

---

---

## CIS 736 Computer Graphics Lecture 6 of 42

Viewing 4: Clipping and Culling

Monday, 04 February 2008

Reading: Sections 2.4, 4.1, 4.5 Eberly 2e

Adapted with Permission

W. H. Hsu

<http://www.kddresearch.org>

CIS 636/736 Computer Graphics

Mon 04 Feb 2008

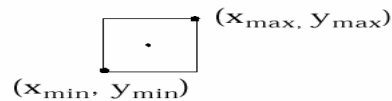
6/42

---

---

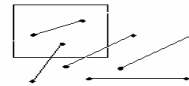
### Line Clipping

- Clipping endpoints



$x_{min} \leq x \leq x_{max}$  and  $y_{min} \leq y \leq y_{max}$  point inside  $\iff$

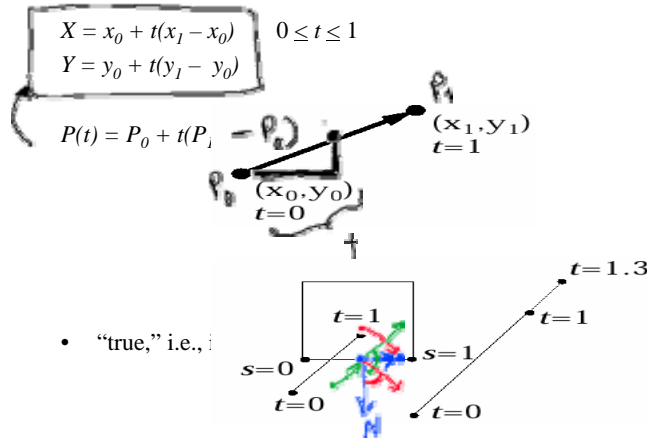
- Endpoint analysis for lines:



- if both endpoints in , can do “trivial acceptance”
  - if one endpoint is inside, one outside, must clip
  - if both endpoints out, don't know
- Brute force clip: solve simultaneous equations using  $y = mx + b$  for line and four clip edges
    - slope-intercept formula handles infinite lines only
    - doesn't handle vertical lines
- 
-

## Parametric Line Formulation For Clipping

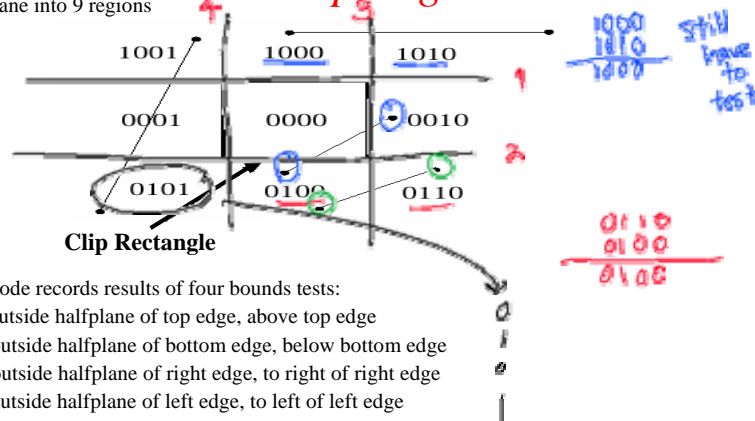
- Parametric form for line segment



- “true,” i.e.,

## Outcodes for Cohen-Sutherland Line Clipping

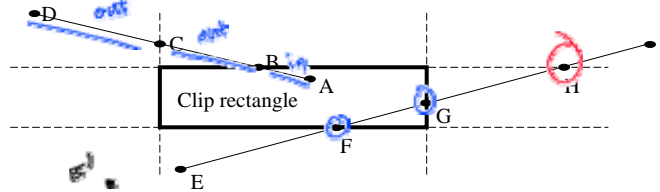
- Divide plane into 9 regions



- 4 bit outcode records results of four bounds tests:
  - First bit: outside halfplane of top edge, above top edge
  - Second bit: outside halfplane of bottom edge, below bottom edge
  - Third bit: outside halfplane of right edge, to right of right edge
  - Fourth bit: outside halfplane of left edge, to left of left edge
- Lines with  $OC_0 = 0$  and  $OC_1 = 0$  can be *trivially accepted*
- Lines lying entirely in a half plane on outside of an edge can be *trivially rejected*:  $OC_0 \& OC_1 \neq 0$ , i.e., they share an “outside” bit

