1. Histórico e Aplicações da Computação Gráfica

1.1. Definições

Métodos e Técnicas de converter dados para um dispositivo via computador

ou

Sub-área da Ciência da Computação que consiste me métodos e técnicas usadas para criar, armazenar e manipular modelos de objetos e suas imagens via computador.

1.2. Nomes Usuais

Compugrafia: Romero Tori

Gráfica Computacional: Harold Santos

1.3. Breve Histórico

1950: Início da Computação Gráfica, pela primeira vez, um tubo de raios catódicos é ligado à um Computador, on MIT e Projeto Sage, onde um radar foi ligado à um Computador, num projeto do exército americano, utilizando interação com Light Pen;

1959: Surge o termo Computer Graphics - criado por Verne L. Hudson, quando o mesmo coordenava um projeto para a Boeing de simulação de fatores humanos em aviões;

1962: No MIT o projeto TX2 com o aluno Ivan Sutherland usa um computador para fazer desenhos em sua tese de doutorado;

1970: Surgem os primeiros monitores à varredura.

1.4. Aplicações da C.G.:

— Análise de dados;

— Computação Científica;

— CAD / CAM e CAE;

— Simulação Visual;

— Processamento de Textos;

— Entretenimento;

— Medicina;

— Outros.

1.5. Equipamentos para Computação Gráfica:

1.5.1. Introdução

Dados Gráfico podem ser classificados em:

Vetoriais: Dados representados por pontos coordenados e linhas, capazes de ligar tais pontos;

Matriciais: Consiste de uma matriz de pontos no espaço e a cada ponto é associado um atributo de Intensidade;

1.5.2. Representação X Apresentação:

Chama-se representação da imagem à estrutura de dados que a contém **e**

Apresentação da imagem à relação da mesma com o equipamento que a reproduz;

1.5.3. Equipamentos de entrada

A entrada pode ser diferenciada em 2 categorias diferentes: coordenadas relativas x coordenadas absolutas.

VETORIAL:

Coordenadas relativas	- mouses (ópticos e mecânicos) - joysticks, bolota e diais
Coordenadas absolutas	- teclados - mesas digitalizadoras - caneta óptica - “light pen” - tela óptica - “touch screen”

MATRICIAL:

- Digitalizadores de vídeo: a partir de um sinal de televisão gera uma matriz de pontos em um monitor de vídeo; - FRAME GRABBER -
- Varredores digitais (SCANNER): são baseados na absorção da luz; versão do FRAME GRABBER para papel.
- FILM SCANNER: o equivalente para filme.

1.5.4. Processadores Gráficos

1.5.4.1. Introdução:

Na década de 60 ⇒ Tecnologia da Computação Gráfica discute o fato de um computador, ao invés de manusear caracteres, manusear pixel (pontos individuais da tela).

Fácil ⇒monitores para branco.

Cor ⇒ além de controlar pontos acesos e apagados, era necessário controlar-se as cores.

Processadores gráficos necessitam administrar dispositivos de visualização complexos, diferente de processadores que se destinam só ao armazenamento e manuseio de processamento matemático. Assim, computadores gráficos necessitam, além da unidade de processamento central, de módulos dedicados exclusivamente ao gerenciamento do dispositivo de visualização.

Problemas:

- No modo gráfico; a máquina deve oferecer recursos para representação de algumas primitivas;
- A máquina, no modo gráfico, perde o acesso ao gerador de caracteres alfanuméricos, logo é necessário prover meios para tal solução.

Definição: Processador gráfico é a parte do hardware responsável pela integração do gerenciamento simultâneo da tela e dos recursos gráficos mínimos residentes (no hardware).

- bit map: múltiplos planos de bits que armazenam a imagem gerada de forma digital;

Requisitos mínimos -circuitos de varredura seqüencial: traduzem este mapa ao processo gráfico digital para sua visualização física na tela;

- circuitos que permitem manusear / modificar este mapa;

–

1.5.5. Equipamentos De Saída

1.5.5.1.Plotters:

traçadores gráficos (de mesa e de tambor)

1.5.5.2.Terminais de vídeo

Atualmente: CRT ⇒ Tubos de raios catódicos ⇒ princípio da televisão.

