Rasterização

1 Introdução

No andar final do pipeline de visualização, as primitivas gráficas são enviadas ao dispositivo físico onde são afixadas, depois de realizada a transformação das suas coordenadas para as coordenadas próprias do dispositivo. A afixação das primitivas em unidades do tipo vectorial não apresenta quaisquer problemas, pelo menos no caso de primitivas simples suportadas por essas unidades.

Porém, no caso de dispositivos gráficos do tipo de quadrícula, há ainda que converter tais primitivas nas quadrículas dos dispositivos, em operações denominadas de rasterização ou, como também são conhecidas, conversão por varrimento. O objectivo desta operação é determinar quais as quadrículas que representarão as primitivas gráficas.

A rasterização de primitivas gráficas é uma operação que é executada milhões de vezes e, portanto, faz todo o sentido que os algoritmos desenvolvidos, além de específicos, devam ser eficientes para que o desenho das primitivas seja o mais rápido possível.

Cada tipo de primitivas é diferente dos restantes e, consequentemente, os algoritmos de rasterização são também diferentes. Neste capítulo apresentaremos os algoritmos de rasterização para o traçado de segmentos de recta e de circunferências e para o preenchimento de polígonos que são três das primitivas mais frequentes.

Na rasterização, algo que é contínuo como, por exemplo, um segmento de recta, é convertido numa colecção de quadrículas que tem uma natureza discreta. Esta conversão faz perder parte da informação existente. Isto tem como consequência que não é possível representar um segmento de recta com a mesma densidade visual se o segmento for horizontal (ou vertical) ou apresentar um declive de 45º. No entanto, as quadrículas que compõem estes segmentos estarão sempre alinhadas segundo a recta que as suporta (veja-se a figura 1-1).

Porém, tal não sucede a, por exemplo, um segmento de recta com declive de 1/3, tal como a figura 1-2 apresenta, surgindo uma colecção de quadrículas dispostas em vários patamares, dando a impressão de uma escada. A este fenómeno visual devido à rasterização dá-se o nome de *aliasing*. No final deste capítulo abordaremos este problema, apresentando algoritmos de o combater, os chamados algoritmos de *antialiasing*[1].

[1] Porque os algoritmos de *antialiasing* podem apresentar soluções dependentes das técnicas de geração de imagem, abordaremos este tema sempre que tal se justificar como, por exemplo, no âmbito da técnica de Ray Tracing.

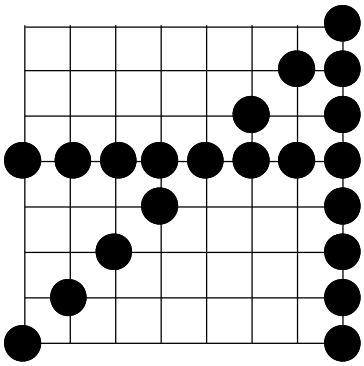

Figura 1-1 – Rasterização de linhas horizontais, verticais e diagonais, de densidade visual variável.

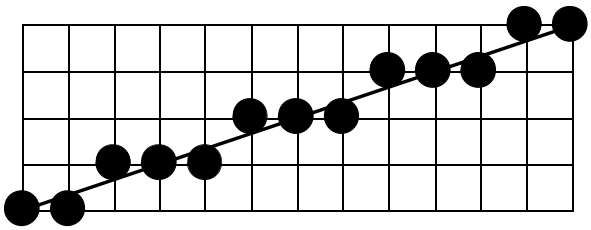

Figura 1-2 – Quadrículas de um segmento de recta rasterizado apresentando o efeito de *aliasing*.

Finalmente, convém deixar aqui uma nota de natureza histórica referente às capacidades dos dispositivos de quadrícula. Quando estes dispositivos foram pela primeira vez introduzidos, não possuíam a capacidade de realizar a rasterização e esta tinha que ser realizada por software pelos controladores dos dispositivos (*device drivers*). Aos poucos, estas capacidades foram sendo implementadas em hardware, com o consequente aumento do desempenho gráfico. Hoje em dia, praticamente todas as operações de rasterização são realizadas em hardware cujo desempenho é extremamente elevado.

