

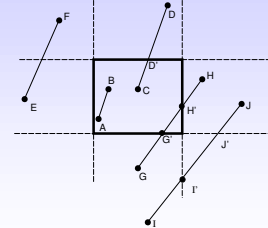
Computación Gráfica

Clipping

- Prof. María Cecilia Rivara
- mcrivara@dcc.uchile.cl
- Semestre 2004/2

MCRivara/CG2004/2

Clipping de líneas contra rectángulos (en ambiente raster)



MCRivara/CG2004/2

2

Algoritmo de Cohen-Sutherland

Ideas básicos:

- (1) Test inicial para detectar casos triviales y evitar cálculo de intersecciones
Se chequean pares de puntos extremos
- (2) Para casos no triviales se realizan chequeos sobre regiones

MCRivara/CG2004/2

3

Casos triviales de rechazo

Rectángulo definido por puntos (x_{min}, y_{min}) , (x_{max}, y_{max})

- Ambos extremos a la izquierda de x_{min} (línea EF)
- Ambos extremos a la derecha de x_{max}
- Ambos extremos bajo y_{min}
- Ambos extremos por arriba de y_{max}

MCRivara/CG2004/2

4

Test triviales de aceptación / rechazo

- Se divide el plano en 9 regiones
- A cada región se asigna un código de 4 bits

1001	1000	1010
0001	0000	0010
0101	0100	0110

Primer bit es 1 si $y > y_{max}$

Segundo bit es 1 si $y < y_{min}$

Tercer bit es 1 si $x > x_{max}$

Cuarto bit es 1 si $x < x_{min}$

MCRivara/CG2004/2

5

Algoritmo de Cohen-Sutherland

- Se usan los códigos binarios de los puntos extremos E1 E2 para decidir casos triviales.
 - Si ambos códigos son 0000, la línea E1 E2 visible
 - Si $(\text{cod}(E1) \text{ AND } \text{cod}(E2)) \neq 0000$ la línea es trivialmente descartada (no visible).
- Sin no hay descarte trivial es necesario iterar calculando las intersecciones.

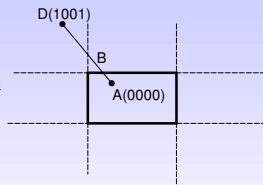
MCRivara/CG2004/2

6

Ejemplo caso no trivial (1 iteración)

Iteración 1

- Se elige D punto fuera de la ventana
- D está arriba del lado superior y a la izquierda del lado izquierdo de la ventana
- Se usa el orden de los códigos para calcular primera intersección (en este caso con arista superior)
- Se calcula intersección B(0000) con arista superior



Iteración 2

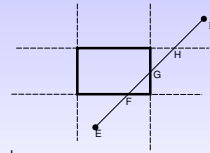
- Se descarta AB por aceptación trivial

MCRivara/CG2004/2

7

Ejemplo caso no trivial con varias iteraciones

- Iteración 1 (proceso de EI)
E(0100) se elige como punto exterior, se intersecta EI con lado de abajo encontrando F(0000) y se clipa EI a FI.
- Iteración 2 (proceso de FI)
FI no es descartado trivialmente, I(1010) se elige como punto exterior. Se intersecta FI con lado superior encontrado H(0010) y se clipa FI a FH
- Iteración 3 (proceso de FH)
se clipa contra lado derecho obteniendo FG
- Iteración 4 (proceso de FG)
aceptado trivialmente



