax + By + Cz + D = 0$$

Normal Vector: $\boldsymbol{n} = (\boldsymbol{b} - \boldsymbol{a}) \times (\boldsymbol{c} - \boldsymbol{a})$



$$f(\boldsymbol{p}) = \boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{a}) = 0$$

$$A = n_x \;\&\; B = n_y \;\&\; C = n_z \;\&\; D = -n \cdot a$$

How do we build the tree?

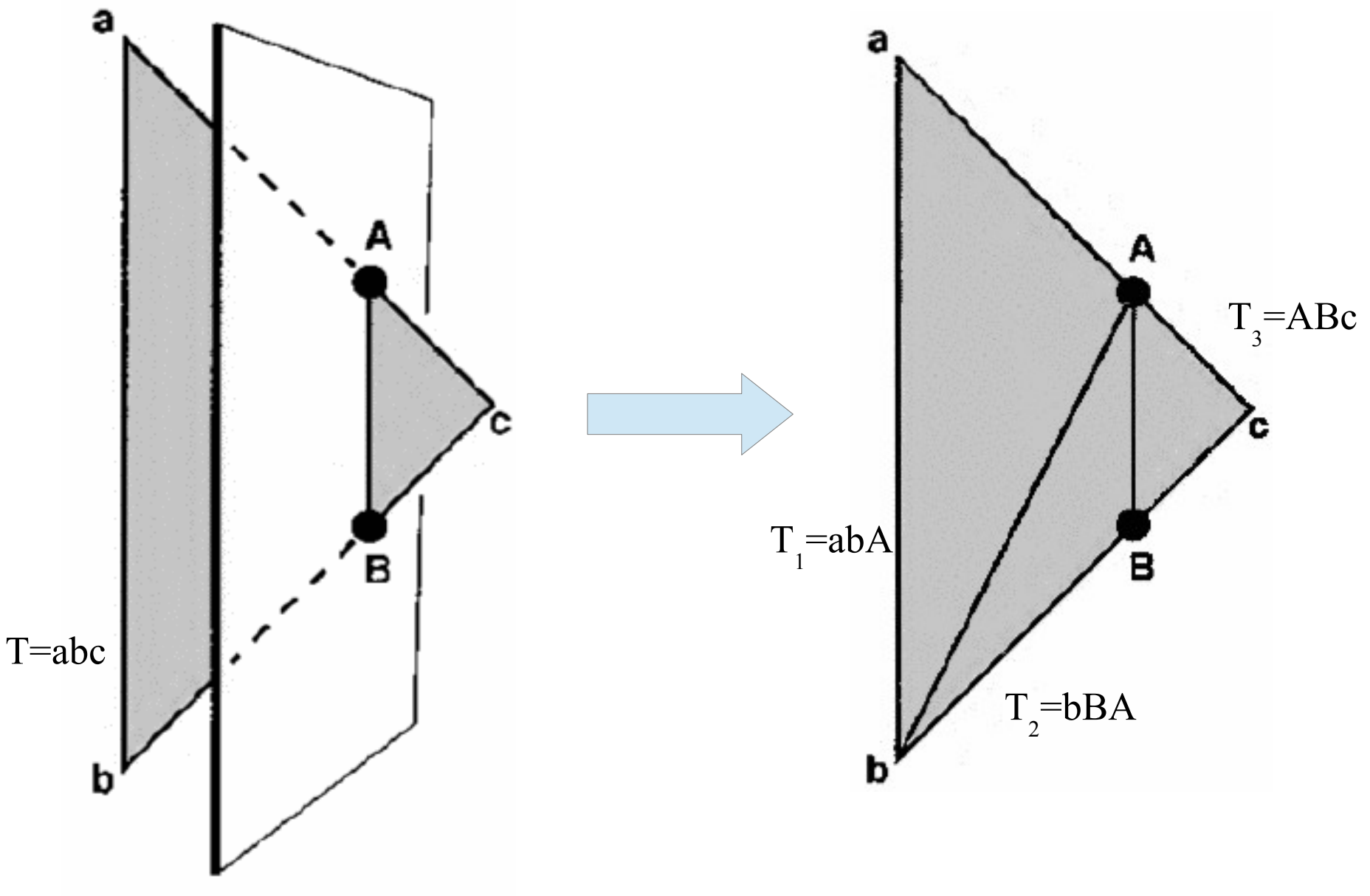
BSP Trees

```
tree.root = T1
for i = 2 .. N do
  tree.root.add(Ti)

function add(T)
if (f(a)<0 & f(b)<0 & f(c)<0) then
  tree.negative.add(T)
else if (f(a)>0 & f(b)>0 & f(c)>0) then
  tree.positive.add(T)
else
  Not possible yet !!
```

