INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

RASTERIZAÇÃO

por

João Manuel Brisson Lopes

Departamento de Engenharia Informática

texto elaborado para a disciplina de

Computação Gráfica

Licenciatura em Engenharia Informática e de Computadores

publicado em Abril de 2004

reeditado em Janeiro e Setembro de 2009, Abril 2013

Este texto, elaborado no contexto da disciplina de Computação Gráfica da Licenciatura em Engenharia Informática e de Computadores do Instituto Superior Técnico, foi originalmente concebido para fazer parte de um conjunto de textos sobre Computação Gráfica, apresentando-se agora como um texto independente.

Contacto do autor: brisson@ist.utl.pt

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.

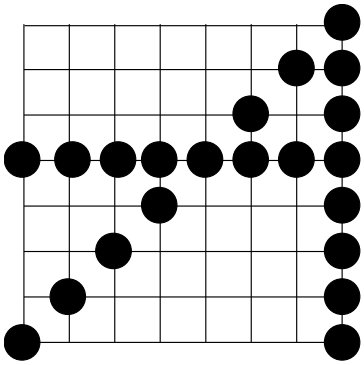


Figura 1-1 – Rasterização de linhas horizontais, verticais e diagonais, de densidade visual variável.

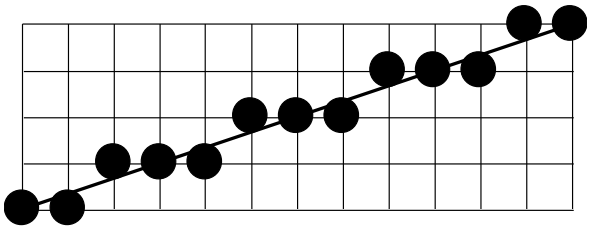


Figura 1-2 – Quadrículas de um segmento de recta rasterizado apresentando o efeito de *aliasing*.

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.

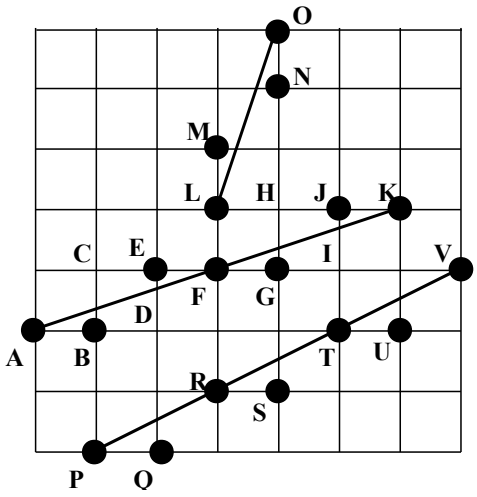


Figura 2-1 – Rasterização de segmentos de recta mostrando píxeis seleccionáveis e píxeis seleccionados.

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

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

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

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

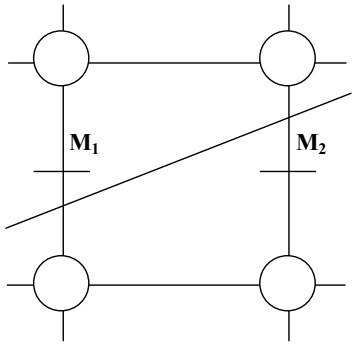


Figura 2-2 – Posicionamento do ponto médio em relação à linha a traçar na selecção de píxeis.

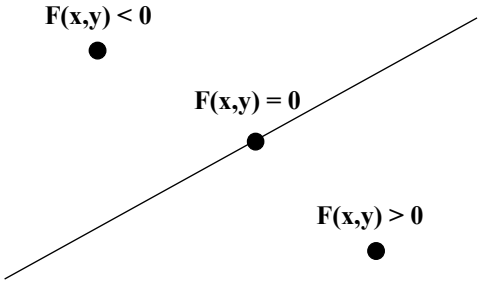


Figura 2-3 – Sinal da função implícita da recta em função da posição de um ponto relativamente à recta.

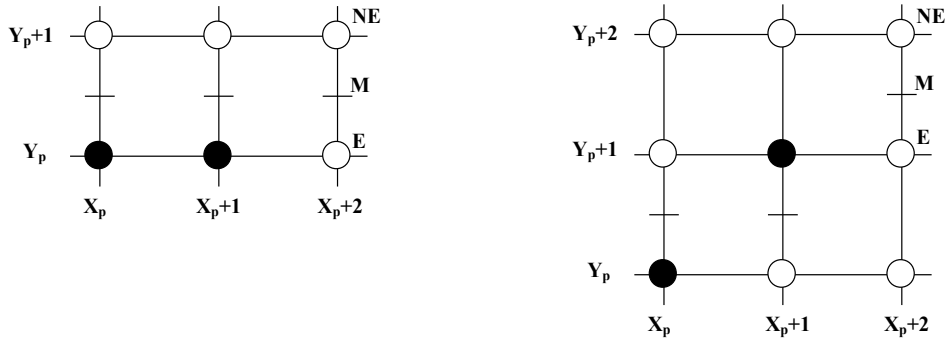


Figura 2-4 – Localização do ponto médio consoante o pixel anteriormente seleccionado (E à esquerda e NE à direita).

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

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

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

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

que será sempre inteiro e as expressões de recursividade serão agora

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

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 $(-R, 0)$.

A	x, y
B	y, x
C	y, -x
D	x, -y
E	-x, -y
F	-y, -x
G	-y, x
H	-x, y

Tabela 3-1 – Rasterização de uma circunferência: relação entre as coordenadas dos píxeis gerados por simetria a partir de um pixel calculado no segundo octante.

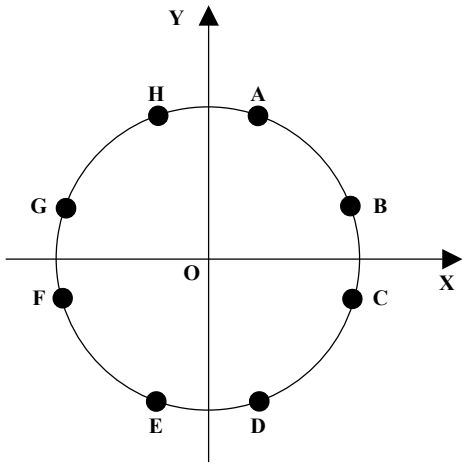


Figura 3-1 – Rasterização de uma circunferência empregando a simetria entre octantes (os píxeis B a H são imagens do pixel A do segundo octante).

A determinação dos píxeis do segundo octante de uma circunferência recorre ao critério do ponto médio. Uma circunferência de raio R centrada na origem do espaço é o lugar geométrico dos píxeis tais que

$$F(x,y) = x^2 + y^2 - R^2 \tag{3-1}$$

seja igual a zero. Esta função é positiva para pontos exteriores à circunferência e negativa para os pontos localizados no seu interior.

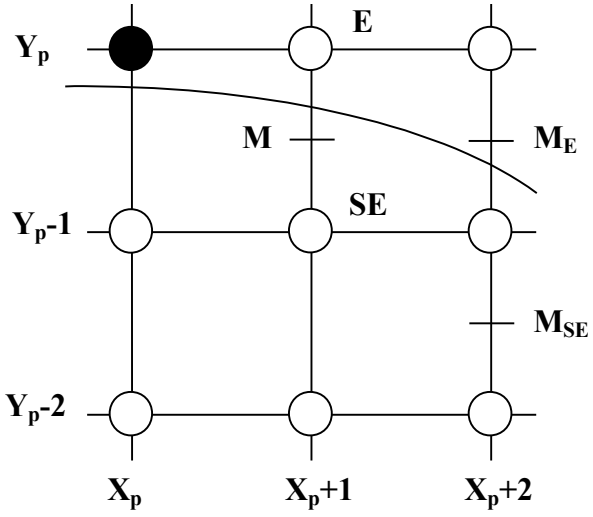


Figura 3-2 – Posições dos pontos médios necessários à rasterização de uma circunferência.