MCRivara/CG2004/2

8

```

procedure CohenSutherlandLineClipAndDraw (
  x0, y0, x1, y1, xmin, xmax, ymin, ymax: real; value: integer);
{Cohen-Sutherland clipping algorithm for line P0 = (x0, y0) to P1 = (x1, y1) and clip
rectangle with diagonal from (xmin, ymin) to (xmax, ymax).}
type
  edge = (LEFT, RIGHT, BOTTOM, TOP);
  outside = set of edge;
var
  accept, done: boolean;
  outside0, outside1, outsideOut: outside;
  {Outside0 for P0, P1 and whichever point lies outside the clip rectangle}
  x: real;
  procedure CompOutCode (x, y: real; var code: outside);
  {Compute outcode for the point (x, y)}
  begin
    code := [];
    if x > xmax then code := (TOP);
    else if x < xmin then code := (BOTTOM);
    if x > xmax then code := code + (RIGHT);
    else if x < xmin then code := code + (LEFT);
  end;
begin
  accept := false; done := false;
  CompOutCode (x0, y0, outside0); CompOutCode (x1, y1, outside1);
  repeat
    if (outside0 = [] and (outside1 = [])) then {Trivial accept and exit}
    begin accept := true; done := true; end
  until (outside0 + outside1 <> []);
  {Logical intersection is true, so trivial reject and exit.}
  done := true;
  {Failed both tests, so calculate the line segment to clip:
  from an outside point to an intersection with clip edge.}
  begin
    {At least one endpoint is outside the clip rectangle; pick it.}
    if (outside0 <> []) then
      outsideOut := outside0;
    else outsideOut := outside1;
    {Now find intersection point:
    use formulae x = y0 + slope * (x0 - y0), x = x0 + (1/slope) * (y - y0).}
    if TOP in outsideOut then
      begin
        {Divide line at top of clip rectangle}
        x := (x0 + (x1 - x0) * (ymax - y0)) / (y1 - y0);
        y := ymax;
      end
    else if BOTTOM in outsideOut then
      begin
        {Divide line at bottom of clip rectangle}
        x := x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
        y := ymin;
      end
    else if RIGHT in outsideOut then
      begin
        {Divide line at right edge of clip rectangle}
        y := y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
        x := xmax;
      end
    end
  end
  CompOutCode (x, y, outsideOut);
  if (outside0 + outsideOut = []) then
    {The line segment is now inside the clip rectangle.}
    accept := true; done := true;
  else
    {The line segment is still outside the clip rectangle.}
    outside0 := outsideOut;
  end
end;
  
```

Fig. 3.41 (Cont.)

9

```

else if LEFT in outsideOut then
  begin
    {Divide line at left edge of clip rectangle}
    y := y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
    x := xmin;
  end;
  {Now we move outside point to intersection point to clip,
  and get ready for next pass.}
  if (outsideOut = outside0) then
    begin
      x0 := x; y0 := y; CompOutCode (x0, y0, outside0);
    end
  else
    begin
      x1 := x; y1 := y; CompOutCode (x1, y1, outside1);
    end
  end;
  {Subdivide}
until done;
if accept then MidpointLineReal(x0, y0, x1, y1, value); {Version for real coordinates}
end; {CohenSutherlandLineClipAndDraw}
  
```

Fig. 3.41 Cohen-Sutherland line-clipping algorithm.

MCRivara/CG2004/2

10

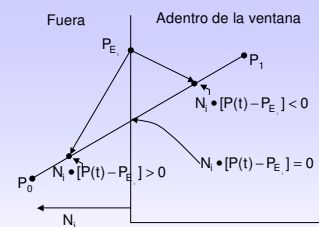
Algoritmo de Clipping de Líneas Paramétricas

- Algoritmo de Cohen Sutherland es muy usado pero no es el más eficiente ya que efectúa clippings "innecesarios".
- El algoritmo de Cyrus-Beck es más eficiente y puede usarse para clipar contra un rectángulo o cualquier polígono arbitrario convexo. Y se generaliza directamente a 3D.