Um cátodo aquecido emite um feixe eletrônico que é convenientemente focalizado por um conjunto de lentes e acelerado. Ao incidir na superfície de tubo, que é recoberta de fósforo: pontos bombardeados pelos elétrons emite luz e a associação de pontos acesos forma a imagem. O feixe eletrônico é desviado pela ação de campos magnéticos produzidos por bobinas à qual é aplicada uma d.d.p. proporcional ao desvio desejado.

Característica dos terminais de vídeo:

i) Resolução: n° de pontos distintos que pode ser aceso, que depende da capacidade do sistema computacional que gerencia o display. Propriedade: **aspect ratio** ⇒ aspect ratio = x/y ⇒ uma linha vertical de x pontos é de mesmo de uma linha horizontal de y pontos;

ii) Persistência do fósforo: capacidade do elemento químico reter a luminosidade;

iii) Sistema de deflexão: controlador do esboço produzido na tela;

MÉTODOS DE VARREDURA

2 metodologias básicas para geração de imagens na tela gráfica:

- Varredura randômica ou vetorial - random/vector scan
- Varredura rastreada ou matricial - raster scan

⇒ Até 1980 ⇒ prevalecia equipamento de varredura randômica ⇒ método ideal para representar vetores, típicos das aplicações de CAD.

⇒ Hoje ⇒ predominam equipamentos de varredura matricial.

Funcionamento:

- Random-scan: o feixe de elétrons é direcionado somente para as partes da tela onde a figura é desenhada.
- Raster-scan: o feixe de elétrons é disparado sobre todas as partes da tela, sendo a tela criada por um conjunto de pontos ⇒ processo da televisão ⇒ varre-se a tela

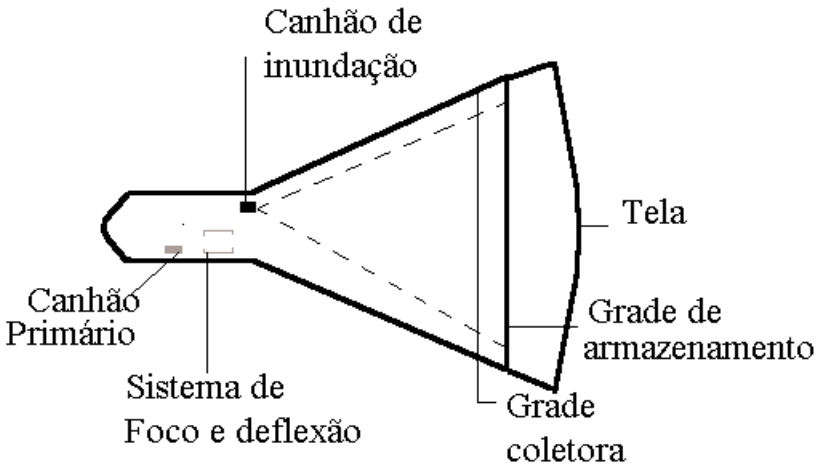
de cima para baixo e da esquerda para a direita marcando os trechos visíveis durante o trajeto.

TV
- PAL - 256 linhas de varredura
- NTSC - 525 linhas de varreduras

CONAP
- 256 → 2048, já chegando a 4096 linhas.

MONITORES COLORIDOS

Composto de 3 camadas de fósforo diferentes e sensíveis às cores "vermelho, verde e azul". O tubo possui 3 diferentes canhões de luz. Estes canhões podem ser montados numa configuração triangular ou linear. Os fósforos na tela são formados como tríades. A convergência é feita por uma placa metálica chamada "shadow-mask".