## Cohen-Sutherland Algorithm

- If we can neither trivial reject nor accept, divide and conquer: subdivide line into two segments, then can T/A or T/R one or both segments:
  - use a clip edge to cut the line
  - use outcodes to choose edge that is crossed: if outcodes differ in the edge's bit, endpoints must straddle that edge
  - pick an order for checking edges: top - bottom - right - left
  - compute the intersection point; the clip edge fixes either  $x$  or  $y$ , can be substituted into the line equation
  - iterate for the newly shortened line
  - "extra" clips may happen, e.g., E-I at H

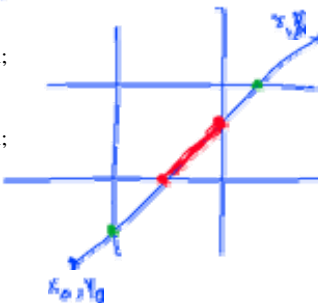


## Pseudocode for the Cohen-Sutherland Algorithm

•  $y = y_0 + \text{slope} * (x - x_0)$  and  $x = x_0 + (1/\text{slope}) * (y - y_0)$

```

ComputeOutCode(x0, y0, outcode0)
ComputeOutCode(x1, y1, outcode1)
repeat
  check for trivial reject or trivial accept
  pick the point that is outside the clip rectangle
  if TOP then
     $x = x_0 + (x_1 - x_0) * (y_{\text{max}} - y_0) / (y_1 - y_0)$ ;  $y = y_{\text{max}}$ ;
  else if BOTTOM then
     $x = x_0 + (x_1 - x_0) * (y_{\text{min}} - y_0) / (y_1 - y_0)$ ;  $y = y_{\text{min}}$ ;
  else if RIGHT then
     $y = y_0 + (y_1 - y_0) * (x_{\text{max}} - x_0) / (x_1 - x_0)$ ;  $x = x_{\text{max}}$ ;
  else if LEFT then
     $y = y_0 + (y_1 - y_0) * (x_{\text{min}} - x_0) / (x_1 - x_0)$ ;  $x = x_{\text{min}}$ ;
  end {calculate the line segment}
  if (outcode = outcode0) then
     $x_0 = x$ ;  $y_0 = y$ ; ComputeOutCode(x0, y0, outcode0)
  else
     $x_1 = x$ ;  $y_1 = y$ ; ComputeOutCode(x1, y1, outcode1)
  end {Subdivide}
until done
    
```





## C-B/L-B Param. Line Clipping-2

Now we can solve for the value of  $t$  at the Intersection of  $P_0 P_1$  with the edge  $E_i$ :

$$N_i \cdot [P(t) - P_{E_i}] = 0$$

First, substitute for  $P(t)$ :

$$N_i \cdot [P_0 + (P_1 - P_0)t - P_{E_i}] = 0$$

Next, group terms and distribute the dot product:

$$N_i \cdot [P_0 - P_{E_i}] + N_i \cdot [P_1 - P_0]t = 0$$

Let  $D$  be the vector from  $P_0$  to  $P_1 = (P_1 - P_0)$ , and solve for  $t$ :

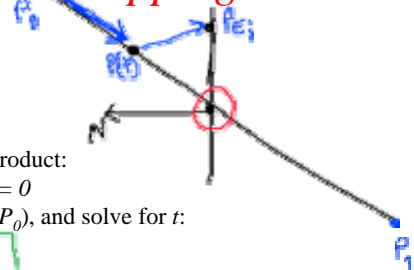
$$t = \frac{N_i \cdot [P_0 - P_{E_i}]}{-N_i \cdot D}$$

Note that this gives a valid value of  $t$  only if the denominator of the expression is nonzero. For this to be true, it must be the case that

$N_i \neq 0$  (that is, the normal should not be 0;  
this could occur only as a mistake)

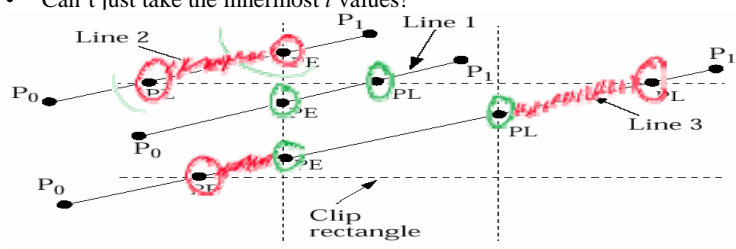
$D \neq 0$  (that is,  $P_1 \neq P_0$ )

$N_i \cdot D \neq 0$  (edge  $E_i$  and line  $D$  are not parallel; if they are, no intersection).  
The algorithm checks these conditions.