What to do when a triangle cuts the plane of another?

BSP Trees



Divide the triangle into 3 triangles !

BSP Trees

```
tree.root = T1
for i = 2 .. N do
  tree.root.add(Ti)

function add(T)
fa = f(a)
fb = f(b)
fc = f(c)
if fabs(fa)<ε then fa = 0
if fabs(fb)<ε then fb = 0
if fabs(fc)<ε then fc = 0

if (fa<=0 & fb<=0 & fc<=0) then
  tree.negative.add(T)
else if (fa>=0 & fb>=0 & fc>=0) then
  tree.positive.add(T)
else
  Divide the triangle into three and add each one
```

Why?

How?

BSP Trees

Plane equation

$$f(\boldsymbol{p}) = \boldsymbol{n} \cdot \boldsymbol{p} + D = 0$$

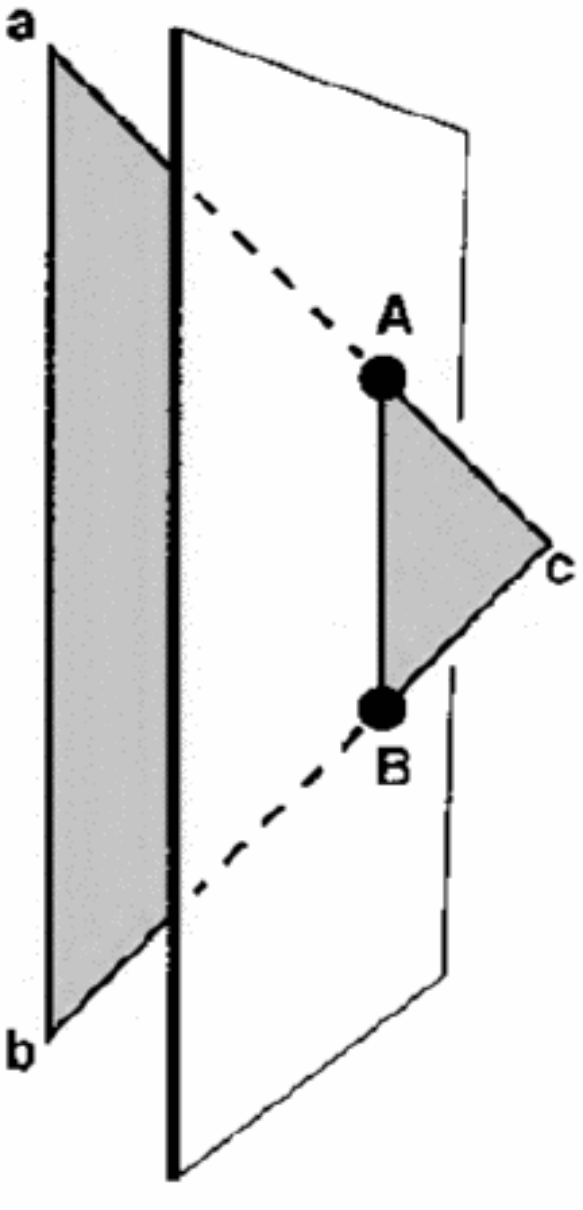
Line through $\boldsymbol{a}$ and $\boldsymbol{c}$:

$$p(t) = \boldsymbol{a} + t(\boldsymbol{c} - \boldsymbol{a})$$

$$t = \frac{-\boldsymbol{n} \cdot \boldsymbol{a} + D}{\boldsymbol{n} \cdot (\boldsymbol{c} - \boldsymbol{a})}$$