DSVT - "Direct View Storage Tube"

Funciona como CRT convencional. Há, no entanto, no seu interior, um monitor auxiliar (o canhão de inundação) - “Flood Gun”.

Funcionamento: O canhão primário é usado para desenhar a definição da imagem sobre a grade de armazenamento, formada de material não-condutor. Elétrons de alta velocidade do canhão primário colidem a grade de armazenamento fazendo sair alguns elétrons que são atraídos para a grade coletora, de carga positiva. Como a grade de armazenamento é não-condutora, a área onde os elétrons foram removidos manter-se-a positiva. Esta área positiva armazenada é a definição da figura. O canhão de inundação produz um contínuo jato de elétrons de baixa velocidade que passam através da grade coletora e são atraídos para as áreas positivas da tela de armazenamento. Estes elétrons atingem a tela de fósforo através da grade de armazenamento sem afetar sua carga.

Para se apagar a imagem basta aplicar uma carga positiva elevada na superfície de armazenamento. Uma enxurrada de elétrons de inundação é atraída para ela, apagando a imagem e causando um forte clarão, como se explodisse um FLASH.

DISPLAYS DE CRISTAL LÍQUIDO

Fundamentos: **Painel de Plasma**: Uma camada de gás neon, misturada com outros gases é comprimida entre duas placas de vídeo. Essas placas possuem no seu interior fileiras de condutores milimétricos. O gás é ionizado pela passagem de corrente elétrica nestes condutores. Ao passar por uma interseção a corrente provoca emulsão de luz pelo gás.

No caso do display de cristal líquido, as moléculas de cristal líquido encontram-se, inicialmente, sem direcionamento. Aplicando-se um potencial elétrico aos condutores, tais moléculas se arranjam, impedindo a passagem de luz através das

superfícies de polarização, com isto, certas células tornam-se escuras, mostrando a figura nos cruzamentos.

1.5.5.3.Impressoras:

- de impacto
- laser

2. SISTEMAS DE COORDENADAS

2.1. RESOLUÇÃO

Todos dispositivos gráficos usam uma grade retangular de localizações endereçáveis. Este retângulo é chamado de "*Display Rectangle*" ou "*Graphics I/O Rectangle*". Dispositivos gráficos são fixados de acordo com sua resolução gráfica, o número de horizontais versus posições verticais.

Parâmetros importantes: características gráficas básicas;

1. ndh => Nº de localizações gráficas endereçáveis horizontalmente;
2. ndv => Nº de localizações gráficas endereçáveis verticalmente;
3. width => a largura física de retângulo em milímetros;
4. height => a altura física de retângulo em milímetros.

Geral: monitor PC

width - 245,0	ndh - 640
heigth - 186,0	ndv - 200

Baseado nestes conceitos, define-se:

1. Resolução horizontal

   Res-horiz = ndh/width

2. Tamanho de pontos na horizontal (horizontal dot size)

   Hor-dot-size = width/ndh

3. Resolução vertical

   res-vert = ndv/height

4. Tamanho de pontos na vertical (vertical dot size)

vert-dot-size = height/ndv

5. Total de pontos endereçáveis

total-nr-dot = ndv*ndh

6. Graphics aspect ratio

aspect-ratio = vert-dot-size/horiz-dot-size

7. Physical aspect ratio

Phys-aspect-ratio = height/width

2.2. SISTEMAS DE COORDENADAS DO USUÁRIO

Chama-se coordenada do usuário ao sistema de coordenadas que um usuário escolhe para trabalhar.

mais freqüente := sistema cartesiano

outra opção := sistema polar

Para o GKS => Sistema cartesiano => chamado de world coordinates (WC).

Os objetos (gráficos) são especificados no sistema de coordenadas do usuário e devem ser convertidos em coordenadas apropriadas do dispositivo físico.

A porção do desenho que deve aparecer na tela e chamada de JANELA. Uma janela é geralmente definida por seus limites mais altos e menores.

