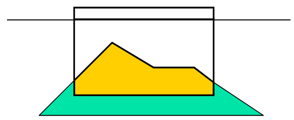
Line and Polygon Clipping

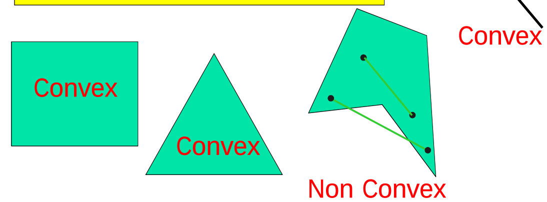
Chapter 7

Clipping



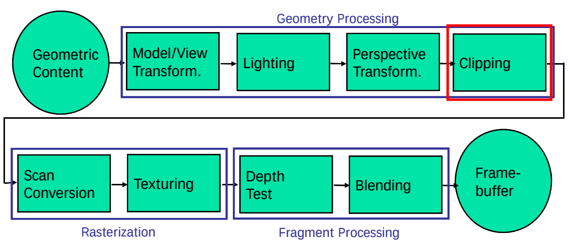
Convexity

Set $C \subseteq R^d$ is ***convex*** if for any two points $p,q \in C$ and any $\alpha \in [0,1]$, $\alpha p+(1-\alpha)q \in C$



2D Projection of **convex** 3D shape is **convex**

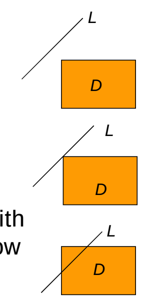
Rendering Pipeline



- Discard geometry outside viewport window

Explicit Solution: Line Segments

- *Intersection* of convex regions is convex
  - Why?
- *L* & *D* are *convex* - intersection is convex
  - **single** connected segment of *L*
- Clipping uses intersections of *L* with four boundary segments of window *D*



Line/Polygon Clipping (2D)

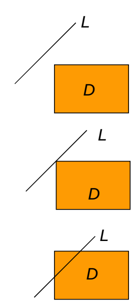
Problem:
Given a 2D line/polygon and a window, clip the line/polygon to their regions that are *inside* the window.

- Objectives
  - Efficiency
  - (Parallelization)
- Two approaches
  - Explicit (continuous setting)
  - Implicit (discrete setting) – part of scan conversion

Basic Method

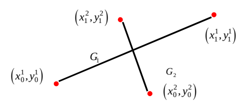
Clip($P_0, P_1, x_{min}, x_{max}, y_{min}, y_{max}$)
if (P_0 and P_1 inside window) then draw(P_0, P_1);
test if segment (P_0, P_1) intersects any of the edges
if not, return;
else let P_2 be the first intersection found
 Clip($P_0, P_2, x_{min}, x_{max}, y_{min}, y_{max}$);
 Clip($P_2, P_1, x_{min}, x_{max}, y_{min}, y_{max}$);
end

- Works, but inefficient for lines OUTSIDE D
  - Four intersection tests
- Note: need special care for vertices ON window edges



Line and Polygon Clipping

Segment-Segment Intersection



$$G_1=\begin{cases}x^1(t)=x_0^1+(x_1^1-x_0^1)t\\ y^1(t)=y_0^1+(y_1^1-y_0^1)t\end{cases}\quad t\in[0,1]\qquad G_2=\begin{cases}x^2(r)=x_0^2+(x_1^2-x_0^2)r\\ y^2(r)=y_0^2+(y_1^2-y_0^2)r\end{cases}\quad r\in[0,1]$$

Intersection: *x* & *y* values equal in both representations - two linear equations in two unknowns (*r,t*)
test if resulting r & t are inside the [0,1] range

$$x_0^1+(x_1^1-x_0^1)t=x_0^2+(x_1^2-x_0^2)r$$
$$y_0^1+(y_1^1-y_0^1)t=y_0^2+(y_1^2-y_0^2)r$$

Cohen-Sutherland Algorithm

Purpose:
Fast treatment of line segments that are trivially inside/outside window.

0101	0100	0110
0001	0000	0010
1001	1000	1010

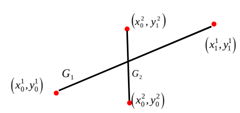
P = (*x,y*) - point to be classified against window *D*

Idea: Assign to P a binary code consisting of a bit for each edge of *D*, using lookup table:

bit	1	0
1	$y < y_{\min}$	$y \geq y_{\min}$
2	$y > y_{\max}$	$y \leq y_{\max}$
3	$x > x_{\max}$	$x \leq x_{\max}$
4	$x < x_{\min}$	$x \geq x_{\min}$

10

Intersection with axis-aligned lines

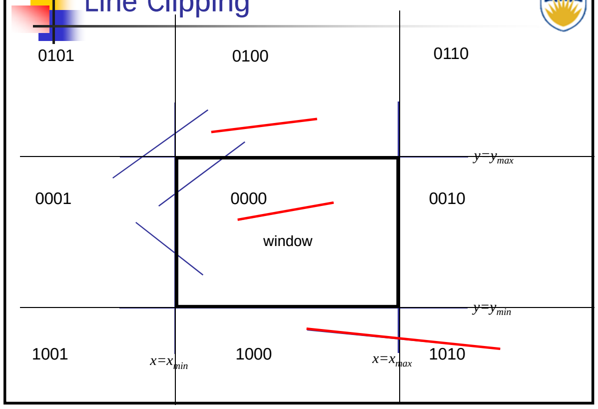


$$G_1=\begin{cases}x^1(t)=x_0^1+(x_1^1-x_0^1)t\\ y^1(t)=y_0^1+(y_1^1-y_0^1)t\end{cases} t\in[0,1],\ G_2=\begin{cases}x^2(r)=x_0^2\\ y^2(r)=y_0^2+(y_1^2-y_0^2)r\end{cases} r\in[0,1]$$