Consideremos agora a figura 3-2 em que se encontra já seleccionado o pixel de coordenadas (x_p, y_p). Para escolher entre os píxeis E e SE da figura para x_p+1 é necessário determinar o sinal de (3-1) avaliado no ponto médio entre estes dois píxeis

$$d_{ant} = F\left(x_p + 1, y_p - 1/2\right) = \left(x_p + 1\right)^2 + \left(y_p - 1/2\right)^2 - R^2 \tag{3-2}$$

Se o valor de (3-2) for negativo, é escolhido o pixel E e, para x_p+2, o teste é então aplicado ao ponto médio M_E, de que resulta

$$\begin{aligned} d_{novo} &= F\left(x_p + 2, y_p - 1/2\right) = \left(x_p + 2\right)^2 + \left(y_p - 1/2\right)^2 - R^2 \\ &= \left(x_p^2 + 4\,x_p + 4\right) + \left(y_p - 1/2\right)^2 - R^2 \\ &= \left(x_p^2 + 2\,x_p + 1\right) + \left(y_p - 1/2\right)^2 - R^2 + 2\,x_p + 3 \\ &= \left(x_p + 1\right)^2 + \left(y_p - 1/2\right)^2 - R^2 + 2\,x_p + 3 \\ &= d_{ant} + 2\,x_p + 3 \end{aligned} \tag{3-3}$$

Quando o valor de (3-2) for positivo, será escolhido o pixel SE e, para x_p+2, o valor de teste no ponto médio M_{SE} será

$$\begin{aligned} d_{novo} &= F\left(x_p + 2, y_p - 3/2\right) = \left(x_p + 2\right)^2 + \left(y_p - 3/2\right)^2 - R^2 \\ &= \left(x_p^2 + 4\,x_p + 4\right) + \left(y_p^2 - 3\,y_p + 9/4\right) - R^2 \\ &= \left(x_p^2 + 2\,x_p + 1\right) + \left(y_p^2 - y_p + 1/4\right) - R^2 + 2\,x_p + 3 - 2\,y_p + 2 \\ &= \left(x_p + 1\right)^2 + \left(y_p - 1/2\right)^2 - R^2 + 2\,x_p - 2\,y_p + 5 \\ &= d_{ant} + 2\,x_p - 2\,y_p + 5 \end{aligned} \tag{3-4}$$

ou seja, para x_p+2, o valor de decisão para seleccionar um novo pixel será o valor de decisão empregue em x_p+1 adicionado de $2\,x_p+3$ ou $2\,x_p - 2\,y_p+5$, ambos calculados em (x_p, y_p), consoante em x_p+1 se tenha escolhido o pixel E ou o pixel SE.

Os valores de decisão iniciais são

$$\begin{aligned} d_0 &= F\left(0, R\right) = 0 \\ d_1 &= F\left(1, R - 1/2\right) = 1 + R^2 - R + 1/4 - R^2 = 5/4 - R \end{aligned} \tag{3-5}$$

Note-se que, enquanto (3-3) e (3-4) permitem o emprego de aritmética de inteiros, pois x_p, y_p e R são inteiros, o valor de $F(1, R-1/2)$ não o é e, portanto, obriga a empregar aritmética de vírgula flutuante.

Se definirmos s tal que $d = s+1/4$, o valor de teste em x_p+1 será $s = 1-R$, que é representável como valor inteiro e todos os testes ao valor de d ($d < 0$) serão substituídos pelo teste $s < -1/4$. Porém, como estamos a empregar aritmética de inteiros, isto equivale a testar se $s < 0$.

A subtracção de $1/4$ não altera portanto o sinal da função de teste devido ao emprego de aritmética de inteiros. Assim, mantendo a consistência de nomenclatura com o

algoritmo para rasterização de segmentos de recta, manteremos a designação de *d* para o valor da função de teste.

O algoritmo para rasterização de circunferências, tal como até agora apresentado, obriga a adicionar os valores de $2\,x_p+3$ ou $2\,x_p - 2\,y_p+5$ calculados em x_p, consoante se tenha escolhido o pixel E ou o pixel SE em x_p+1, ao valor de teste já determinado para x_p+1, para obter o valor de teste em x_p+2. Veremos agora como é possível simplificar o cálculo destes incrementos.

Se em x_p+1 for escolhido o pixel E, o incremento da função de decisão para x_p+2 , se for novamente escolhido o pixel E, será

$$\Delta E\left(x_p+1, y_p\right) - \Delta E\left(x_p, y_p\right) = \left[2\left(x_p+1\right)+3\right] - \left[2\,x_p+3\right] = 2$$

enquanto que, se for escolhido avançar para o pixel SE, o incremento será

$$\Delta SE\left(x_p+1, y_p\right) - \Delta SE\left(x_p, y_p\right) = \left[2\left(x_p+1\right) - 2y_p+5\right] - \left[2\,x_p - 2\,y_p+5\right] = 2$$

Se em x_p+1 tiver sido escolhido o pixel SE de coordenadas (x_p+1, y_p-1), teremos, de forma idêntica,

$$\Delta E\left(x_p+1, y_p-1\right) - \Delta E\left(x_p, y_p\right) = \left[2\left(x_p+1\right)+3\right] - \left[2\,x_p+3\right] = 2$$

$$\Delta SE\left(x_p+1, y_p-1\right) - \Delta SE\left(x_p, y_p\right) = \left[2\left(x_p+1\right) - 2\left(y_p-1\right)+5\right] - \left[2\,x_p - 2\,y_p+5\right] = 4$$

notando que, no ponto inicial de coordenadas $(0, R)$, os valores para os incrementos ΔE e ΔSE são, respectivamente, 3 e $-2R+5$, podemos agora apresentar o algoritmo completo para rasterização de circunferências cujos passos são

1. Calcular
   $d = 1 - R$
   $deltaE = 3$
   $deltaSE = -2R+5$
2. Inicializar $x = 0$ e $y = R$ e marcar os 4 píxeis da circunferência correspondentes.
3. Repetir os passos 4 a 5 enquanto $y > x$.
4. Se $d < 0$
   incrementar *d* de *deltaE*
   incrementar *deltaE* de 2
   incrementar *deltaSE* de 2
   Caso contrário
   incrementar *d* de *deltaSE*
   incrementar *deltaE* de 2
   incrementar *deltaSE* de 4
   decrementar *y* de uma unidade
5. Incrementar *x* de uma unidade e marcar os 8 píxeis da circunferência correspondentes às coordenadas *x* e *y*.

x	y	d	ΔE	ΔSE
0	10	-9	3	-15
1	10	-6	5	-13
2	10	-1	7	-11
3	10	6	9	-9
4	9	-3	11	-5
5	9	8	13	-3
6	8	5	15	1
7	7	6	17	5

Tabela 3-2 – Valores das variáveis do algoritmo de rasterização de circunferências ao rasterizar a circunferência da figura 3-3.

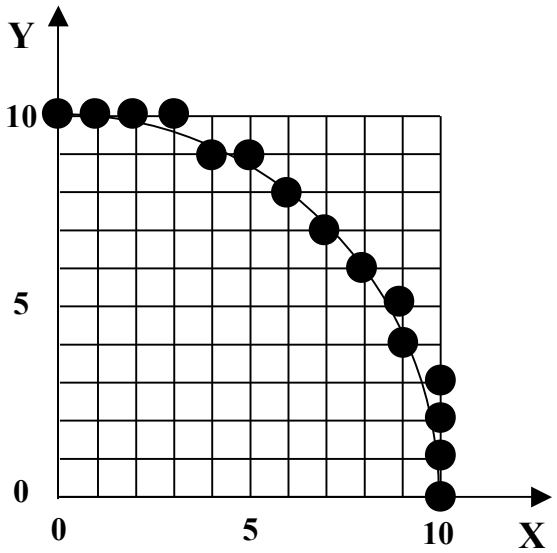


Figura 3-3 – Píxeis do primeiro quadrante de uma circunferência de raio 10 calculados pelo respectivo algoritmo de rasterização.

A figura 3-3 apresenta os píxeis do primeiro quadrante de uma circunferência de raio 10, centrada na origem, que resultam da aplicação do algoritmo de rasterização de circunferências. Os valores das variáveis do algoritmo nos seus vários passos são apresentados pela tabela 3-2.

4 Preenchimento de Polígonos

A operação de preenchimento de polígonos é a operação de rasterização que tem por objectivo atribuir a cor do polígono aos píxeis pertencentes ao polígono[4].

Para um polígono simples (veja-se a figura 4-1), a definição do que constitui o seu interior não apresenta quaisquer ambiguidades, mesmo quando o polígono é multiplamente conexo, tal como a figura mostra.

Porém, quando o perímetro do polígono se intersecta a si próprio, criando um polígono auto-intersectante, a definição do que constitui o interior pode ser ambígua e, portanto, susceptível de diferentes interpretações. Com efeito, se não existem dúvidas sobre o que constitui o interior do polígono A da figura 4-2, o mesmo não se verifica quanto ao interior do polígono B, que pode ser interpretado das duas formas que a figura apresenta.

