REO2 – Computação Gráfica (GCC124)

Pedro Antônio de Souza – 201810557

EXERCÍCIO 17

Inicialmente, algoritmo de Cohen-Sutherland divide o plano do retângulo de recorte em nove regiões, cada uma possuindo um código binário único de 4 bits. Cada bit do código é utilizado para determinar uma posição de um ponto em relação ao retângulo de recorte, como descrito abaixo:

- O **primeiro bit** será 1 caso o ponto analisado esteja **acima** do retângulo de recorte;
- O **segundo bit** será 1 caso o ponto analisado esteja **abaixo** do retângulo de recorte;
- O **terceiro bit** será 1 caso o ponto analisado esteja à **direta** do retângulo de recorte;
- O **quarto bit** será 1 caso o ponto analisado esteja à **esquerda** do retângulo de recorte.

Um segmento de reta não precisará ser processado caso suas extremidades satisfaçam alguma das condições abaixo:

1. Se o código de ambas extremidades forem 0, o segmento está totalmente dentro da região de recorte.
2. Se o resultado de uma operação "e" lógico bit a bit (operador `&` em C++) entre os códigos das extremidades for diferente de zero, o segmento está totalmente fora da região de recorte.

Em um programa de computador, o algoritmo deve receber as extremidades como as coordenadas de dois pontos, sendo P_0 e P_1. Esses pontos devem ser processados por uma função auxiliar que retornará seus códigos binários. Assim, o programa pode verificar se os dois pontos possuem código 0 ou se a operação "e" lógico bit a bit é diferente de zero.

EXERCÍCIO 18

O aperfeiçoamento de Liang e Barsky tornou o algortimo de Cyrus-Beck mais eficiente nos casos em que a área de recorte é um retângulo alinhado com os eixos do sistema de coordenadas.

EXERCÍCIO 20

A modificação apresentada por Liang e Barsky obteve sucesso pois a normal de qualquer lado *i* da área de recorte **sempre** terá uma coordenada igual a zero. Dessa forma, o cálculo do produto escalar entre a normal N_i e o vetor que vai do ponto sobre o lado *i* até o ponto inicial do seguimento, também poderá desconsiderar a coordenada nula. É importante notar que a coordenada a ser desconsiderada é precisamente aquela que precisaria ser definida aleatoriamente.

Para demonstrar essa simplificação, deduziremos o cálculo de interseção de um seguimento de reta *D* que vai dos pontos $P_1(x_1, y_1)$ ao ponto $P_2(x_2, y_2)$ com um retângulo de recorte definido pelos limites x_{min}, x_{max}, y_{min}, y_{max}, utilizando a fórmula $t=\frac{N_i\cdot[P_1-P_{Li}]}{-N_i\cdot D}$.

Para calcular o parâmetro t_L da interseção para o **limite esquerdo** do retângulo de recorte, temos que $N_L = (-1, 0)$ e $P_L(x_{min}, y_{min})$. Substituindo na fórmula:

$$t_L=\frac{(-1,0)\cdot[(x_1,y_1)-(x_{min},y_{min})]}{-(-1,0)\cdot[(x_2,y_2)-(x_1,y_1)]} \quad (1)$$

$$t_L=\frac{(-1,0)\cdot[(x_1-x_{min},y_1-y_{min})]}{-(-1,0)\cdot(x_2-x_1,y_2-y_1)} \quad (2)$$

$$t_L=\frac{(-1)\times(x_1-x_{min})+0\times(y_1-y_{min})}{-[-1\times(x_2-x_1)+0\times(y_2-y_1)]} \quad (3)$$

Perceba que nesse passo, fica demonstrado que a coordenada *y* poderia ter sido desconsiderada desde o princípio da aplicação da fórmula. Continuando:

$$t_L=\frac{(-1)\times(x_1-x_{min})}{-[-1\times(x_2-x_1)]} \quad (4)$$

$$t_L=\frac{x_{min}-x_1}{x_2-x_1} \quad (5)$$

Nos próximos cálculos, assumiremos que a coordenada nula da normal do lado que iremos analisar, deverá ser desconsiderada dos cálculos já que foi demonstrada essa propriedade no cálculo anterior.