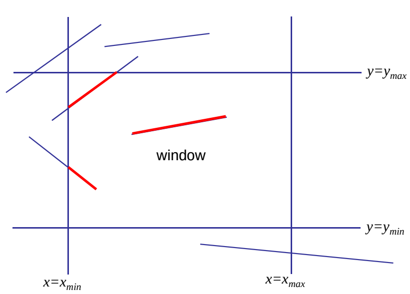
Intersection: *x* & *y* values equal in both representations - two linear equations in two unknowns (*r,t*)

$$x_0^1+(x_1^1-x_0^1)t=x_0^2$$
$$t=\frac{x_0^2-x_0^1}{x_1^1-x_0^1},\ \text{if } t<0 \text{ or } t>1 \text{ no intersecti on}$$
$$y_0^1+(y_1^1-y_0^1)t=y_0^2+(y_1^2-y_0^2)r,\ \text{(relevant only for segments)}$$

Line Clipping



Line Clipping

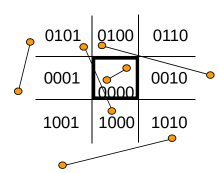


Cohen-Sutherland Algorithm (cont'd)

Given *L* from (x_0,y_0) to (x_1,y_1) & rectangle D.

If bitwise **and** of the codes of (x_0,y_0) and (x_1,y_1) is not zero, or the bitwise **or** is zero,

then *L* can be trivially handled (it is either totally outside or totally inside D).

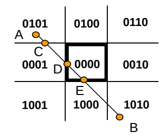


Why?

12

Cohen-Sutherland Algorithm (cont'd)

C-S-Clip($P_0 = (x_0, y_0), P_1 = (x_1, y_1), x_{min}, x_{max}, y_{min}, y_{max}$)
(assumes $x_0 < x_1$)
$C_0 \Leftarrow$ code (P_0); $C_1 \Leftarrow$ code (P_1);
if ((C_0 and C_1) != 0) then return;
if ((C_0 or C_1) == 0) then draw(P_0, P_1);
else if (OutsideWindow(P_0)) then
begin
$Edge \Leftarrow$ Window boundary of leftmost non-zero bit of C_0;
$P_2 \Leftarrow \overline{P_0, P_1} \cap Edge$;
C-S-Clip($P_2, P_1, x_{min}, x_{max}, y_{min}, y_{max}$);
end
else
$Edge \Leftarrow$ Window boundary of leftmost non-zero bit of C_1;
$P_2 \Leftarrow \overline{P_0, P_1} \cap Edge$;
C-S-Clip($P_0, P_2, x_{min}, x_{max}, y_{min}, y_{max}$);
end



AB → CB → DB → DE

bit	1	0
1	$y < y_{min}$	$y \geq y_{min}$
2	$y > y_{max}$	$y \leq y_{max}$
3	$x > x_{max}$	$x \leq x_{max}$
4	$x < x_{min}$	$x \geq x_{min}$

13

Cohen-Sutherland Algorithm for convex polygons

C-S-Clip(poly = $P_0, ..., P_n, x_{min}, x_{max}, y_{min}, y_{max}$)
for i = 1 to n $C_i \Leftarrow$ code (P_i);
if ((C_0 and C_1 and ... and C_n) != 0) then return;
if ((C_0 or C_1 or ... or C_n) == 0) then draw(poly);
else
for i = 1 to n if (OutsideWindow(P_i)) then
begin
$Edge \Leftarrow$ Window boundary of leftmost non-zero bit of C_i;
$P_{i-1,i} \Leftarrow \overline{P_{i-1}, P_i} \cap Edge$;
$P_{i,i+1} \Leftarrow \overline{P_i, P_{i+1}} \cap Edge$;
C-S-Clip($P_0, ..., P_{i-1}, P_{i-1,i}, P_{i,i+1} P_{i+1}, ..., P_n, x_{min}, x_{max}, y_{min}, y_{max}$);
end

0101	0100	0110
0001	0000	0010
1001	1000	1010

bit	1	0
1	$y < y_{min}$	$y \geq y_{min}$
2	$y > y_{max}$	$y \leq y_{max}$
3	$x > x_{max}$	$x \leq x_{max}$
4	$x < x_{min}$	$x \geq x_{min}$

16

3D clipping

- Determine portion of line inside axis-aligned box (viewing frustum in NDC)
- Simple extension of 2D algorithms
- After projection transform
  - clipping volume always the same
    - xmin=ymin=zmin= -1, xmax=ymax=zmax= 1
  - boundary lines become boundary planes
    - **but bit-codes still work the same way**

14

Triangle Clipping

- How does intersection of rectangle & triangle looks like?
  - How many sides?
- How to expand clipping to triangles?
  - Hint: it is convex
  - Will sketch on the board...

15