O algoritmo para preenchimento de polígonos, que a seguir apresentamos, foi concebido para tratar quaisquer polígonos simples, côncavos ou convexos, e polígonos multiplamente conexos. Pode também ser aplicado a polígonos auto-intersectantes desde que a interpretação do que constitui o interior de tais polígonos seja a

[4] Esta cor não é necessariamente uniforme, podendo variar de pixel para pixel, como sucede quando se procede ao sombreamento de um polígono.

interpretação do tipo do polígono B que apresenta uma zona interior não pertencente ao polígono.

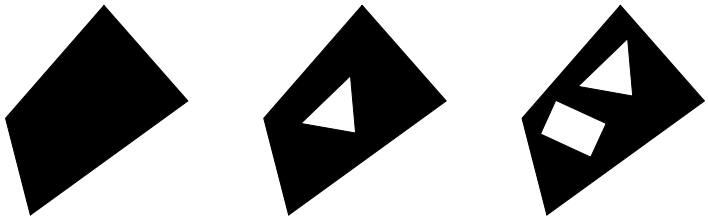


Figura 4-1 – Polígonos preenchidos. Da esquerda para a direita: simples, simplesmente conectado e duplamente conectado,

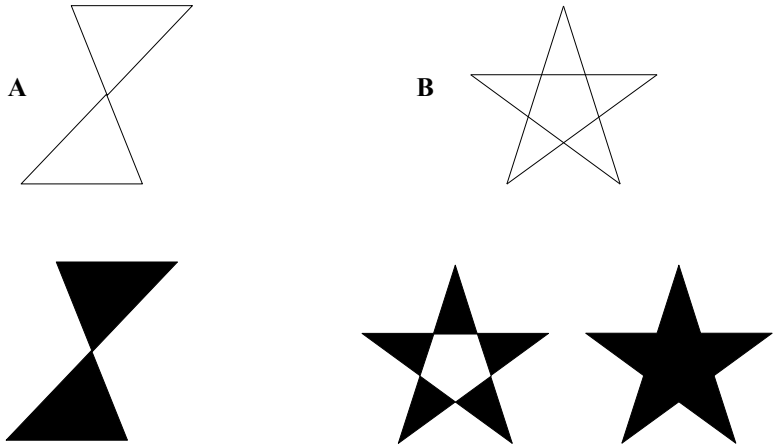


Figura 4-2 – Polígonos auto-intersectantes: identificação do interior única (A) e ambígua (B).

4.1 Algoritmo Básico de Preenchimento de Polígonos

Considere-se a figura 4-3 onde se encontra representado o perímetro de um polígono e uma linha horizontal que corresponde a uma linha de píxeis do dispositivo de quadrícula no qual se pretende rasterizar o polígono da figura.

A linha horizontal é chamada *linha de varrimento* (ou *scan-line*) e intersecta os lados do polígono em pontos bem determinados. Estes pontos, uma vez ordenados por abcissa (coordenada x) crescente, podem ser agrupados em pares de pontos consecutivos. O conjunto dos píxeis existentes entre dois pontos de intersecção consecutivos (x_1 e x_2, por exemplo) é denominado de *span*. Os píxeis de um dado *span* pertencem ao interior do polígono e, portanto, serão os píxeis a que será atribuída a cor do polígono.

Note-se que os valores das abcissas dos pontos de intersecção da linha de varrimento com os lados do polígono não são, em geral, valores inteiros. Isto significa que, para determinar o primeiro e último pixel de um dado *span* há que proceder ao arredondamento dos valores das abcissas dos pontos de intersecção anteriormente determinados. Porque se trata de determinar pontos interiores, o arredondamento deve ser feito por excesso à esquerda e por defeito à direita do intervalo de abcissas.

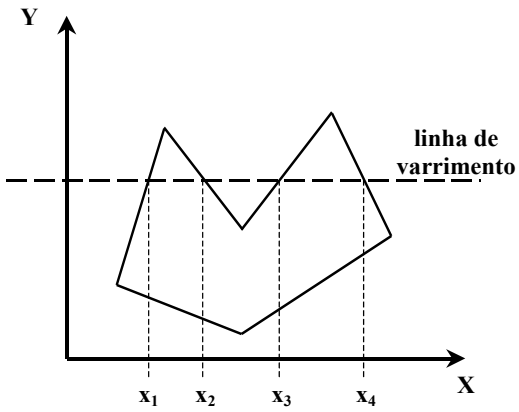


Figura 4-3 – Linha de varrimento e suas intersecções com os lados de um polígono.

Para proceder ao preenchimento do polígono, basta agora considerar todas as linhas de varrimento que intersectam o polígono, isto é, as linhas de varrimento cuja ordenada se situa entre os valores das ordenadas mínima e máxima dos vértices do polígono, e repetir o procedimento anterior para cada uma dessas linhas de varrimento.

4.2 Contabilização de Intersecções

A determinação das intersecções de uma linha de varrimento com os lados de um polígono apresenta dois casos particulares que é necessário considerar: duplicação de intersecções e lados horizontais.

4.2.1 Duplicação de intersecções

O primeiro caso ocorre quando a linha de varrimento intersecta um vértice de um polígono onde, como é natural, concorrem sempre dois lados do polígono. Tal como a figura 4-4 apresenta, isto não constitui qualquer problema para as linhas de varrimento 1 e 3 pois essas linhas intersectam o polígono nos seus vértices de menor e maior ordenada e do cálculo das intersecções resultam sempre dois pontos de intersecção.

Porém, para um vértice como o vértice B que não é nem o vértice superior nem o vértice inferior do polígono, do cálculo da intersecção da linha de varrimento 2 com o polígono resultam 3 pontos de intersecção: um com o lado AB, outro com o lado BC e um outro ponto situado sobre o lado AD. Não havendo dúvida sobre este último ponto de intersecção, torna-se claro que existe uma duplicação de pontos de intersecção em B. Uma forma de evitar esta duplicação seria descartar uma destas intersecções se o vértice não fosse nem superior nem inferior. Mas a detecção desta condição seria uma operação demasiado complicada que teria impacte negativo no desempenho do algoritmo.

A solução para este caso passa por eliminar um dos lados do polígono. Se, por convenção, não se considerar o lado do polígono que num dado vértice atinge o valor máximo da sua ordenada (coordenada *y*), elimina-se aquela duplicação. Para o ponto B da figura 4-4, isto equivale a apenas considerar as intersecções da linha de varrimento com os lados AB e AD do polígono.

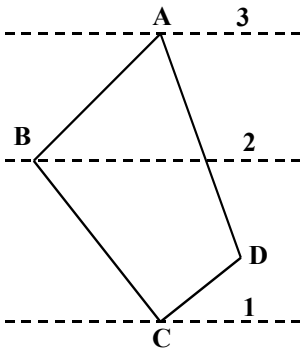


Figura 4-4 – Duplicação de intersecções no vértice B removida por exclusão do lado BC nesse vértice.

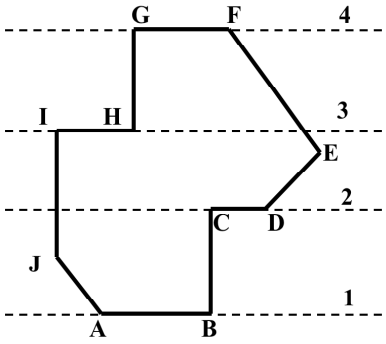


Figura 4-5 – Multiplicidade de intersecções com lados horizontais removida por exclusão de tais lados.

Esta solução apresenta apenas um pequeno (e aparente) inconveniente que consiste em que, no vértice superior de um polígono (veja-se a figura 4-4), já não existam quaisquer lados a serem intersectados pela linha de varrimento e, portanto, o pixel correspondente ao vértice superior não será desenhado. Mas, como na maioria dos casos irão ser desenhados vários polígonos adjacentes, é fácil constatar que o vértice em falta será um vértice não superior de um outro polígono e será desenhado quando esse outro polígono for preenchido.

4.2.2 Lados horizontais

O outro caso particular que é necessário considerar prende-se com a existência de lados horizontais num polígono, pois a intersecção de um lado horizontal com uma linha de varrimento, também horizontal, resulta não num ponto, mas no próprio lado.

Porém, cada um dos vértices de um lado horizontal é também o vértice de um outro lado do polígono e, portanto, existirá um lado à esquerda e outro à direita do lado horizontal. Eliminando este, aqueles dois lados definirão dois pontos de intersecção que criarão um *span* coincidente com o lado horizontal eliminado e assim ficará resolvido o problema.

