Computação Gráfica

Algoritmo de

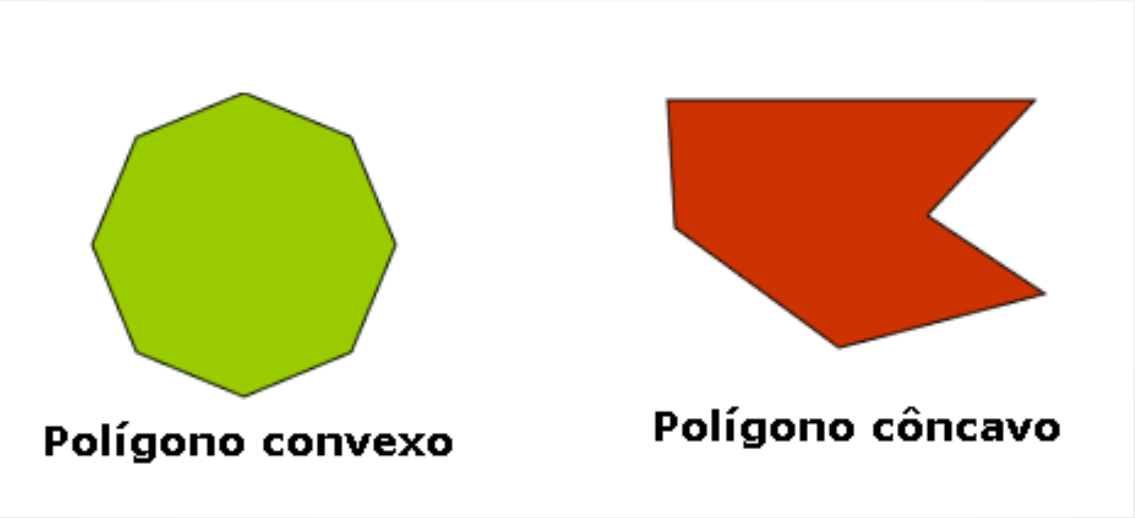
Cyrus-Beck

Giovanni Garcia
Gustavo Alves
Marcos Treviso
Paulo Almeida
Wolgan Quepfert

Roteiro

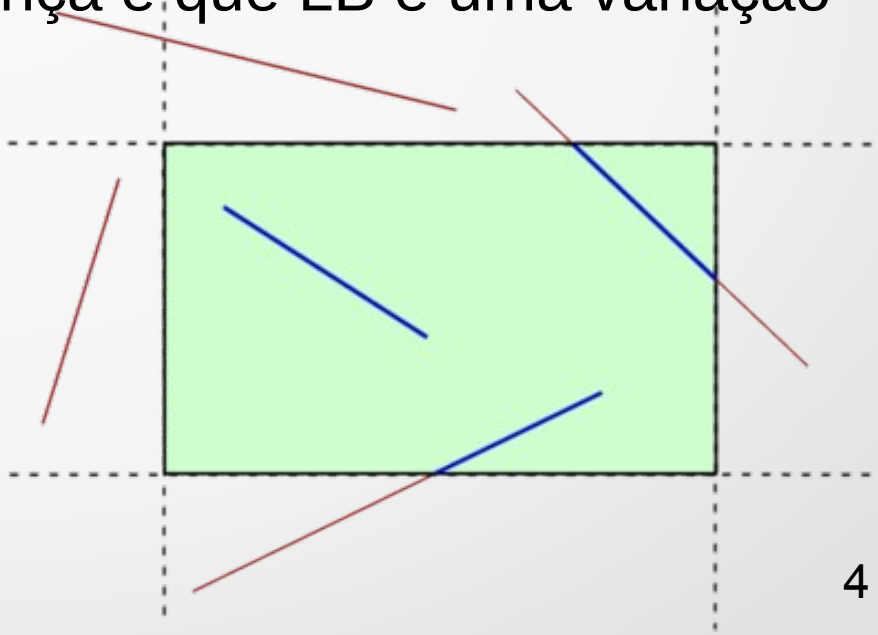
História

- Os cientistas Cyrus e Beck seguiram uma aproximação ao problema do recorte de segmentos de reta no qual conceberam um algoritmo.



Definição

- Foi primeiramente criado para o recorte de uma linha usando uma forma paramétrica contra um polígno convexo de duas dimensões.
- Similiar ao Liang-Barsky, diferença é que LB é uma variação simplificada do CB.
- Possui complexidade de O(n).