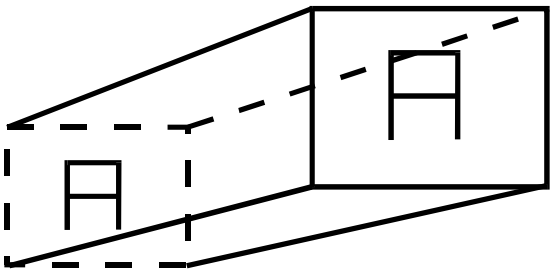
Formato geral da definição de uma janela:

WINDOW(min-x,max-x,min-y,max-y)

A janela pode apresentar porção do objeto, objeto inteiro ou objeto mais área vazia.

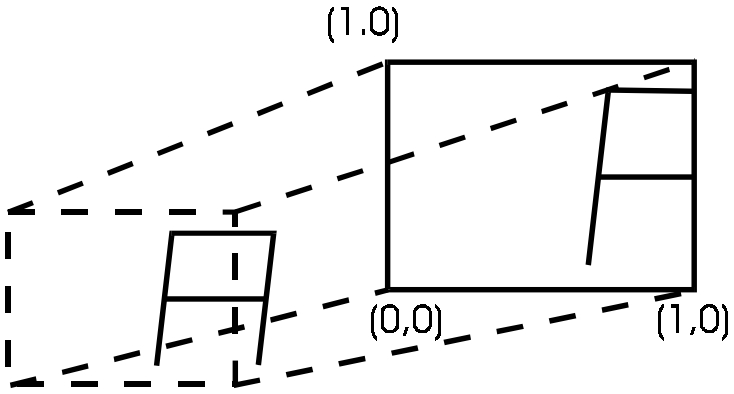
Wraparound => envolver a vizinhança

clipping => efeito pelo qual há uma porção visível do objeto na janela e porções invisíveis de objeto fora da janela.



JANELA POSSIBILITA
VISÃO DE TODO
(NO CLIPPING)

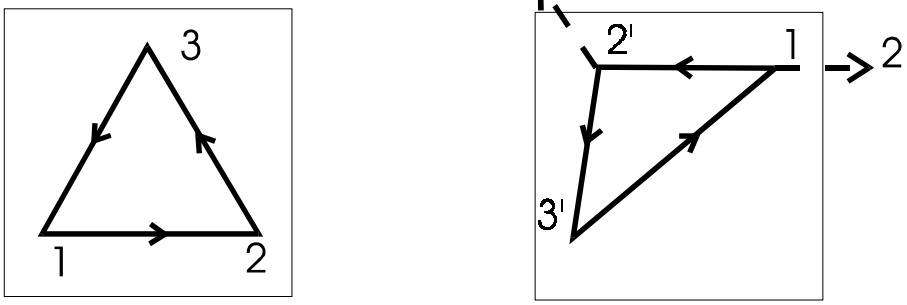
Análogo: janela possibilita visão do todo e de grande área em branco:



JANELA C/ CLIPPING

O exemplo acima demonstra a aplicação de escalas arbitrárias para mapeamento de um objeto dentro de uma janela.

Descrição de wraparound: o uso de escalas deferentes por programadores diferentes pode gerar problemas de um desenho aparecer recortado ao se mudar o dispositivo de saída pelo novo programador/operador. A falta de uso de escalas normalizadas pode gerar tal problema.