2 Rasterização de Segmentos de Recta

A rasterização de segmentos de recta consiste em, num dispositivo de quadrícula, dados os dois píxeis extremos de um segmento de recta, determinar que píxeis localizados entre eles devem ser seleccionados para compor visualmente o segmento.

Este problema tem solução trivial quando o segmento a traçar é horizontal, vertical ou diagonal (veja-se a figura 1-1), se bem que, como já referimos e a figura mostra, a densidade visual de segmentos diagonais não seja a mesma dos que são verticais ou horizontais. Fora destes casos, a determinação dos píxeis que irão constituir um dado segmento de recta coloca alguns problemas que a figura 2-1 ilustra.

Se apenas fossem seleccionados os píxeis pertencentes ao segmento de recta, no caso do segmento PV da figura só seriam seleccionados os píxeis P, R, T e V, daqui resultando uma linha de fraca densidade visual. Este problema acentuar-se-ia no segmento AK, fazendo com que só fossem seleccionados os píxeis A, F e K, e, no limite, o segmento LO ficaria reduzido aos seus extremos. Por esta razão, esta regra não pode constituir uma solução para o problema proposto.

Para que a densidade visual do segmento seja o mais uniforme possível, há então que seleccionar o maior número possível de píxeis entre os dois píxeis dos extremos de um segmento de recta, seleccionando os píxeis mais próximos do segmento. Para o caso do segmento AK, isto corresponderia a ter como possíveis candidatos os píxeis B, C, D, E, G, H, I e J. A selecção de todos estes píxeis produziria no entanto uma linha cuja densidade variaria ao longo do segmento.

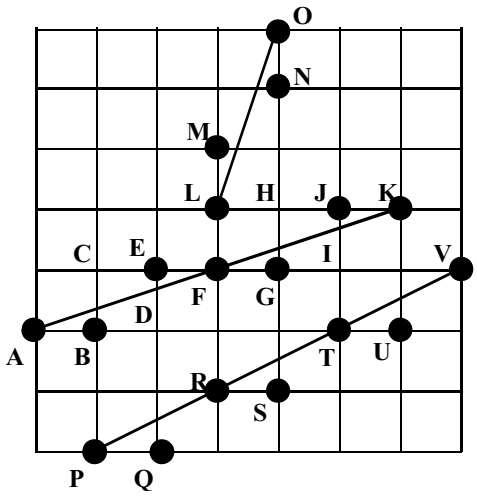

Figura 2-1 – Rasterização de segmentos de recta mostrando píxeis seleccionáveis e píxeis seleccionados.

A solução para esta questão consiste em, para dois píxeis candidatos situados sobre uma mesma linha vertical de píxeis, escolher o pixel mais próximo do segmento de recta. Desta forma, os píxeis B, E, G e I seriam os escolhidos, tal como a figura 2-1 apresenta.

Consideremos agora o caso do segmento PV. Excluindo os píxeis pelos quais o segmento passa (R e T), em todos os outros casos o segmento passa exactamente pelo ponto médio entre dois píxeis candidatos. É normal convencionar que a escolha recaia então sobre o pixel localizado imediatamente abaixo do segmento, pelo que os píxeis a serem seleccionados serão os píxeis O, S e U.

Para tratar o segmento LO, em vez de escolher dois píxeis candidatos localizados sobre uma linha vertical de píxeis, teremos que escolher agora entre dois píxeis que se localizem sobre uma linha horizontal de píxeis. Isto equivale a realizar uma transformação de coordenadas trocando os dois eixos, seleccionar os píxeis que farão parte da representação do segmento de recta e voltar a trocar os eixos, numa transformação inversa da transformação precedente.

Esta solução para o caso do segmento LO, que apresenta um declive de módulo superior a 1, permite tratar todos os casos de uma única forma, isto é, fazendo a selecção de píxeis somente sobre linhas verticais de píxeis. Assim, para estes casos, a selecção de píxeis será realizada incrementando a coordenada horizontal de uma em uma unidade, mas será precedida de uma transformação que procederá à troca de coordenadas (eixos) e seguida por uma nova transformação de troca de eixos. Este tema será retomado mais adiante.