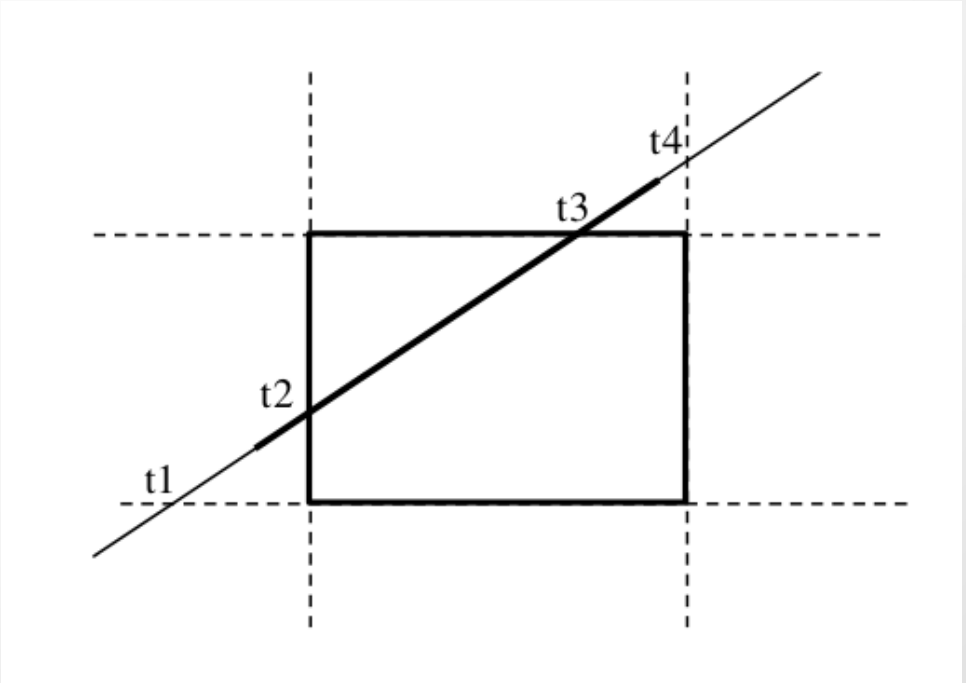


Intersecção entre linhas

- Através da classificação de cada um dos pontos de intersecção é possível saber quais os valores de t dos vértices do segmento recortado, caso existam.

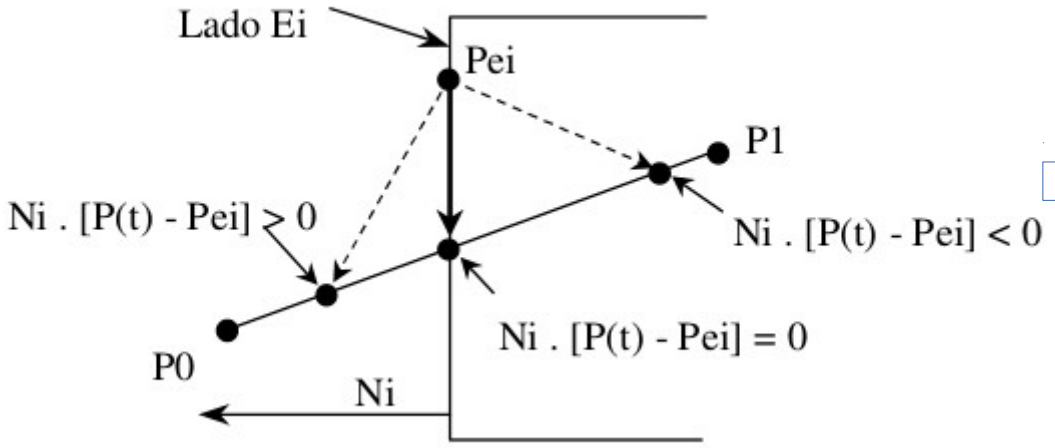
Equação paramétrica da reta:

$$P(t) = P_0 + (P_1 - P_0)t; \quad t \in (0, 1)$$



Intersecção entre linhas

- **Recorte paramétrico:** pontos de interseção da reta contendo o segmento de reta a recortar com o retângulo de recorte.