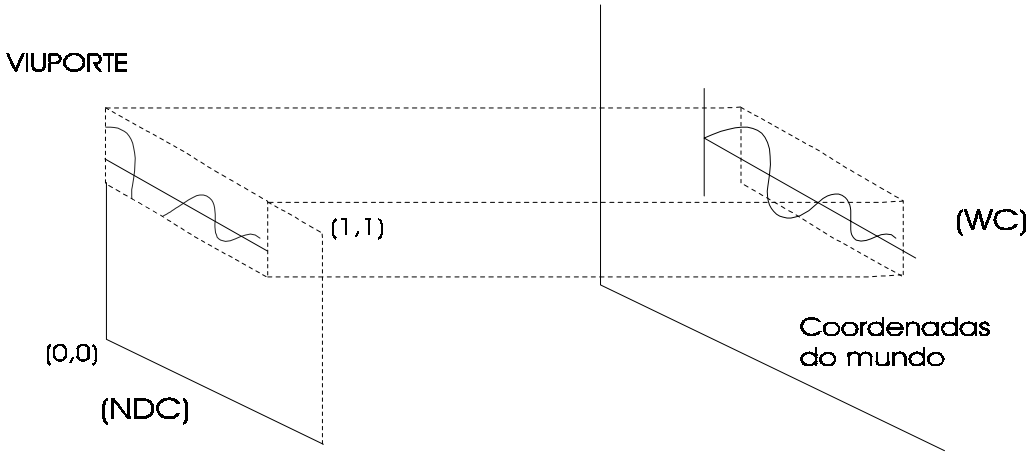


2.3. SISTEMAS DE COORDENADAS NORMALIZADAS

Para contornar o problema anterior, considerando a grande diversidade de equipamentos existentes, com diferentes resoluções gráficas, onde há um sistema de coordenadas para cada tipo de dispositivo, propõe-se o sistema de coordenadas normalizadas - NDC - (Normalized Device Coordinates) com valores de coordenadas variando de 0 a 1.

A porção retangular da tela sobre a qual a janela e todo seu conteúdo são mapeadas é chamada de:

VIUPORTE - VIEWPORT -



definição de uma viuporte:

VIEWPORT(min-x,max-x,min-y,max-y)

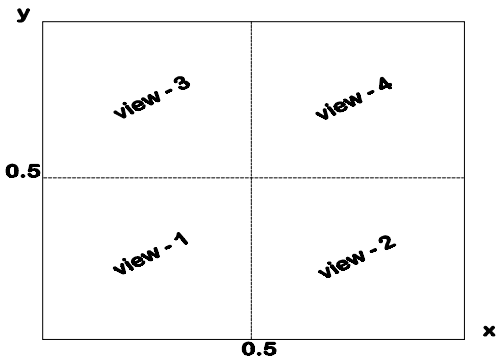
Numa situação em que definamos diferentes viuports conforme:

VIEWPORT-1 (0.0 , 0.5 , 0.0 , 0.5)

VIEWPORT-2 (0.5 , 1.0 , 0.0 , 0.5)

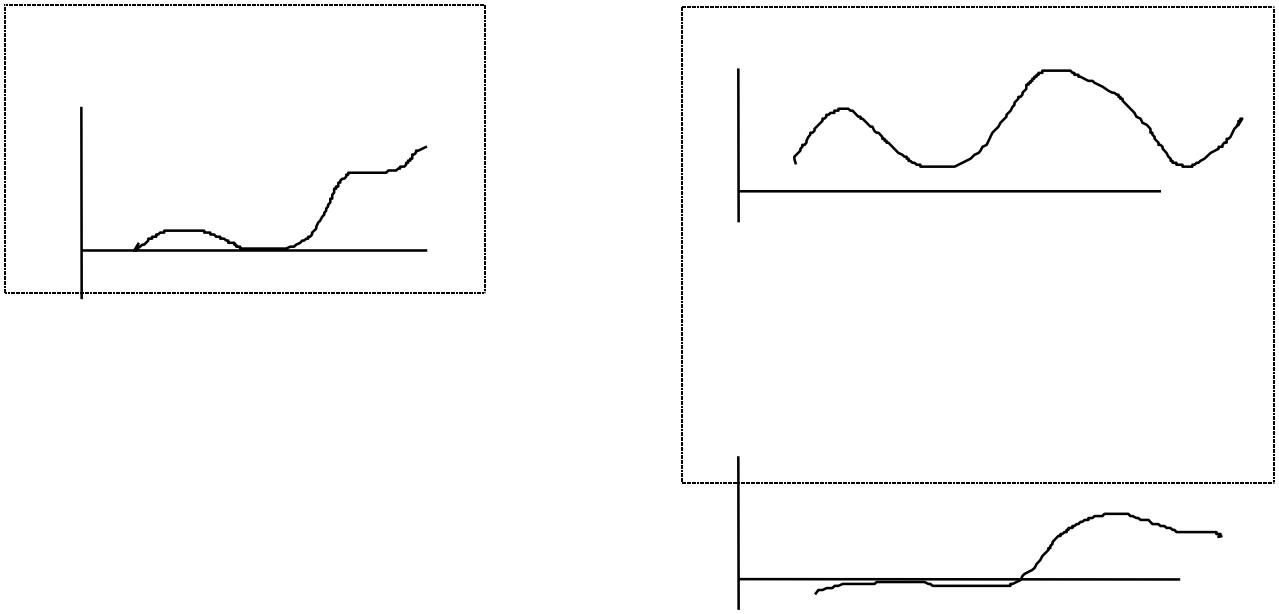
VIEWPORT-3 (0.0 , 0.5 , 0.5 , 1.0)