## C-B/L-B Param. Line Clipping-3

- Eliminate  $t$ s outside  $[0,1]$  on the line
- Which remaining  $t$ s produce interior intersections?
- Can't just take the innermost  $t$  values!



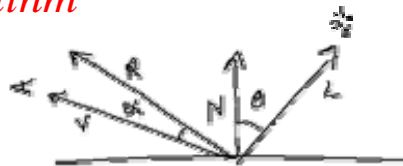
- Move from  $P_0$  to  $P_1$  for a given edge, just before crossing: if  $N_i \cdot D < 0$  Potentially Entering (PE), if  $N_i \cdot D > 0$  Potentially Leaving (PL)
- Pick inner PE, PL pair:  $t_E$  for  $P_{PE}$  with max  $t$ ,  $t_L$  for  $P_{PL}$  with min  $t$ , and  $t_E > 0, t_L < 1$ .
- If  $t_L < t_E$ , no intersection

## Pseudocode for Cyrus-Beck/ Liang-Barsky Line Clipping Algorithm

```

precalculate  $\vec{N}_i$  and select  $P_{E_i}$  for each edge;
for each line segment to be clipped
  if  $P_1 = P_0$  then
    line is degenerate so clip as a point;
  else
    begin
       $t_E = 0$ ;  $t_L = 1$ ;
      for each candidate intersection with a clip edge
        if  $\vec{N}_i \cdot \vec{D} \neq 0$  then {Ignore edges parallel to line}
          begin
            calculate  $t$ :  $t$  of line and clip edge intersection
            use sign of  $\vec{N}_i \cdot \vec{D}$  to categorize as PE or PL;
            if PE then  $t_E = \max(t_E, t)$ ;
            if PL then  $t_L = \min(t_L, t)$ ;
          end
        if  $t_E > t_L$  then
          return nil
        else
          return  $P(t_E)$  and  $P(t_L)$  as true clip intersections
      end
    end
  end

```



Diffuse:  $\text{dot}(\vec{0}) = \vec{N} \cdot \vec{L}$   
 Specular:  $\text{cos}(\theta) = \frac{\vec{R} \cdot \vec{V}}{|\vec{R}| |\vec{V}|}$

Andries van Dam

September 25, 2003

2D Clipping

11/14

## Calculations for Parametric Line Clipping for Upright Clip Rectangle (1/2)

- $\vec{D} = P_1 - P_0 = (x_1 - x_0, y_1 - y_0)$
- Leave  $P_{E_i}$  as an arbitrary point on the clip edge; it's a free variable and drops out

Calculations for Parametric Line Clipping Algorithm

Clip Edge <sub>i</sub>	Normal $\vec{N}_i$	$P_{E_i}$	$P_0 - P_{E_i}$	$t = \frac{\vec{N}_i \cdot (P_0 - P_{E_i})}{\vec{N}_i \cdot \vec{D}}$
left: $x = x_{\min}$	$(-1, 0)$	$(x_{\min}, y)$	$(x_0 - x_{\min}, y_0 - y)$	$\frac{-(x_0 - x_{\min})}{(x_1 - x_0)}$
right: $x = x_{\max}$	$(1, 0)$	$(x_{\max}, y)$	$(x_0 - x_{\max}, y_0 - y)$	$\frac{-(x_0 - x_{\max})}{(x_1 - x_0)}$
bottom: $y = y_{\min}$	$(0, -1)$	$(x, y_{\min})$	$(x_0 - x, y_0 - y_{\min})$	$\frac{-(y_0 - y_{\min})}{(y_1 - y_0)}$
top: $y = y_{\max}$	$(0, 1)$	$(x, y_{\max})$	$(x_0 - x, y_0 - y_{\max})$	$\frac{-(y_0 - y_{\max})}{(y_1 - y_0)}$