MCRivara/CG2004/2

11

Idea del algoritmo de Cyrus-Beck



$$P(t) = P_0 + (P_1 - P_0)t \text{ con } 0 \leq t \leq 1$$

Se busca: solución en t para $N_i \cdot [P(t) - P_E] = 0$

MCRivara/CG2004/2

12

Algoritmo de Cyrus-Beck

- Puede aplicarse a cualquier polígono convexo
- Calcula intersecciones con todos los lados de la ventana (segmentos de borde del polígono convexo de clipping) pero de manera más eficiente y barata

MCRivara/CG2004/2

13

Ideas Algoritmo de Cyrus-Beck

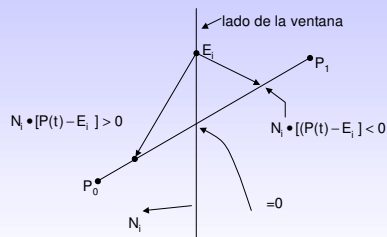
- Se usa ecuación paramétrica del segmento a clippear
- $P(t) = P_0 + (p_1 - P_0) t$
- E_i es punto cualquiera sobre lado de la ventana
- Se busca punto de intersección como solución a

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

N_i : normal a lado de ventana

MCRivara/CG2004/2

14



MCRivara/CG2004/2

15

Solución de $N_i \cdot [P(t) - E_i] = 0$

$$N_i \cdot [P_0 - (p_1 - P_0) t - E_i] = 0$$

y despejando t

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

Observaciones

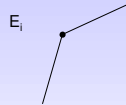
- Es necesario chequear $N_i \neq 0$; $D \neq 0$; $N_i \cdot D \neq 0$

$N_i \cdot D = 0 \Rightarrow N_i$ y D son paralelos y el algoritmo se mueve al caso siguiente

MCRivara/CG2004/2

16

- E_i se elige como punto extremo del polígono



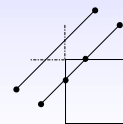
- Se usan los mismo E_i para todos los segmentos a clippear

MCRivara/CG2004/2

17

Algoritmo para Ventana Rectangular

- Para cada segmento se calculan los 4 valores t (para cada lado de la ventana)
- Se determinan los cortes internos a los segmentos de la ventana
- ¿Cómo?

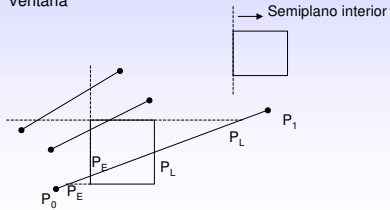


MCRivara/CG2004/2

18

- Las intersecciones se caracterizan como
 - Potencialmente entrantes PE
 - Potencialmente salientes PL

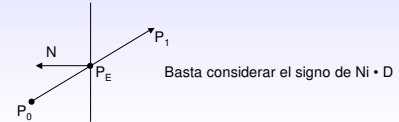
con respecto al semiplano que define el interior de la ventana



MCRivara/CG2004/2

19

- se considera ángulo entre $P_1 - P_0 = D$ y N
 - Si ángulo $< 90^\circ \Rightarrow P_L \mid N_i \cdot D > 0$
 - Si ángulo $> 90^\circ \Rightarrow P_E \mid N_i \cdot D < 0$



MCRivara/CG2004/2

20

- Solución es par de puntos (P_E, P_L) con valores de t

P_E con el mayor valor de t

P_L con el menor valor de t

Observación

$t_E > t_L \Rightarrow$ No hay porción de la línea dentro del rectángulo de clipping

MCRivara/CG2004/2

21

Tabla resumen

Lado ventana de clipping	N_i	E_i	$P_0 - E_i$	t
izquierdo	$(-1, 0)$	(x_{\min}, y)	$(x_0 - x_{\min}, y_0 - y)$	$\frac{x_0 - x_{\min}}{x_1 - x_0}$
derecha	$(1, 0)$	(x_{\max}, y)	$(x_0 - x_{\max}, y_0 - y)$	$\frac{x_0 - x_{\max}}{-x_1 - x_0}$
abajo	$(0, -1)$			
arriba	$(0, 1)$			

- Cálculos simples y eficientes
Los denominadores negativos son importantes para determinar si el segmento entra o sale en la ventana

MCRivara/CG2004/2

22