REGRAS

$$t = \frac{N_i \bullet (P_0 - P_{ei})}{-N_i \bullet D}$$

Intersecção vetor normal Pei e Ponto da Reta: Ni • [P(t) − Pei] = 0

Intersecção entre linhas

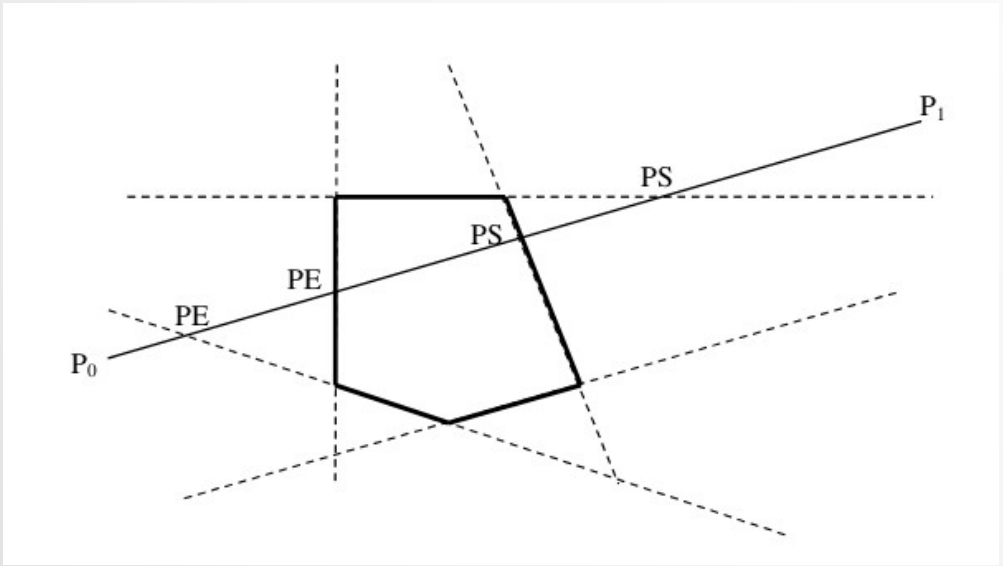
- O algoritmo paramétrico é eficiente quando é necessário recortar muitos segmentos de reta.

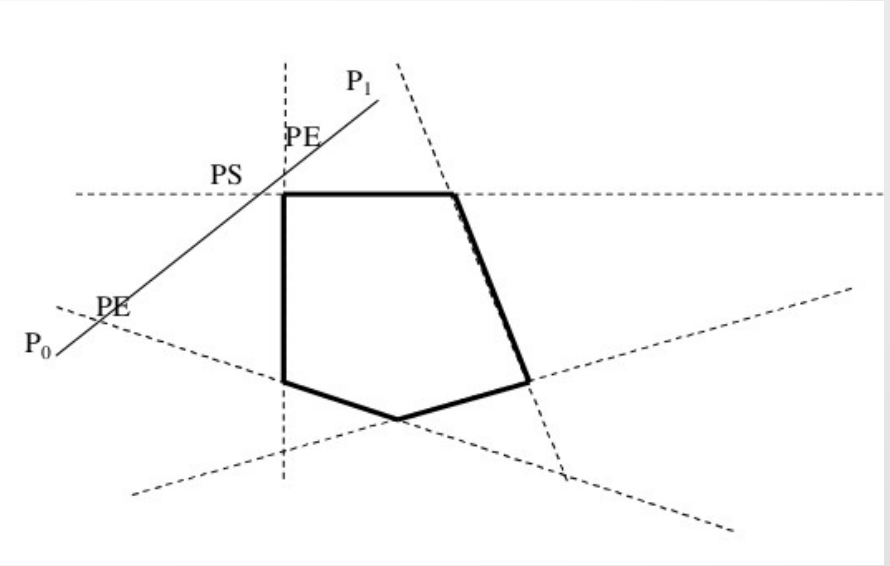
$$t = \frac{N_i \bullet (P_0 - P_{ei})}{-N_i \bullet D}$$

Lado:	Ni	Pei
Left ($X=X_{min}$)	[-1, 0]	(X_{min}, Y)
Right ($X=X_{max}$)	[1, 0]	(X_{max}, Y)
Bottom ($Y=Y_{min}$)	[0, -1]	(X, Y_{min})
Top ($Y=Y_{max}$)	[0, 1]	(X, Y_{max})

Descrição do Algoritmo

- **Algoritmo de Cyrius-Beck:** segmento de reta a recortar com identificação do tipo (PE ou PS) das suas intersecções com o polígono de recorte.