Andries van Dam

September 25, 2003

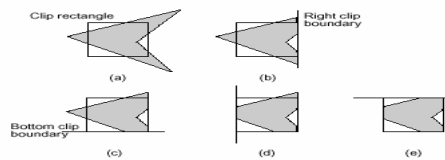
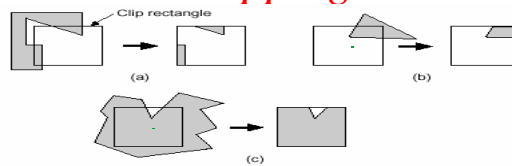
2D Clipping

12/14

## Calculations for Parametric Line Clipping for Upright Clip Rectangle (2/2)

- Examine  $t$ :
  - Numerator is just the directed distance to an edge; sign corresponds to OC
  - Denominator is just the horizontal or vertical projection of the line,  $dx$  or  $dy$ ; sign determines PE or PL for a given edge
  - Ratio is constant of proportionality: “how far over” from  $P_0$  to  $P_j$  intersection is relative to  $dx$  or  $dy$

## Sutherland-Hodgman Polygon Clipping

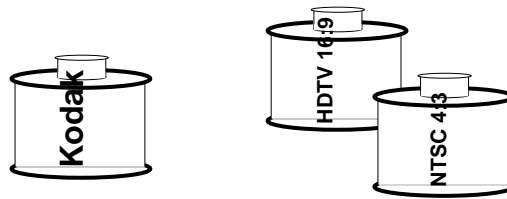


---

---

## Aspect Ratio

- Analogous to the size of film used in a camera
- Determines proportion of width to height of image displayed on screen
- Square viewing window has aspect ratio of 1:1
- Movie theater “letterbox” format has aspect ratio of 2:1
- NTSC television has an aspect ratio of 4:3, and HDTV is 16:9



---

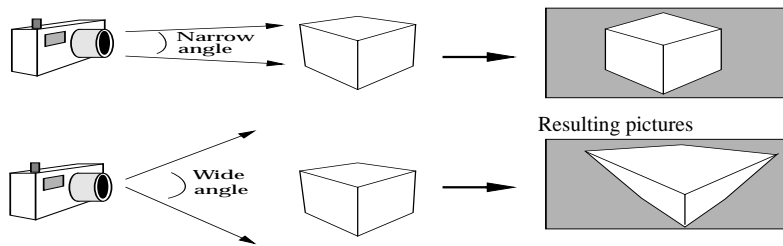
---

## View Angle (1/2)

- Determines amount of perspective distortion in picture, from none (parallel projection) to a lot (wide-angle lens)
- In a *frustum*, two viewing angles: width and height angles. We specify *Height angle*, and get the *Width angle* from (*Aspect ratio \* Height angle*)
- Choosing *View angle* analogous to photographer choosing a specific type of lens (e.g., a wide-angle or telephoto lens)

## View Angle (2/2)

- Lenses made for distance shots often have a nearly parallel viewing angle and cause little perspective distortion, though they foreshorten depth
- Wide-angle lenses cause a lot of perspective distortion



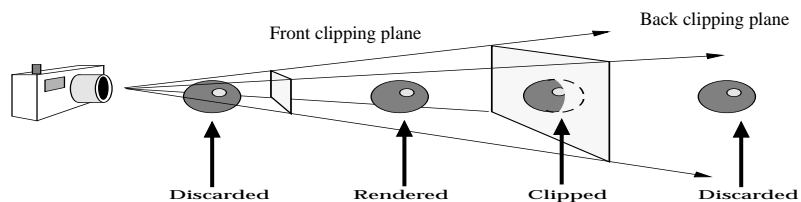
Andries van Dam

September 16, 2003

3D Viewing II 11/21

## Front and Back Clipping Planes (1/3)

- Volume of space between *Front* and *Back clipping planes* defines what camera can see
- Position of planes defined by distance along *Look vector*
- Objects appearing outside of view volume don't get drawn
- Objects intersecting view volume get clipped