No caso do polígono da figura 4-5, eliminar-se-ão os lados AB, CD, IH e GF. Os lados AB e CD serão no entanto desenhados, mas os lados IH e GF não. O lado GF não será desenhado porque, para a linha de varrimento 4 da figura, os lados GH e EF já não devem ser considerados por força da regra anterior destinada a evitar a duplicação de intersecções. Mas, tal como para o caso do não desenho do vértice superior de um polígono, um lado horizontal superior será necessariamente um lado de um polígono adjacente ao polígono que foi preenchido e, não sendo agora um lado superior, será desenhado quando esse outro polígono for preenchido. O mesmo sucederá ao lado IH.

4.3 Determinação de Intersecções

A determinação da intersecção de uma linha de varrimento com um lado de um polígono pode ser realizada por substituição do valor da ordenada (coordenada y) da linha de varrimento na equação da recta que suporta o lado do polígono, seguida da

explicitação do valor da abcissa (coordenada *x*). Considerando que um lado tem como extremos os pontos $(x(y_{min}), y_{min})$ e $(x(y_{max}), y_{max})$, em que y_{min} e y_{max} são, respectivamente, o menor e o maior valor das ordenadas dos vértices de um lado e que se definem

$$\begin{aligned} \Delta x &= x\left(y_{\max}\right) - x\left(y_{\min}\right) \\ \Delta y &= y_{\max} - y_{\min} \end{aligned} \tag{4-1}$$

tal equivale a calcular o valor de

$$x = \frac{1}{m}\, y - \frac{1}{m} b \tag{4-2}$$

em que *1/m* e *b* são

$$\begin{aligned} \frac{1}{m} &= \frac{\Delta x}{\Delta y} \\ b &= y_{\min} - m\, x\left(y_{\min}\right) \end{aligned} \tag{4-3}$$

o que envolve uma multiplicação e uma adição para determinar a intersecção de um lado de um polígono com cada linha de varrimento que o intersecte, além do cálculo inicial dos dois coeficientes.

Poderemos simplificar este cálculo considerando duas linhas de varrimento consecutivas, cujas ordenadas são *y* e *y*+1, pois diferem de uma unidade. Assim, o valor da abcissa para *y*+1 será

$$\begin{aligned} x\left(y+1\right) &= \frac{1}{m}\left(y+1\right) - \frac{1}{m} b \\ &= \frac{1}{m}\, y + \frac{1}{m} - \frac{1}{m} b \\ &= \left(\frac{1}{m}\, y - \frac{1}{m} b\right) + \frac{1}{m} \\ &= x\left(y\right) + \frac{1}{m} \end{aligned} \tag{4-4}$$

Isto significa que o valor da abcissa da intersecção de uma linha de varrimento com um lado de um polígono é igual ao valor da abcissa da sua intersecção com a linha de varrimento imediatamente anterior adicionado do valor de *1/m* e, por consequência, disporemos de uma forma recursiva para realizar este cálculo. Naturalmente, o valor inicial da abcissa será o valor da coordenada *x* do vértice de ordenada y_{min}.

Desta forma, torna-se necessário calcular apenas o valor do coeficiente *1/m* e, para cada linha de varrimento, adicioná-lo ao valor da abcissa da intersecção do lado do polígono com a linha de varrimento anterior, o que implica apenas uma adição. Daqui resulta um óbvio aumento do desempenho.

4.4 Lados, Lados Activos e Tabelas de Lados

Uma dada linha de varrimento não pode intersectar todos os lados de um polígono. Por outro lado, uma dada linha de varrimento só pode intersectar um lado de um polígono se a sua ordenada estiver compreendida no intervalo das ordenadas dos vértices desse lado.

Arbitrando que as linhas de varrimento começam a varrer o polígono a partir do menor valor de y dos vértices de um polígono e progridem para o maior valor desta coordenada, só faz sentido determinar a intersecção de uma linha de varrimento com um lado a partir do momento em a ordenada da linha de varrimento se torna igual ao y_{min} desse lado. Por outro lado, deve-se deixar de calcular a intersecção de uma linha de varrimento com um lado logo que a ordenada da linha de varrimento atinja o valor de y_{max} do lado para evitar duplicações de intersecções nos vértices.

Diz-se que um lado se encontra activo quando está a ser intersectado pela linha de varrimento. É apenas sobre os lados activos num dado momento que devem ser calculadas as suas intersecções com a linha de varrimento corrente, pois é inútil tentar calcular intersecções da linha de varrimento com lados que não se encontrem activos.

Assim, poderemos construir uma lista de lados que, inicialmente, contém todos os lados do polígono, excepto os lados horizontais, e uma lista de lados activos que se encontra vazia. Quando um lado se torna activo, esse lado é retirado da lista de lados e colocado na lista de lados activos, passando a pertencer ao conjunto de lados com os quais são calculadas as intersecções. Finalmente, quando a ordenada da linha de varrimento atinge o valor do y_{max} do lado, o lado deve ser retirado da lista de lados activos e eliminado pois não será mais intersectado por qualquer linha de varrimento.

Para facilitar a detecção do momento em que um lado se torna activo, os lados que possuem o mesmo valor de y_{min} devem ser agrupados numa lista, ordenada de preferência segundo o valor da abcissa do vértice para o qual a ordenada tem o valor y_{min}.[5] Por sua vez, estas listas devem ser inseridas num vector com y_{max}-y_{min}+1 posições, na posição correspondendo ao valor da ordenada mínimo dos lados da lista.

Esta lista de lados é chamada *Tabela de Lados*.

A figura 4-6 apresenta um polígono e a respectiva tabela de lados no seu estado inicial. A lista de lados correspondendo ao valor 1 da menor ordenada de vértices contém os lados AB e BC, por esta ordem, dado que o valor de *1/m* para o lado AB (-5/2) é menor que o valor correspondente do lado BC (6/4). Identicamente, ao nível 7, o lado EF deve preceder o lado DE.

À lista de lados activos, chamada *Tabela de Lados Activos*, inicialmente vazia, é adicionada a lista de lados cujo y_{min} é igual ao valor da ordenada da linha de varrimento corrente. Se a tabela de lados activos estiver sempre ordenada segundo as abcissas e porque a lista de lados que lhe é adicionada está também ordenada, é fácil então manter ordenada a tabela de lados activos.

[5] No caso de dois lados apresentarem a mesma abcissa, os lados devem ser ordenados através do respectivo declive.