Esta regra será aplicada na apresentação dos algoritmos para selecção dos píxeis fazendo parte da representação de segmentos de recta em dispositivos de quadrícula e em que apresentaremos os seguintes três algoritmos:

- Algoritmo imediato
- Algoritmo incremental básico
- Algoritmo de Bresenham

Estes algoritmos diferem na precisão dos resultados e na velocidade de execução, ou seja, na carga computacional envolvida. Todos eles se destinam a seleccionar quais os pontos que compõem a representação de um segmento de recta num dispositivo de

quadrícula quando os seus extremos se localizam nos píxeis P_1 e P_2 de coordenadas (x_1, y_1) e (x_2, y_2), respectivamente.

2.1 Algoritmo Imediato

O algoritmo imediato para a rasterização de segmentos de recta parte da equação da recta que suporta o segmento, cuja expressão é

$$y = m\,x + b \tag{2-1}$$

Para um segmento de recta cujos extremos cujas coordenadas são (x_1, y_1) e (x_2, y_2), é imediato que os coeficientes m e b da equação (2-1) deverão ser

$$\begin{aligned} m &= \frac{y_2 - y_1}{x_2 - x_1} \\ b &= y_1 - m\,x_1 \end{aligned} \tag{2-2}$$

Fazendo então variar x entre x_1 e x_2 por incrementos de uma unidade, o valor de y calculado será um valor real que deve ser arredondado para o inteiro mais próximo, o que pode ser realizado através de uma das seguintes expressões equivalentes

$$y = Round(m\,x + b) \quad y = Floor(0{,}5 + m\,x + b) \tag{2-3}$$

em que *Round* é a função de arredondamento matemático e *Floor* a função de truncatura.

O custo computacional do cálculo do valor de y para cada valor de x corresponde à realização de três operações de vírgula flutuante (uma multiplicação e duas adições) e à operação de truncatura de um valor real para valor inteiro.

2.2 Algoritmo Incremental Básico

Poderemos melhorar o desempenho da rasterização de segmentos de recta se diminuirmos o número de operações matemáticas envolvidas. Para tal consideremos o valor de y para dois valores consecutivos de x que diferem entre si de uma unidade, ou seja,

$$x_{i+1} = x_i + 1 \tag{2-4}$$

O valor de y para x_{i+1}, tendo em conta (2-4), é

$$\begin{aligned} y_{i+1} &= m\,x_{i+1} + b \\ &= m\,(x_i + 1) + b \\ &= m + m\,x_i + b \\ &= m + y_i \end{aligned} \tag{2-5}$$

Isto significa que, para calcular o valor de *y* para um novo pixel cujo valor da coordenada *x* dista uma unidade do valor anterior, basta adicionar o declive da recta (*m*) ao anterior valor de *y*.

Novamente, é ainda necessário proceder ao arredondamento matemático do valor calculado.

Em relação ao algoritmo imediato, este algoritmo dispensa a realização da multiplicação efectuada por aquele e, portanto o número de operações de vírgula flutuante a realizar reduz-se agora a duas adições e uma truncatura. Como a multiplicação é uma operação mais cara que uma adição em termos computacionais, é de esperar um aumento significativo do desempenho da rasterização de segmentos de recta quando o algoritmo incremental básico é empregue.

Este algoritmo apresenta porém um problema de acumulação de erros devido ao número limitado de algarismos significativos com que um valor real pode ser representado em memória, pois o valor representado pode não ser exactamente o valor do declive. Assim, a realização de sucessivas adições irá aumentar o erro com que o valor de *y* é calculado, o que não é de forma alguma desejável.

2.3 Algoritmo de Bresenham

Os algoritmos anteriores apresentam vários inconvenientes, tanto quanto a precisão, como no referente ao tipo de aritmética empregue, o que é muito importante para o desempenho da rasterização. Com efeito, uma operação entre operandos inteiros é mais rápida do que a correspondente operação entre operandos em vírgula flutuante. O algoritmo de Bresenham realiza a rasterização de segmentos de recta empregando apenas operações de aritmética de inteiros e, portanto, permite um maior desempenho. O algoritmo baseia-se no critério do ponto médio.