VIEWPORT-4 (0.5 , 1.0 , 0.5 , 1.0)

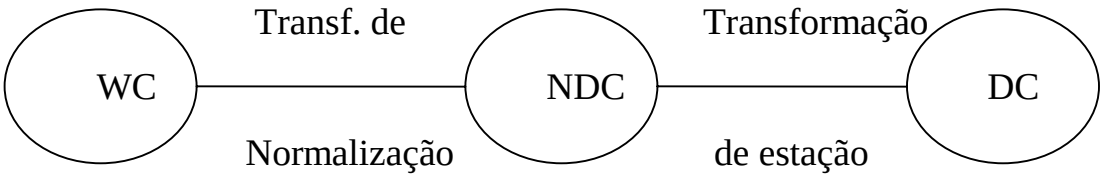


Obs.: Pode ser que o mapeamento para uma viuporte gere soluções com desfiguração de imagem caso os aspect ratios da window e viuport não correspondam,

isto é, as dimensões dos lados da janela não correspondem as dimensões de lados da viuporte. O mapeamento da lados das WC para as NDC produz a imagem com escalas diferentes.



"Janela e viuporte com diferentes aspectos aspect-ratios"



2.3.1. Transformação De Normalização

Tal mapeamento é descrito através de fórmulas que surgem da proporcionalidade, ou seja, de um posicionamento relativo de um ponto na janela deve refletir no ponto correspondente na viuporte.

Por interpolação, obtém-se:

$$\frac{XW - XWmin}{XWmax - XWmin} = \frac{XN - XNmin}{XNmax - XNmin} \Rightarrow$$

$$XN = \left(\frac{XW - XWmin}{XWmáx - XWmin}\right)*(XNmáx - XNmin) + XNmin$$

$$XN = \left(\frac{XNmáx - XNmin}{XWmáx - XWmin}\right)*(XW - XWmin) + XNmin$$

e finalmente;

$$XN = XNmin + fat\text{-}vis\text{-}x\ (*)\ (XW - XWmin)$$

ou

$$XN = Sx.(XW - XWmin) + XNmin$$

Sx = fator de escala window/viewport

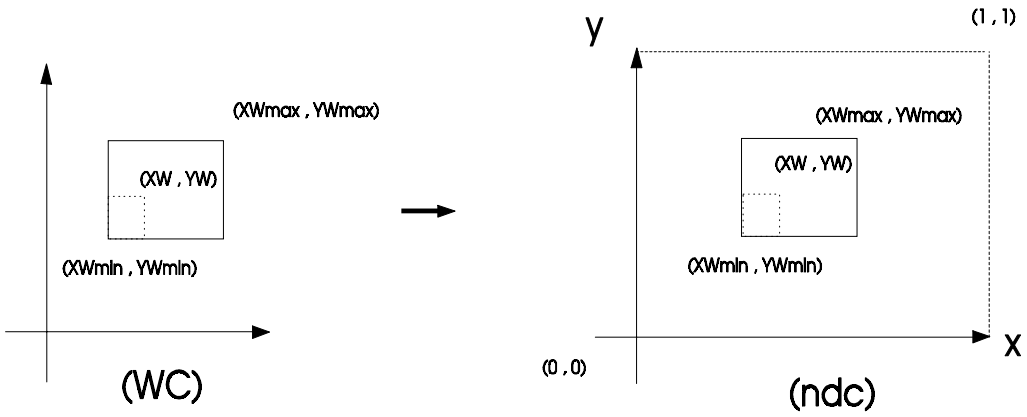
Analogamente:

$$YN = YNmin + fat\text{-}vis\text{-}y\ .\ (YW - YWmin)$$

sendo $fat\text{-}vis\text{-}y = \frac{YNmáx - YNmin}{}$

YNmax - YWmin

TRANSFORMAÇÃO (graficamente)



2.3.2. Transformação de estação (NDC-DC)

Conseguida com

map-ndc-dc(Xn,Yn,Xd,Yd):-

XD1 is Xn*NPX,

YD1 is Yn*NPV,

round(Xd1,0,XD2),

round(Yd1,0,YD2),

XD is integer(XD2),

YD is NPV - integer(YD2).

onde: round(X,N,Xz) => arredonda X com N casas decimais, devolvendo o resultado e Xz.

NPX = ndh - 1; NPY - ndv - 1

CGA - ndh - 640 ; ndv = 200 (baixa resolução)

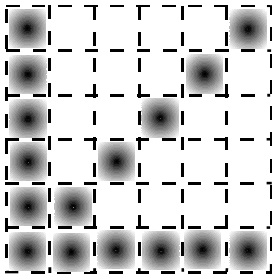
OPERAÇÕES DE VIEWING (Visualização)

3. GERAÇÃO DE PRIMITIVAS GRÁFICAS

3.1. Geração de Linhas

3.1.1. Método Analítico

Um TRC (Raster) pode ser considerado como uma matriz de pontos discretos, não é possível traçar diretamente uma linha unindo ponto a ponto. O processo de descrever da melhor maneira possível a linha encontrando as melhores aproximações é chamado de rasterização. Para linhas horizontais, verticais ou inclinadas 45º, o processo não acusa problemas.



Para outras linhas, será necessário decidir que pixel será usado (?)

Quando o dispositivo apresentar razoável

3.1.2. Analisador Diferencial Digital (DDA)

Técnica baseada na solução da equação diferencial. Para uma linha reta:

$\frac{dy}{} = K$ (constante) ou $\frac{\Delta y}{} = \frac{y2 - y1}{}$

$dx \qquad \Delta x \qquad x2 - x1$

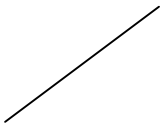
A solução é:

$$yi+1 = yi + \Delta y$$

$$yi+1 = yi + \frac{y2 - y1}{x2 - x1}.\Delta x$$

$\lvert x_2 - x_1 \rvert > \lvert y_2 - y_1 \rvert$	
Sim	Não
incremento = $y_2 - y_1 / x_2 - x_1$	incremento = $x_2 - x_1 / y_2 - y_1$
$y = y_1$	$x = x_1$
p/ x de x_1 até x_2 faça	p/ y de y_1 até y_2 faça
dot (x,y,cor)	dot (x,y,cor)
$y = y + \text{incremento}^{(*)}$	$x = x + \text{incremento}^{(*)}$

$^{(*)}$ determina a densidade da linha



(x_2, y_2)

y_i = valor para um do passo da linha.

(x_1, y_1)

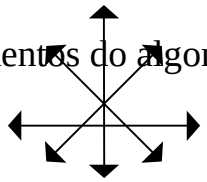
Problema: Truncamentos e arredondamentos.