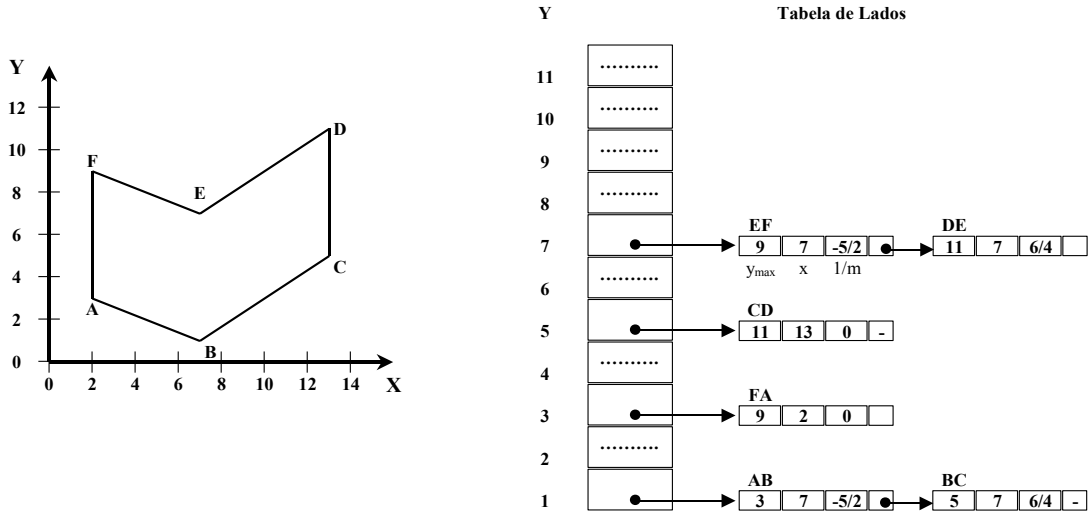


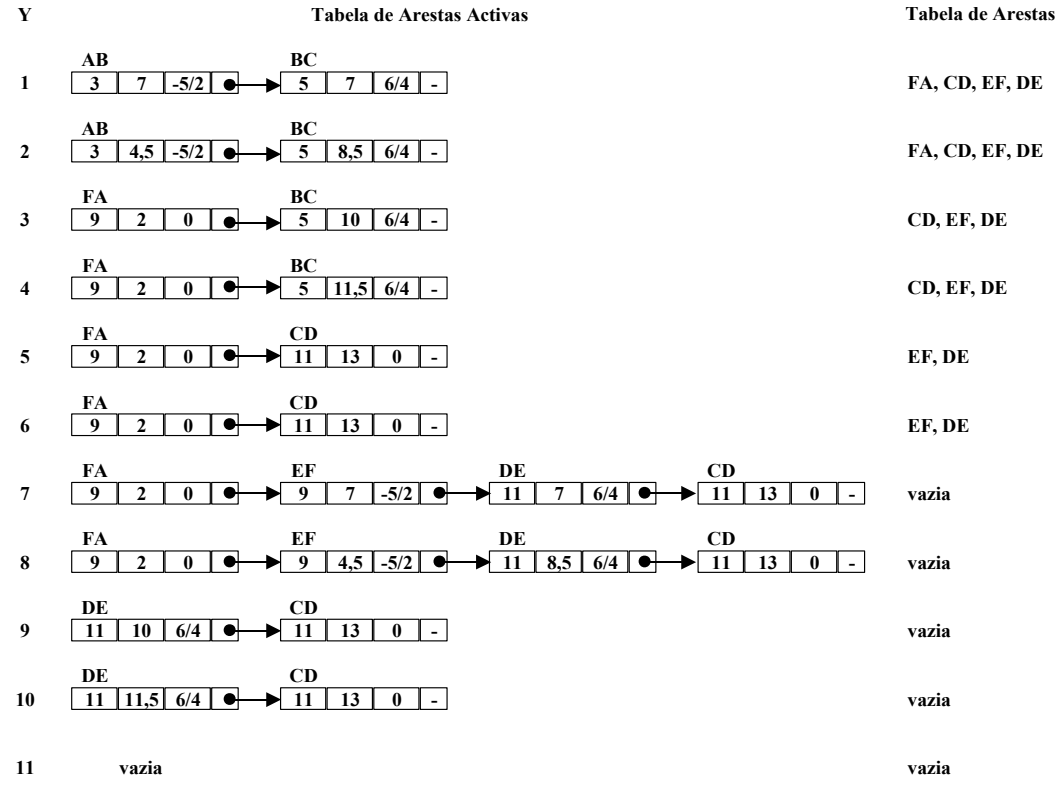
Figura 4-6 – Polígono e respectiva tabela de lados no início do processo de preenchimento.

Também como a figura 4-6 mostra, a estrutura de dados de cada lado deve conter os valores de y_{max}, x e $1/m$, em que, inicialmente, é atribuído a x o valor de $x(y_{min})$. Esta estrutura não necessita de quaisquer outros valores depois de calculado o valor de $1/m$. Por conveniência na construção de listas de lados, é usual incluir nesta estrutura a referência ao lado que se lhe segue na lista.

4.5 Algoritmo de Preenchimento de Polígonos

Em face do anterior, podemos agora apresentar completamente o algoritmo para o preenchimento de polígonos que será

1. Construir a tabela de lados e inserir nela todos os lados do polígono excepto os lados horizontais.
2. Construir a tabela de lados activos, deixando-a vazia.
3. Inicializar Y como o valor da menor ordenada de todos os vértices do polígono.
4. Repetir os passos 5 a 9 até que tanto a tabela de lados como a tabela de lados activos fiquem vazias
5. Mover da tabela de lados para a tabela de lados activos os lados para os quais Y= y_{min}, ordenando a tabela de lados activos.
6. Preencher os *spans* da linha de varrimento corrente utilizando os pares dos valores de x dos lados existentes na tabela de lados activos pela ordem em que os lados se encontrem na tabela.
7. Remover da tabela de lados activos todos os lados para os quais Y é igual a $y_{max} - 1$.
8. Incrementar o valor de Y de uma unidade.
9. Incrementar de $1/m$ o valor de x dos lados existentes na tabela de lados activos.



11. Figura 4-7 – Evolução das tabelas de lados activos (à esquerda) e de lados (à direita) durante o preenchimento de um polígono.

Para poder aplicar este algoritmo a polígonos auto-intersectantes é necessário o seguinte passo adicional

12. Reordenar segundo *x* os lados existentes na tabela de lados activos.

Exemplo

A figura 4-7 exemplifica a evolução da tabela de lados activos, da tabela de lados e dos valores dos campos dos lados quando se procede ao preenchimento do polígono representado na figura 4-6, partindo da menor ordenada do polígono (1) até se atingir a sua maior ordenada (11).

Inicialmente, a tabela de lados activos recebe os lados AB e BC da tabela de lados onde permanecem os lados FA, CD, EF e DE. Da linha de varrimento com ordenada 1 para a linha de varrimento com ordenada 2 apenas ocorre a alteração dos valores do campo *x* dos lados contidos na tabela de lados activos por adição dos respectivos incrementos (*m*).

Ao ser atingida a ordenada de valor 3, o lado AB é eliminado da tabela de lados activos e o lado FA é retirado da tabela de lados e adicionado à tabela de lados activos, onde é colocado antes do lado BC dado que o valor da sua abcissa (2) é inferior ao da abcissa do lado BC (10). Estes lados permanecem na tabela para a linha de varrimento de ordenada 4, apenas com o incremento das respectivas abcissas.

Para a linha de varrimento de ordenada 5, o lado BC é removido da tabela de lados activos e, em seu lugar surge nesta tabela o lado CD, retirado da tabela de lados que passa conter apenas os lados EF e DE. Os lados FA e CD, com as respectivas abcissas

modificadas, permanecem na tabela de lados activos quando a ordenada da linha de varrimento assume o valor 6.

Quando a linha de varrimento tem como ordenada o valor 7, os lados EF e DE são retirados da tabela de lados para a tabela de lados activos. A tabela de lados fica assim vazia. A tabela de lados activos é reordenada pela sequência que a figura apresenta durante a inserção dos lados EF e EF. Os quatro lados (FA, EF, DE e CD) permanecem activos para a linha de varrimento de ordenada 8.

Os lados FA e EF atingem o valor da sua ordenada máxima quando a ordenada da linha de varrimento assume o valor 9 e são eliminados da tabela de lados activos onde permanecem os lados DE e CD. Estes lados continuam nesta tabela apara a linha de varrimento de ordenada 10 mas, quando a ordenada da linha de varrimento atinge o valor 11, têm que ser retirados. A tabela de lados activos fica vazia e, como a tabela de lados já se encontra vazia, o processo de preenchimento do polígono está terminado.

A tabela 4-1 apresenta os intervalos de intersecções definidas por pares de lados consecutivos (*spans*) e os píxeis interiores do polígono para todos os valores da ordenada da linha de varrimento deste exemplo.

Y	Spans	Píxeis
1	[7; 7]	7 a 7
2	[4,5; 8,5]	5 a 8
3	[2; 10]	2 a 10
4	[2; 11,5]	2 a 11
5	[2; 13]	2 a 13
6	[2; 13]	2 a 13
7	[2; 7] [7; 13]	2 a 7 e 7 a 13
8	[2; 4,5] [8,5; 13]	2 a 4 e 9 a 13
9	[10; 13]	10 a 13
10	[11,5; 13]	12 a 13

Tabela 4-1 – Spans entre intersecções consecutivas e píxeis seleccionados para cada ordenada da linha de varrimento do exemplo.

4.6 Fragmentação de Polígonos

No preenchimento de polígonos podem ocorrer situações em que, de um par de pontos de intersecção da linha de varrimento com dois lados de um polígono, resultam um ou nenhum pixel devido à proximidade dos lados. Isto ocorre em polígonos ou zonas de polígonos muito estreitas denominadas *zonas de fragmentação* (*slivers*, em inglês) que podem criar descontinuidades no preenchimento dos polígonos, tal como a figura 4-8 apresenta.

Na figura, o triângulo esquerdo, com os vértices em (1, 5), (4, 5) e (7, 14), perde parte da sua informação para as linhas de varrimento de ordenada 7, 8, 10 e 11 porque,

segundo a regra de preenchimento, só os píxeis interiores ou aqueles que pertençam à fronteira devem ser seleccionados[6]. Esta perda é total junto do vértice superior do triângulo porque não existem quaisquer píxeis interiores ao triângulo para a linha de varrimento de ordenada 13 e a linha de varrimento de ordenada 14 não deve ser considerada devido à regra que elimina os lados activos quando estes atingem a sua maior ordenada. A tabela 4-2 apresenta os píxeis seleccionados para cada linha de varrimento dos triângulos da figura 4-8.