Pseudocódigo

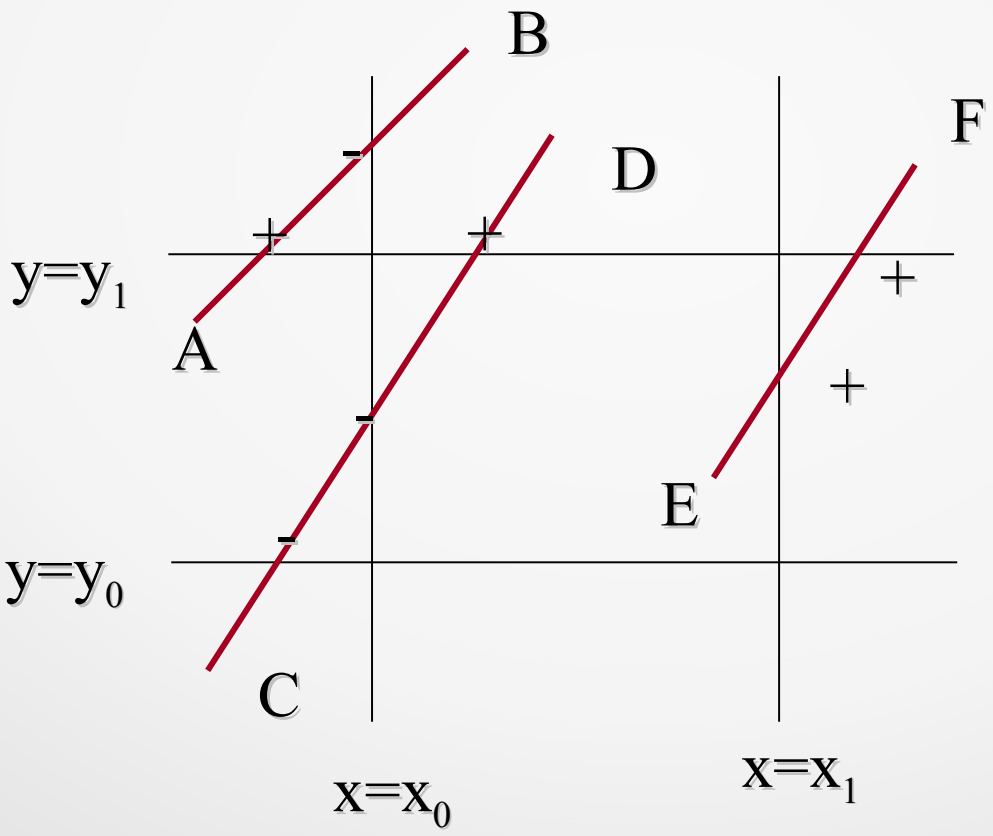
```
Calcule Ni e escolha um PEi para cada aresta

tE = 0;
tL = 1;
for(cada aresta){
  if (Ni.(P1-P0)!=0 ){ /* aresta não é paralela ao segmento */
    calcule t;
    use sign of Ni.(P1-P0) para categorizar como PE ou PL;
    if( PE ) tE = max(tE, t);
    if( PL ) tL = min(tL, t);
  }
  else{   /* aresta paralela ao segmento */
    if (Ni.(P0-PEi) > 0) /* está fora */
    return nil;
  }
  if (tE > tL)
    return nil;
  else
    return P(tE) and P(tL) as true clip intersections;
}
```

Código em C

```
#include <windows.h>   //caso precise
#include <gl/Gl.h>
#include <gl/glut.h>
#include <iostream>


//********* Estrutura de Dados


struct GLintPoint{
        GLint x,y;
};


struct GlfloatPoint{
        GLfloat x,y;
};
```

Demonstração

- Usando OpenGL.



Nossa Opinião

- É necessário entendimento de Geometria.
- Bom desempenho.
- Baixa complexidade.
- Várias referências na internet e em livros.

Referências

- Cyrus, M., Beck, J.: Generalized Two and Three Dimensional Clipping, Computers&Graphics, Vol.3, No.1, pp.23-28, 1978

- CS 535 Test Programs, http://cs1.bradley.edu/public/jcm/cs535OpenGLCyrusBeck.html

- TI Especialistas - Geometria Computacional, http://www.tiespecialistas.com.br/2013/08/geometria-computacional-algoritmo-de-cyrus-beck

Fim

Obrigado à todos pela atenção!

Giovanni Garcia

Gustavo Alves

Marcos Treviso

Paulo Almeida

Wolgan Quepfert