Andries van Dam

September 16, 2003

3D Viewing II 12/21

## Front and Back Clipping Planes (2/3)

- Reasons for *Front (near) clipping plane*:
- Don't want to draw things too close to the camera
  - would block view of rest of scene
  - objects would be prone to distortion
- Don't want to draw things behind camera
  - wouldn't expect to see things behind the camera
  - in the case of the perspective camera, if we decided to draw things behind the camera, they would appear upside-down and inside-out because of perspective transformation
- Reasons for *Back (far) clipping plane*:
- Don't want to draw objects too far away from camera
  - distant objects may appear too small to be visually significant, but still take long time to render
  - by discarding them we lose a small amount of detail but reclaim a lot of rendering time
  - alternately, the scene may be filled with many significant objects; for visual clarity, we may wish to declutter the scene by rendering those nearest the camera and discarding the rest

Andries van Dam

September 16, 2003

3D Viewing II 13/21

## Front and Back Clipping Planes (3/3)

- Have you ever played a video game and all of the sudden some object pops up in the background (e.g. a tree in a racing game)? That's the object coming inside the far clip plane.
- The old hack to keep you from noticing the pop-up is to add fog in the distance. A classic example of this is from *Turok: Dinosaur Hunter*



- FYFH 97:
1. Conservative visible surface plot
    - 3-D depth sort
    - z-buffer
  2. Object precision (handing lighting)
  3. Image precision
    - z - buffer

- Now all you notice is fog and how little you can actually see. This practically defeats the purpose of an outdoor environment! And you can *still* see pop-up from time to time.
- Thanks to fast hardware and level of detail algorithms, we can push the far plane back now and fog is much less prevalent
- Putting the near clip plane as far away as possible helps Z precision. Sometimes in a game you can position the camera in the right spot so that the front of an object gets clipped letting

Andries van Dam

September 16, 2003

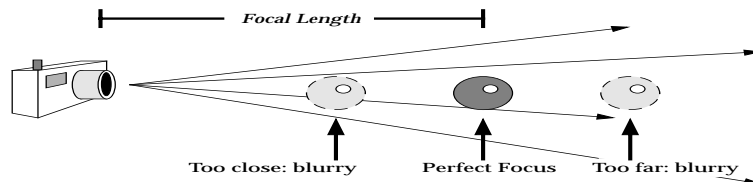
3D Viewing II 14/21

---

---

## Focal Length

- Some camera models take a *Focal length*
- *Focal Length* is a measure of ideal focusing range; approximates behavior of real camera lens
- Objects at distance of *Focal length* from camera are rendered in focus; objects closer or farther away than *Focal length* get blurred
- *Focal length* used in conjunction with clipping planes
- Only objects within view volume are rendered, whether blurred or not. Objects outside of view volume still get discarded



---

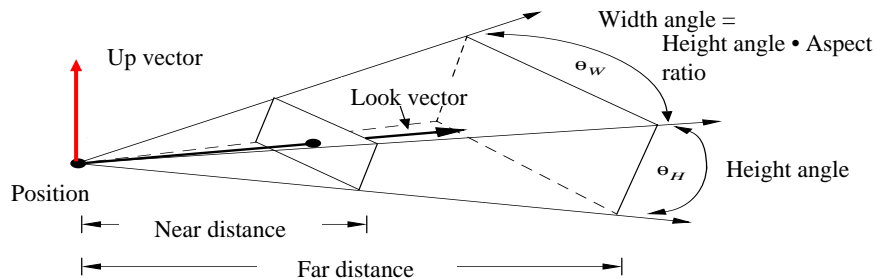
---

## View Volume Specification

- From *Position*, *Look vector*, *Up vector*, *Aspect ratio*, *Height angle*, *Clipping planes*, and (optionally) *Focal length* together specify a truncated view volume
- Truncated view volume is a specification of bounded space that camera can “see”
- 2D view of 3D scene can be computed from truncated view volume and projected onto film plane
- Truncated view volumes come in two flavors: parallel and perspective

## Truncated View Volume (Frustum) for Perspective Projection

- Removes objects too far from *Position*, which otherwise would merge into “blobs”
- Removes objects too close to *Position* (would be excessively distorted)



## Sources

- Carlbom, Ingrid and Paciorek, Joseph, “Planar Geometric Projections and Viewing Transformations,” *Computing Surveys*, Vol. 10, No. 4 December 1978
- Kemp, Martin, *The Science of Art*, Yale University Press, 1992
- Mitchell, William J., *The Reconfigured Eye*, MIT Press, 1992
- Foley, van Dam, et. al., *Computer Graphics: Principles and Practice*, Addison-Wesley, 1995
- Wernecke, Josie, *The Inventor Mentor*, Addison-Wesley, 1994