O triângulo à direita na figura, com vértices em (4, 1), (8, 10) e (6, 10), apresenta uma situação ainda mais extrema dado que do seu preenchimento surgem duas zonas separadas. O vértice inferior encontra-se representado mas está completamente desligado do resto do triângulo porque não existem quaisquer píxeis interiores a este para as linhas de varrimento de ordenadas 2 e 3.

Finalmente, o conjunto dos dois triângulos apresenta uma perda de detalhe pois, embora se trate de dois triângulos separados, os seus preenchimentos coalescem para as linhas de varrimento de ordenadas 5 e 9 e, inversamente, ficam demasiado separados para as linhas de varrimento de ordenadas 6, 7 e 8.

Casos como este surgem da aplicação das regras para preenchimento de triângulos, mas não são consequência de tais regras. A sua origem encontra-se no *aliasing* porque a quadrícula empregue não é suficientemente fina para capturar os detalhes dos objectos. É aos algoritmos de *antialiasing*, que a seguir apresentamos, que cabe apresentar soluções para estes problemas.

Y	span Δ esquerdo	span Δ direito
14	-	
13	-	
12	6-6	
11	5-5	
10	5-5	-
9	4-5	6-8
8	3-4	6-7
7	3-4	6-7
6	2-4	6-6
5	1-4	6-6
4		5-5
3		-
2		-
1		4-4

Tabela 4-2 – Spans dos píxeis seleccionados no preenchimento dos triângulos da figura 4-8.

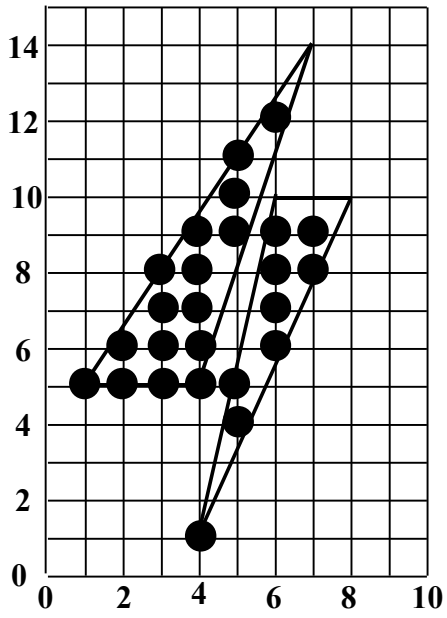


Figura 4-8 – Preenchimento de dois triângulos de reduzida largura com perda de detalhe.

[6] Para as ordenadas 8 e 11, ocorre ainda o facto do declive do lado direito do triângulo apresentar um declive inverso (1/3) que é aproximado por defeito.

5 Aliasing e Antialiasing

A discretização de uma grandeza contínua implica perda de informação. Por vezes a informação perdida não é significativa e é possível reconstruir quase fielmente a grandeza original mas, a partir de uma certa perda, os valores discretos medidos são insuficientes para que se possa proceder à reconstrução da grandeza original. Quando tal sucede, a reconstrução produz informação que pouco ou nada tem a ver com a informação original. Este fenómeno tem o nome de *aliasing*.

Um exemplo simples de *aliasing* é o resultado da amostragem de um sinal sinusoidal com uma frequência de amostragem idêntica à frequência do sinal. Os valores amostrados são todos iguais e qualquer tentativa de reconstrução do sinal original resultará num sinal sem qualquer relação com o original (veja-se a figura 5-1) pois, não só se perdeu a amplitude do sinal, como igualmente se perdeu a sua própria periodicidade. Este exemplo mostra que a frequência de amostragem deve ter em conta a periodicidade do sinal amostrado. Se, por exemplo, se tivessem tomado quatro amostras por período, conforme a figura 5-2 apresenta, teria sido possível reconstruir quase exactamente o sinal original. No entanto, se aumentarmos a frequência do sinal e mantivermos a frequência de amostragem, verificamos que esta deixou de ser adequada, pois o sinal que é possível reconstruir apresenta uma frequência mais baixa do que a frequência original, ainda que a informação sobre a amplitude do sinal tenha sido correctamente reconstruída.

Estes exemplos ilustram o essencial do teorema de Nyquist. Segundo este teorema, a frequência mínima de amostragem deve ser o dobro da frequência do sinal amostrado para que se possa reconstruir o sinal original. É por esta razão que nas gravações digitais de música em áudio CD, realizadas com a frequência de amostragem de 44 kHz, todas as frequências superiores a 22 kHz, que não são audíveis, são eliminadas do sinal antes do seu registo, pois, caso contrário, seriam reproduzidas como frequências mais baixas que iriam contaminar o som escutado.

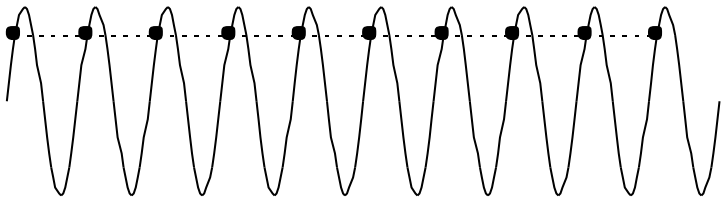


Figura 5-1 – Amostragem de um sinal sinusoidal com frequência de amostragem igual à do sinal. A reconstrução do sinal a partir das amostras discretas conduz a um sinal contínuo cujo valor nada tem a ver com a amplitude do sinal original.

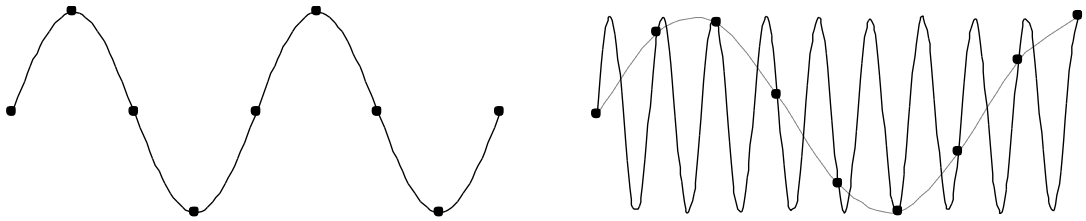


Figura 5-2 – Amostragem com frequência constante. A amostragem é suficiente para o sinal à esquerda e é insuficiente para o sinal à direita.

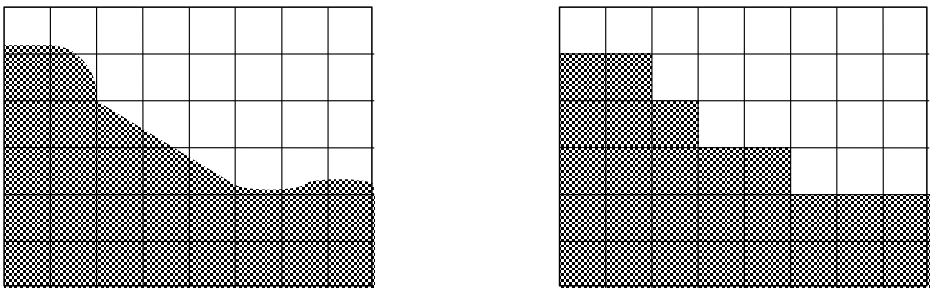


Figura 5-3 – Perda de detalhe devida ao "efeito de escada" na fronteira de um objecto.

5.1 Aliasing em Computação Gráfica

O fenómeno de *aliasing* ocorre em Computação Gráfica, uma vez que a informação contínua dos objectos a representar graficamente é discretizada e representada associada a píxeis nos dispositivos de quadrícula. Uma das formas mais comuns de *aliasing* em Computação Gráfica consiste no chamado "efeito de escada" ou "efeito de serra" que ocorre quando se representam linhas que não sejam horizontais, verticais ou diagonais, como a figura 1-2 apresenta. Este efeito (veja-se também a figura 5-3) aparece ainda na representação de polígonos, mais propriamente nas arestas, e é particularmente visível quando existe grande contraste entre as cores do polígono e do exterior.

A perda de detalhe das imagens, quando a densidade dos píxeis é insuficiente (ou não suficientemente fina) para reproduzir os detalhes ou mesmo os objectos a representar, é também devida ao *aliasing*. Um exemplo desta ocorrência foi já anteriormente apresentado (figura 4-8). Um outro exemplo de perda de detalhe ocorre quando se pretende representar um conjunto de triângulos, de largura de base δ e uma altura de várias unidades, que se encontram justapostos. Se o espaçamento horizontal dos píxeis corresponder, por exemplo, a 1,05 δ, a representação apresentará o aspecto de um triângulo cuja base é de 21 δ, conforme a figura 5-4 apresenta.