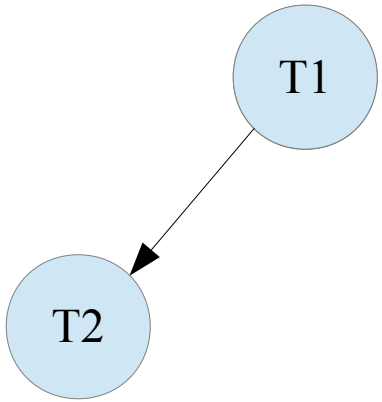
$$\boldsymbol{A} = \boldsymbol{a} + t(\boldsymbol{c} - \boldsymbol{a})$$

Similarly for B

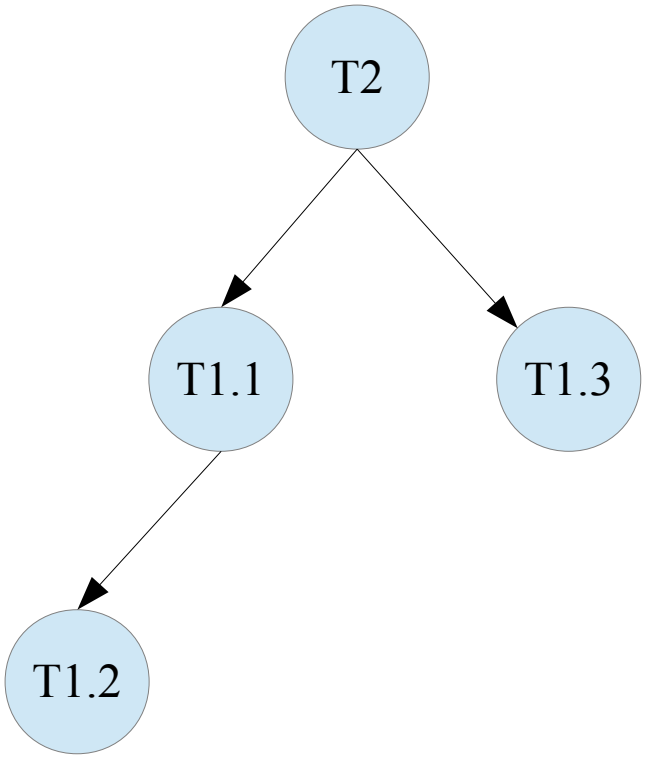


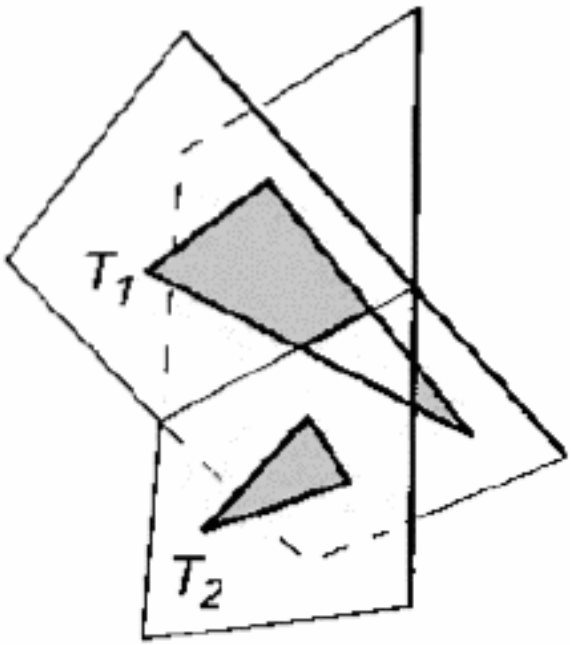
BSP Trees

T1 as root



T2 as root





Which tree is better ? How do we find the *best* tree?

Recap

- Z-Buffer
- Painter's Algorithm
- BSP Trees