Consideremos a figura 2-2 onde está representada uma linha recta que intersecta duas colunas de píxeis. Para cada coluna de píxeis existem dois píxeis que se encontram mais próximos da recta, um abaixo e outro acima desta. A escolha do pixel a seleccionar poderia ser realizada determinando a distância da intersecção da recta com a coluna de píxeis a cada um dos dois píxeis, escolhendo-se então o pixel mais próximo da intersecção. No entanto, esta determinação acarretaria uma carga computacional elevada.

Porém, se atentarmos na localização do ponto médio entre os dois píxeis em relação à recta, teremos uma forma simples de determinar qual dos píxeis deve ser seleccionado. Como a figura 2-2 mostra, se o ponto médio se encontrar abaixo da recta (M_2), deve ser seleccionado o pixel imediatamente acima desta. Caso contrário (ponto M_1), o pixel a seleccionar deverá ser o pixel imediatamente abaixo desta.

Para determinar a posição do ponto médio em relação à recta, consideremos a forma implícita da equação de uma recta, que é

$$F(x,y) = ax + by + c \tag{2-6}$$

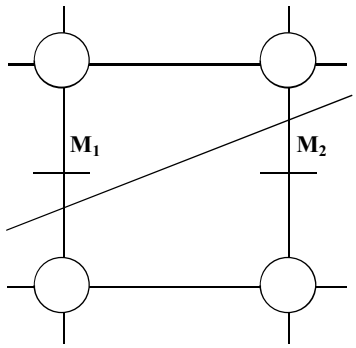

Figura 2-2 – Posicionamento do ponto médio em relação à linha a traçar na selecção de píxeis.

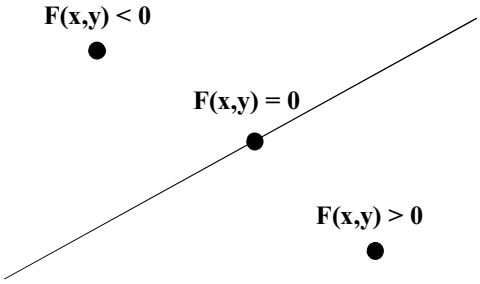

Figura 2-3 – Sinal da função implícita da recta em função da posição de um ponto relativamente à recta.

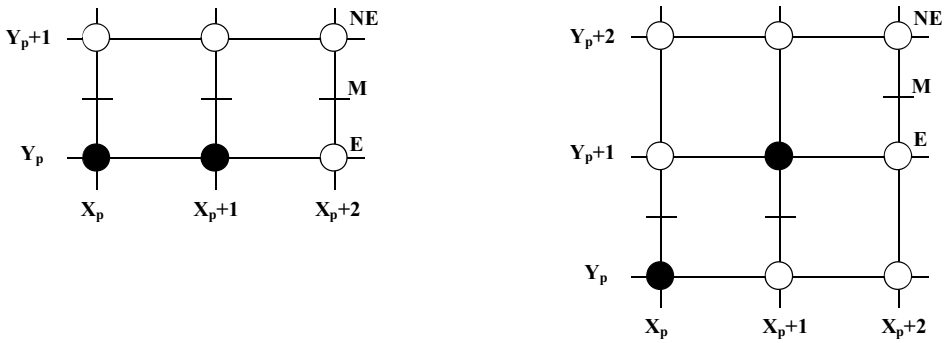

Figura 2-4 – Localização do ponto médio consoante o pixel anteriormente seleccionado (E à esquerda e NE à direita).

Para um dado ponto de um plano, esta função tem valor nulo se o ponto pertencer à recta. O valor da função será negativo se o ponto se encontrar acima da recta e positivo se o ponto se encontrar abaixo desta (veja-se a figura 2-3).

Isto significa que, se a expressão (2-6) assumir um valor positivo no ponto médio, este ponto encontra-se abaixo da recta e deve ser seleccionado o pixel imediatamente acima desta. Caso contrário, o pixel a seleccionar será o pixel imediatamente abaixo. Para o caso especial da função assumir o valor nulo poder-se-ia escolher qualquer um dos dois píxeis. Neste caso, é prática comum escolher o pixel imediatamente abaixo, tal como já foi referido.