O *aliasing* ocorre porque, de acordo com o teorema de Nyquist, deveríamos ter empregado uma quadrícula em que os píxeis estivessem espaçados de 0,5 δ, em vez de 1,05 δ. Em princípio, a perda de detalhe devida ao *aliasing* poderia ser eliminada empregando uma quadrícula mais fina. No entanto, poderemos sempre questionar se na imagem não poderão existir outros detalhes de ainda menor dimensão que criam, por

sua vez, artefactos de baixa frequência e que nos obrigariam a empregar quadrículas ainda mais finas.

Um outro exemplo de perda de detalhe devido ao *aliasing* pode ser observado quando se representam texturas regulares cuja dimensão aparente diminui com o aumento da distância ao observador, como é o caso de um chão de ladrilhos alternadamente brancos e negros que a figura 5-5 apresenta. Os ladrilhos deveriam diminuir de tamanho e manter a sua forma à medida que ficassem mais distantes. Em vez disso, aparecem formas irregulares e de tamanhos maiores devido à coalescência de ladrilhos próximos. O aumento da resolução da imagem não elimina o problema pois o problema persiste para ladrilhos mais distantes.

A mesma constatação pode ser obtida observando a figura 5-6. O emprego de maior resolução não elimina o "efeito de escada" pois surgem mais "patamares", embora de menor dimensão. Também a mancha visual do segmento de recta se torna menos nítida porque a sua espessura aparente diminui.

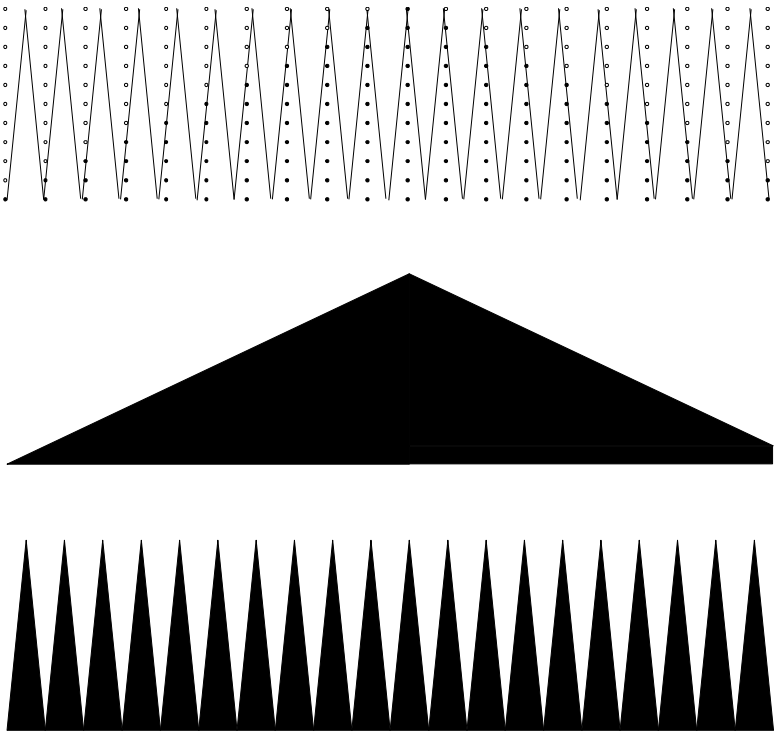


Figura 5-4 – Rasterização de um conjunto de triângulos com uma quadrícula (topo), com resultado incorrecto (ao meio) por insuficiência da quadrícula onde se perdeu completamente o detalhe da imagem correcta (em baixo).

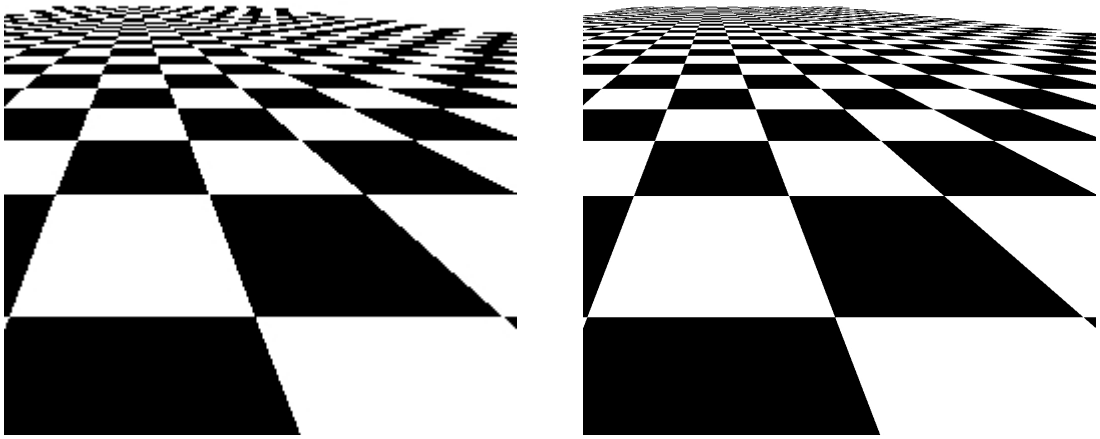


Figura 5-5 – Perda de detalhe de texturas com o aumento da profundidade (à esquerda) numa imagem de 256x256 píxeis. O aumento da resolução da imagem (à direita) para 1024x1024 píxeis apenas consegue relegar esta perda para distâncias maiores, mas não a elimina.

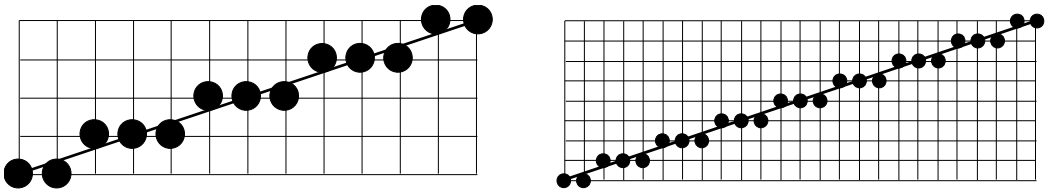


Figura 5-6 – Traçado de segmentos de recta com diferentes resoluções: com maior resolução, o segmento recta aparenta maior fidelidade mas é mais fino.

5.2 Antialiasing

Como vimos, o fenómeno de *aliasing* tanto pode ter origem na discretização no espaço (*aliasing espacial*), como na discretização no tempo (*aliasing temporal*). Neste capítulo trataremos apenas da correcção dos efeitos do *aliasing* espacial.

Existem várias técnicas designadas genericamente por *técnicas de antialiasing* ou *algoritmos de antialiasing* para resolver de modo satisfatório, mas nunca total, os problemas criados pelo fenómeno de *aliasing* nas suas várias manifestações. Estas técnicas baseiam-se na filtragem da informação que pode ser realizada tanto no espaço da imagem como no espaço dos objectos. A filtragem realizada no espaço dos objectos designa-se por pré-filtragem, enquanto a realizada a partir da informação que existe no espaço da imagem é designada por pós-filtragem.

5.2.1 Pré-filtragem

As técnicas de pré-filtragem são muito empregues pelas técnicas de geração de imagem que operam no espaço dos objectos, como é o caso da técnica de Ray Tracing.[7]

De uma forma genérica, a pré-filtragem consiste em determinar a cor a atribuir a um pixel por exame da área que lhe corresponde no espaço dos objectos, determinando as cores empregues nessa área e a respectiva proporção. Considere-se, por exemplo, um objecto verde sobre um fundo de cor vermelha. Aos píxeis correspondentes a áreas de cor uniforme (verde ou vermelho) será atribuída a cor da área respectiva. A cor a atribuir a píxeis correspondentes a áreas na fronteira do objecto, que têm parte a verde e parte a vermelho, deverá ser uma mistura de verde e vermelho nas proporções existentes na respectiva área. Assim, numa área em que 30% esteja dentro do objecto de cor verde e 70% fora dele, a cor do pixel correspondente na imagem deverá ser uma mistura de 30% de verde com 70% de vermelho.

A pré-filtragem que acabamos de descrever, denominada *pré-filtragem não ponderada*, produz resultados bastante correctos para primitivas que ocupam áreas da imagem correspondendo a um número razoável de píxeis, como é o caso de polígonos.

Porém, a pré-filtragem não ponderada não é directamente aplicável à rasterização de primitivas como menos do que uma dimensão, como é o caso dos segmentos de recta e pontos, pois, não tendo espessura, tais primitivas não contribuem para os píxeis da imagem dado não ocuparem qualquer área no espaço dos objectos.