Para calcular o parâmetro t_R da interseção para o **limite direito** do retângulo de recorte, temos que $N_R = (1, 0)$, portanto:

$$t_R=\frac{1\times(x_1-x_{max})}{-[1\times(x_2-x_1)]}$$

$$t_R=\frac{x_1-x_{max}}{x_1-x_2}$$

Para calcular o parâmetro t_T da interseção para o **limite superior** do retângulo de recorte, temos que $N_T = (0, 1)$, portanto:

$$t_T=\frac{1\times(y_1-y_{max})}{-[1\times(y_2-y_1)]}$$

$$t_T=\frac{y_1-y_{max}}{y_1-y_2}$$

Por fim, para calcular o parâmetro t_B da interseção para o **limite inferior** do retângulo de recorte, temos que $N_B = (0, -1)$, portanto:

$$t_B = \frac{(-1)\times(y_1 - y_{min})}{-[-1\times(y_2 - y_1)]}$$

$$t_B = \frac{y_{min} - y_1}{y_2 - y_1}$$

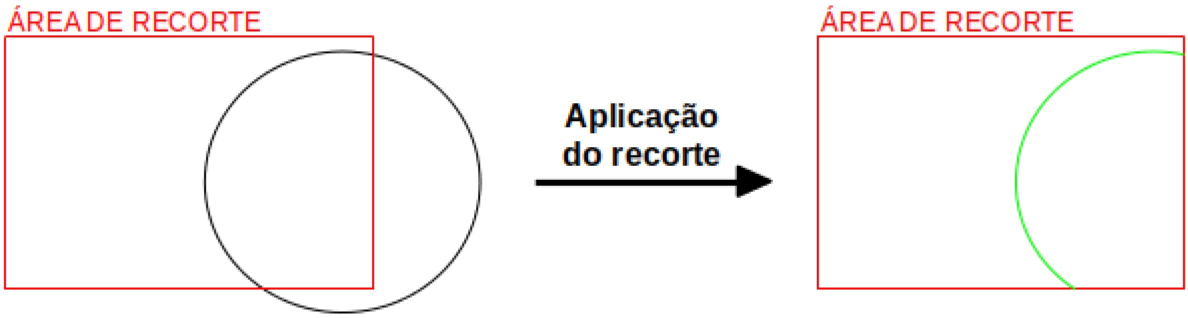
EXERCÍCIO 21

Primeiramente, o algoritmo irá inicializar a posição de início com *0* e a posição de fim com *1*. Para analisar interseções com cada lado, calcula-se um novo parâmetro t_{SIDE}.

- Ao verificar a interseção com o **lado esquerdo**, o segmento está possivelmente entrando e está posterior a posição de início *0*, portanto a posição de início é alterada para t_L.
- Ao verificar a interseção com o **lado inferior**, o segmento está possivelmente entrando e está posterior a posição de início t_L, portanto a posição de início é alterada para t_B.
- Ao verificar a interseção com o **lado direito**, o segmento está possivelmente saindo e está posterior a posição de fim *1*, portanto nada é alterado.
- Ao calcular a interseção com o **lado superior**, o segmento está possivelmente saindo e está posterior a posição de fim *1*, portanto nada é alterado.

Por fim, o algoritmo retorna o segmento de reta dado pelas coordenadas da nova posição de início definida pelo parâmetro t_B e o ponto P_1, já que a posição de fim não foi alterada.

EXERCÍCIO 22



Independentemente da posição do segmento de elipse em relação a área de recorte, será sempre retornada um único novo segmento. Isso se deve ao fato de que uma elipse nunca possui reentrâncias.