Consideremos agora a figura 2-4 onde se apresentam dois casos distintos. Em ambos os casos encontram-se já seleccionados dois píxeis para os quais a coordenada *x* assume os valores x_p e x_p+1 e pretende-se determinar qual o próximo pixel a seleccionar quando a coordenada *x* assumir o valor x_p+2 .

Para o primeiro caso (à esquerda na figura), a selecção realiza-se entre os píxeis E e NE[2]. O valor da função implícita da recta para o ponto médio entre estes dois píxeis é

$$\begin{aligned} F\left(x_p+2, y_p+1/2\right) &= a\left(x_p+2\right)+b\left(y_p+1/2\right)+c \\ &= a\left(x_p+1\right)+a+b\left(y_p+1/2\right)+c \\ &= a\left(x_p+1\right)+b\left(y_p+1/2\right)+c+a \\ &= F\left(x_p+1, y_p+1/2\right)+a \end{aligned} \tag{2-7}$$

ou seja, o valor da função implícita no ponto médio é igual ao valor do coeficiente *a* adicionado ao valor da função implícita calculado quando se pretendeu determinar qual o pixel a seleccionar para $x=x_p+1$.

Para o caso à direita na figura 2-4, o ponto médio encontra-se em $(x_p+2,\ y_p+3/2)$. A função implícita da recta no ponto médio assume agora o valor

$$\begin{aligned} F\left(x_p+2, y_p+3/2\right) &= a\left(x_p+2\right)+b\left(y_p+3/2\right)+c \\ &= a\left(x_p+1\right)+a+b\left(y_p+1/2\right)+b+c \\ &= a\left(x_p+1\right)+b\left(y_p+1/2\right)+c+a+b \\ &= F\left(x_p+1, y_p+1/2\right)+a+b \end{aligned} \tag{2-8}$$

Neste caso, é necessário adicionar os valores dos coeficientes a e b da equação implícita da recta ao valor que esta equação assumiu no ponto médio quando se determinou o pixel a seleccionar para $x=x_p+1$.

Dispomos assim de um algoritmo recursivo. Resta agora determinar os valores dos coeficientes a e b e o valor inicial da função para que o algoritmo esteja completo.

O segmento de recta tem como extremos os píxeis de coordenadas (x_1, y_1) e (x_2, y_2) onde o valor da função implícita é nulo. Definindo agora

$$\begin{aligned} \Delta x &= x_2 - x_1 \\ \Delta y &= y_2 - y_1 \end{aligned} \tag{2-9}$$

a forma explícita da equação da recta em função destas diferenças será

$$y = \frac{\Delta y}{\Delta x} x + B \tag{2-10}$$

Multiplicando então ambos os membros por Δx e agrupando os termos, obteremos a forma implícita da equação da recta que suporta o segmento em função das diferenças Δx e Δy

$$F\left(x, y\right) = \Delta y\, x - \Delta x\, y + \Delta x\, B \tag{2-11}$$

[2] Por analogia com os pontos cardeais.

Comparando as expressões (2-6) e (2-11), é imediato que os coeficientes a, b e c da forma implícita da equação da recta que suporta o segmento serão

$$\begin{aligned} a &= \Delta y \\ b &= -\Delta x \\ c &= \Delta x\, B \end{aligned} \tag{2-12}$$

No pixel inicial do segmento, de coordenadas (x_1, y_1), a função implícita tem o valor nulo pois o pixel encontra-se sobre a recta. Para o pixel seguinte, o ponto médio tem como coordenadas $(x_1+1, y_1+1/2)$, e há que determinar o valor da função implícita nesse ponto, que é

$$\begin{aligned} F(x_1+1, y_1+1/2) &= a(x_1+1) + b(y_1+1/2) + c \\ &= (a\,x_1 + b\,y_1 + c) + a + b/2 \\ &= F\,(x_1, y_1) + a + b/2 \\ &= \Delta y - \Delta x/2 \end{aligned} \tag{2-13}$$

Como os valores de Δx e Δy são inteiros, todo o algoritmo poderia ser implementado em aritmética de inteiros se Δx fosse sempre par, o que não pode ser assegurado.