Para ultrapassar este problema, teremos que considerar que pontos e segmentos de recta possuem duas dimensões e criar no espaço dos objectos círculos e rectângulos que os representam, atribuindo, por exemplo, a largura de um pixel ao diâmetro dos círculos e à espessura dos segmentos de recta. Este procedimento permite então rasterizar estas primitivas por meio da pré-filtragem não ponderada.

A figura 5-7 permite comparar o resultado da rasterização de um segmento de recta com um declive de 1/3 realizada de diferentes formas. Em (a) o segmento apresenta-se tal como resultaria da sua rasterização através do algoritmo de Bresenham, enquanto em (b) se pode observar a rasterização do mesmo segmento aplicando a técnica de pré-filtragem não ponderada, em que se atribuiu ao segmento a largura correspondente a um pixel.[8] O resultado visual é uma linha em que se nota uma atenuação do "efeito de escada" e uma espessura mais uniforme do que em (a), mas ainda se nota alguma irregularidade. Isto sucede porque os valores dos píxeis a que se sobrepõe o rectângulo representando o segmento são calculados sem atender à distância a que se encontram do segmento de recta original.

A pré-filtragem com ponderação entra em conta com esta distância. Em (c) foi aplicado um filtro de ponderação da intensidade dos píxeis em função da distância dos seus centros ao segmento de recta, em que a intensidade varia inversamente com a distância segundo Gupta e Sproull. Como se pode observar, o segmento de recta apresenta-se agora mais nítido e uniforme em toda a sua extensão.

[7] Para mais detalhes sobre a pré-filtragem em Ray Tracing, consultar o opúsculo dedicado a este tema.

[8] A figura deverá ser observada a alguns metros de distância para que os pixéis possam fundir-se visualmente.

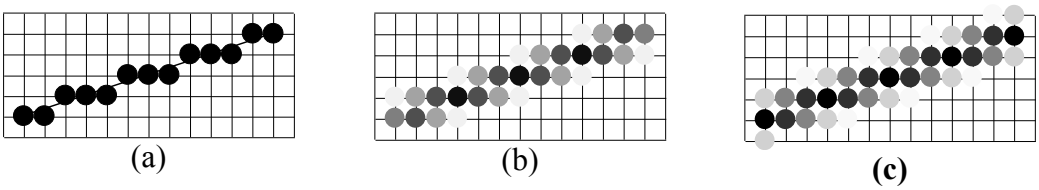


Figura 5-7 – Pré-filtragem de segmento de recta: sem pré-filtragem (a), com pré-filtragem não ponderada (b) e pré-filtragem ponderada empregando filtro cónico (c).

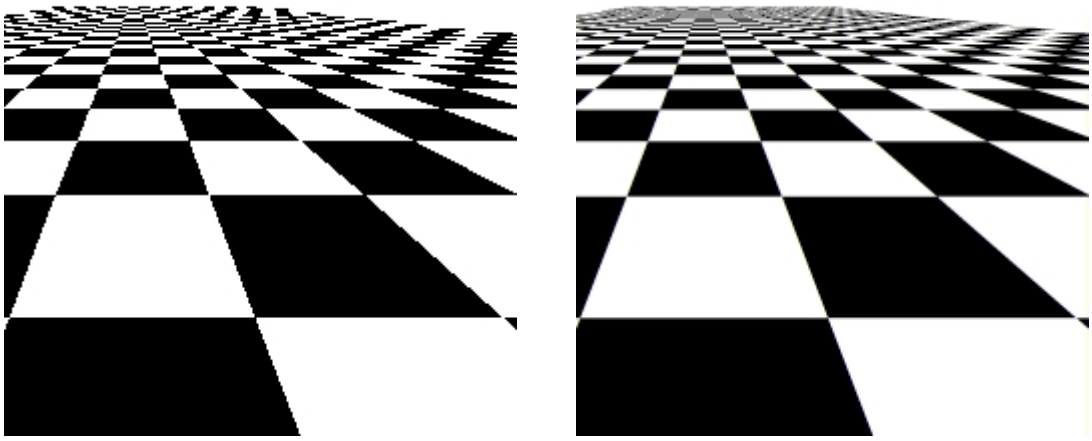


Figura 5-8 – Sobre amostragem. Imagem com a resolução de 256x256 píxeis (à direita) obtida a partir de uma imagem com a resolução de 1024x1024 píxeis. Compare-se este resultado com uma imagem da mesma resolução (à esquerda) sem sobre amostragem.

5.2.2 Pós-filtragem

As técnicas de pós-filtragem, também designadas por técnicas de *sobre amostragem* (*super sampling*), consistem em gerar uma imagem de resolução superior à resolução da imagem pretendida e, em seguida, usar a informação de vários píxeis contíguos para, através da média das suas cores, gerar os píxeis da imagem final. Para uma imagem de 512 x 512 píxeis gerar-se-iam, por exemplo, imagens de 1024 x 1024 ou 1536 x 1536 píxeis com uma resolução respectivamente dupla ou tripla da resolução final pretendida. A seguir, para calcular a cor de cada pixel da imagem final, seria determinada a cor média das cores dos quatro ou nove píxeis correspondentes das imagens de maior resolução. A figura 5-8 apresenta um exemplo de aplicação de pós-filtragem onde se observa uma nítida melhoria da representação dos polígonos localizados no fundo da imagem.

As técnicas de pós-filtragem são muito empregues em Computação Gráfica devido à sua simplicidade, embora sejam computacionalmente mais pesadas porque requerem o emprego de mais memória e a carga computacional aumenta devido a ser necessário

processar um muito maior número de píxeis e, no final, ser também necessário combinar num só pixel da imagem final a informação de vários píxeis da imagem em resolução aumentada

Estas técnicas não são aplicáveis a primitivas com menos de duas dimensões (segmentos de recta e pontos) pois, tal como a figura 5-6 apresenta, o emprego de uma resolução maior não elimina o "feito de escada" dado que surgem mais "patamares", ainda que estes sejam de menor dimensão. Por outro lado, a mancha visual do segmento é muito mais fina e, se se aplicasse a pós-filtragem, obteríamos um segmento cuja intensidade seria bastante inferior à intensidade original.

Exercícios

1 Apresente as diferenças fundamentais entre o algoritmo incremental básico e o algoritmo de Bresenham para a rasterização de segmentos de recta e comenta a influência que estas diferenças têm no desempenho relativo dos dois algoritmos.

2 Descreva as restrições impostas à aplicação do algoritmo de Bresenham e explique como estas restrições devem ser ultrapassadas para que seja possível rasterizar um qualquer segmento de recta.

3 Determine os píxeis que devem resultar da aplicação do algoritmo de Bresenham ao segmento de recta da tabela **2-2** depois de devolvidos ao octante original e verifique se a linha de píxeis resultante é correcta.

4 Por semelhança com o algoritmo para rasterização de circunferências, conceba um algoritmo para a rasterização de elipses sabendo que a sua função implícita é $b^2x^2 + a^2y^2 - a^2b^2 = 0$, não considerando aritmética de inteiros. (Sugestão: estude apenas o primeiro quadrante e tenha em atenção que deve alterar o critério do ponto médio quando a elipse apresente um declive igual a –1.)

5 Construa a tabela de arestas para preencher o triângulo com vértices nos pontos (1, 2), (10, 12) e (1, 15) e descreva o conteúdo da lista de arestas activas para cada uma das linhas de varrimento que intersecta o triângulo, explicando o objectivo de cada uma destas tabelas.

6 Explique como e porquê devem ser contabilizadas as intersecções da linha de varrimento com vértices e com lados horizontais no preenchimento de polígonos e apresente as respectivas consequências.

7 Determine os píxeis correspondentes ao preenchimento do polígono com vértices em (1, 5), (3, 8), (2, 8), (1, 6) e (3, 6), e critique o resultado obtido.

8 Conceba um algoritmo para o preenchimento de círculos com base no algoritmo para rasterização de circunferências que evite o preenchimento de um pixel mais do que uma vez.

9 Explique a razão pela qual a representação de primitivas gráficas, como segmentos de recta, em dispositivos de saída gráfica do tipo de quadrícula, coloca problemas que, em princípio, não ocorrem em dispositivos de saída gráfica do tipo vectorial.

10 Uma série de terminais gráficos com ecrãs de dispositivo de quadrícula proporciona um espaço de endereçamento de 4096×3072 quadrículas, mas o espaço de endereçamento do ecrã dispõe apenas de 1024×762 quadrículas. Explique que problema pretende o fabricante resolver e apresente as estratégias que poderão estar por detrás desta solução.