Porém, dado que a função implícita da recta é uma função homogénea, poderemos reescrevê-la da seguinte forma

$$F(x,y) = 2ax + 2by + 2c \tag{2-14}$$

e, assim, o seu valor no ponto médio para seleccionar o segundo pixel passará a ser

$$F(x_1+1, y_1+1/2) = 2\,\Delta y - \Delta x \tag{2-15}$$

que será sempre inteiro e as expressões de recursividade serão agora

$$\begin{aligned} F(x_p+2, y_p+1/2) &= F\,(x_p+1, y_p+1/2) + 2\,\Delta y \\ F(x_p+2, y_p+3/2) &= F\,(x_p+1, y_p+1/2) + 2\,(\Delta y - \Delta x) \end{aligned} \tag{2-16}$$

para avanços nos píxeis anteriores para E e NE, respectivamente, possibilitando assim a implementação do algoritmo em aritmética de inteiros.

O algoritmo de Bresenham pode então ser apresentado como se segue

1. Calcular
   $dx = x_2 - x_1$
   $dy = y_2 - y_1$
   $d = 2dy - dx$
   $incE = 2dy$
   $incNE = 2(dy - dx)$
2. Inicializar $x = x_1$ e $y = y_1$ e marcar o pixel com estas coordenadas.
3. Repetir os passos seguintes enquanto $x < x_2$
4. Se $d \leq 0$, incrementar d de $incE$.
   Caso contrário, incrementar d de $incNE$ e incrementar y de uma unidade.
5. Incrementar x de uma unidade e marcar o pixel com as coordenadas x e y.

2.3.1 Exemplo

Aplicando os algoritmos incremental e de Bresenham ao segmento de recta unindo os píxeis (5, 8) e (9, 11), obtemos os píxeis que a tabela 2-1 apresenta. Deve notar-se que, para x=7, os dois algoritmos diferem nos resultados obtidos pois o arredondamento matemático do algoritmo incremental arredonda o valor 9,5 para 10, seleccionando o pixel (7, 10), enquanto o algoritmo de Bresenham selecciona o pixel (7, 9)[3].

x_i	Incremental		Bresenham			
	m=(11-8)/(9-5)=0,75		*dx=9-5=4* *dy=11-8=3* *d=2×3-4=2* *incE=2×3=6* *incNE=2×(3-4)=-2*			
	$y_i=y_{i-1}+m$	**round(y_i)**	**d_i**	**avanço**	**d_{i+1}**	**y_i**
5	-	***8***	0	**-**	2	***8***
6	8,75	***9***	2	NE	0	***9***
7	9,5	***10***	0	E	6	***9***
8	10,25	***10***	6	NE	4	***10***
9	11,0	***11***	4	NE	-	***11***

Tabela 2-1 – Comparação entre o algoritmo incremental e o algoritmo de Bresenham no traçado do segmento de recta entre os píxeis (5, 8) e (9, 11).

[3] Se o incremento 0,5 não fosse representado exactamente, o que não é o caso, haveria coincidência entre todos os pixéis seleccionados pelos dois algoritmos.

2.4 Redução ao Primeiro Octante

O algoritmo de Bresenham, tal como foi apresentado, pressupõe que o declive dos segmentos de recta a rasterizar esteja compreendido no intervalo [0; 1] e que o algoritmo é aplicado por valores crescentes da coordenada x distando de uma unidade. Isto significa que o algoritmo de Bresenham é apenas aplicável a segmentos de recta existentes no primeiro octante.

Para segmentos de recta que não existam no primeiro octante, há que transformá-los para o primeiro octante antes de aplicar o algoritmo de Bresenham e que aplicar a transformação inversa aos píxeis que o algoritmo calcule.

A transformação a realizar antes de aplicar o algoritmo consiste nos seguintes passos que devem ser efectuados pela seguinte ordem:

- Transformar segmentos de recta de declive negativo em segmentos de recta de declive positivo, substituindo os valores da coordenada y de cada extremo do segmento pelos respectivos valores simétricos.
- Transformar segmentos de recta de declive superior a 1 em segmentos de recta com declive inferior a 1, trocando as coordenadas x e y em cada um dos extremos do segmento.
- Trocar a ordem dos extremos do segmento de recta se o valor da coordenada x do primeiro extremo for superior ao valor da mesma coordenada do segundo extremo.

A transformação inversa, a ser aplicada a cada um dos píxeis calculados pelo algoritmo, deve inverter a transformação segundo uma ordem dos passos inversa da anterior, isto é:

- Se o declive do segmento de recta for superior a 1, trocar as coordenadas x e y de cada pixel calculado.
- Se o declive do segmento de recta for negativo, substituir a valor da coordenada y do pixel calculado pelo seu valor simétrico.

Note-se que, como é indiferente traçar um segmento de recta do primeiro para o segundo extremo ou do segundo para o primeiro, uma eventual troca da ordem dos extremos não necessita de ser invertida na transformação inversa.

O algoritmo para a transformação de um segmento de recta para o primeiro octante antes de aplicar o algoritmo de Bresenham pode agora ser apresentado como segue

1. Definir dois valores lógicos, *declive* e *simetrico*, e inicializá-los como falsos.
2. Calcular os valores de dx e dy conforme definido pelo algoritmo de Bresenham
3. Testar o produto $dx \times dy$ e, se for negativo, substituir o valor de y em cada um dos extremos e o valor de dy pelos respectivos valores simétricos, e atribuir a *simetrico* o valor verdadeiro
4. Testar se o valor absoluto de dx é menor que o valor absoluto de dy e, caso afirmativo, trocar os valores das coordenadas x e y de cada extremo, trocar os valores de dx e dy, e atribuir a *declive* o valor verdadeiro
5. Verificar se o valor da coordenada x do primeiro extremo é superior ao valor da mesma coordenada do segundo extremo e, em caso afirmativo, trocar os dois extremos e substituir os valores de dx e dy pelos respectivos valores simétricos.

Para cada pixel calculado pelo algoritmo de Bresenham, a transformação inversa para o octante original segue os seguintes passos

1. Se *declive* for verdadeiro, tocar as coordenadas x e y do pixel calculado
2. Se *simetrico* for verdadeiro, substituir o valor da coordenada y do pixel calculado pelo seu valor simétrico

A tabela 2-2 exemplifica os passos da transformação para o primeiro octante de um segmento de recta cujos pontos extremos são P_1 e P_2, por esta ordem, e cujas coordenadas são (11, 5) e (8, 9), respectivamente. Deixa-se como exercício calcular os píxeis resultantes da rasterização do segmento de recta e a verificação da correcção dos resultados obtidos.

Teste	**P_1**	**P_2**	**dx**	**dy**	**simetrico**	**declive**
	(11,5)	(8,9)	-3	4	falso	falso
(dx×dy)<0	(11,-5)	(8,-9)	-3	-4	verdadeiro	falso
|dx|<|dy|	(-5,11)	(-9,8)	-4	-3	verdadeiro	verdadeiro
x1>x2	(-9,8)	(-5,11)	4	3	verdadeiro	verdadeiro

Tabela 2-2 – Transformação para o primeiro octante de um segmento de recta com extremos nos pontos (11,5) e (8,9) antes da aplicação do algoritmo de Bresenham.

3 Rasterização de Circunferências

A rasterização de circunferências emprega a propriedade de simetria que as circunferências apresentam. Observando a figura 3-1, que apresenta uma circunferência de raio R centrada na origem, verificamos que, uma vez calculado o pixel A do segundo octante, os píxeis B a H dos outros octantes encontram-se imediatamente determinados por simetria.

Se (x, y) forem as coordenadas do pixel A, e como a circunferência se encontra centrada na origem, as coordenadas dos píxeis a seleccionar em todos os octantes serão os que a tabela 3-1 apresenta.

Se o centro da circunferência se localizar num ponto (x_c, y_c) que não a origem, bastará calcular os píxeis da circunferência como se ela estivesse centrada na origem e adicionar às suas coordenadas as coordenadas do centro da circunferência.

Para evitar a duplicação de píxeis que ocorrerá quando se representarem os píxeis dos extremos dos octantes, como o pixel $(0, R)$, dever-se-á seleccionar apenas quatro píxeis em vez de oito o que, para este caso, corresponde a seleccionar os píxeis localizados em $(0, R)$, $(0, -R)$, $(R, 0)$ e $(-R, 0)$.