Exercício: Apresentar que pontos serão usados para conceber a reta (0, 0) até (8, 4) utilizando o método acima.

3.1.3. Algoritmo de Bresenham

Justificativa: O método acima consome muito tempo com truncamentos. BRESENHAM propõe trabalho somente com inteiros. Não há uso de variáveis reais. Para simplificar o algoritmo, supomos o incremento como uma unidade e a inclinação da linha está entre 0 e 1. O algoritmo usa uma variável de decisão *di* que a cada passo é proporcional à diferença entre "s" e "t" mostrado na figura. Se s < t então *Si* está mais perto da linha desejada e será o escolhido; senão *Ti* esta mais perto e será o escolhido.

Movimentos do algoritmo de BRESENHAM

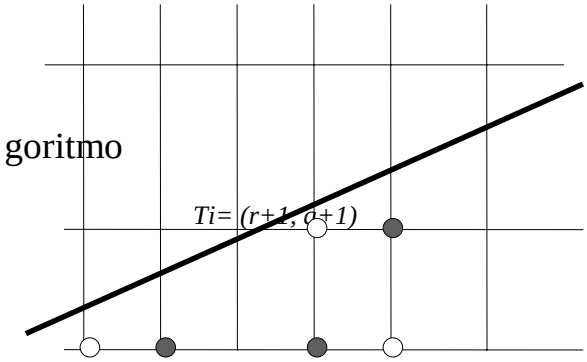


Incrementos unitários serão dados na direção da maior deslocamento. Assim, a inclinação da reta tem $0 \leq tg\delta \leq 1$. A cada incremento na direção maior, o valor da outra coordenada pode ser incremento ou não.

Para uma reta no primeiro octante: incremento em *y* deverá ser igual a 1 ou não haverá incremento, para cada incremento em x. Para decidir, observamos que se y_i ou $y_i + 1$ está mais próximo da posição da reta real.

3.1.3.1.Descrição Algorítmica

Demonstração



Pontos Negros: reta Selecionado pelo algoritmo desejada de BRESENHAM

$y = \frac{dy}{dx}.x$

–

Obs: a linha será desenhada de

➧ $Pi_{-1} = (r, q)$ (x_1, y_1) até (x_2, y_2).

$(= (xi_{-1}, yi_{-1}))$ A linha se transforma de (0, 0) até (dx, dy) onde

$d_x = x_2 - x_1$

$d_y = y_2 - y_1$ e a equação da reta é $y = \frac{dy}{dx}.x$

Nota-se que $y_s - q = s$; mas $y_s = \frac{dy}{dx}.(r+1) \Rightarrow s = \frac{dy}{dx}.(r+1) - q$

$t = (q + 1) - y_t$; mas $y_t = \left(\frac{dy}{dx}\right).(r+1) \Rightarrow t = (q+1) - \frac{dy}{dx}(r+1)$

logo $\Rightarrow (s - t) = 2.\frac{dy}{dx}(r+1) - 2q - 1$, dai $dx.(s - t) = 2.(r.dy - q.dx) + 2dy - dx$;

chamando $dx(s - t) = d_i \Rightarrow$

$d_i = 2.(rdy - qdx) + 2dy - dx$; mas $\begin{cases} r = xi - 1 \\ q = yi - 1 \end{cases} \Rightarrow$

$d_i = 2.(xi - 1)dy - 2.(yi - 1).dx + 2dy - dx$; (1), dai

$d_{i+1} = 2.xi.dy - 2yi.dx + 2dy - dx$; fazendo $d_{i+1} - d_i$, tem-se:

$\Rightarrow d_{i+1} - d_i = 2dy - 2dx(y_i - y_{i-1})$

$$d_{(i+1)} = d_i + 2dy - 2dx.(y_i - y_{i-1})$$

lenbrando que $d_i = dx.(s - t)$; $\begin{cases} \text{se } s \geq t \Rightarrow d_i \geq 0 - T_i \\ \text{se } s < t \Rightarrow d_i < 0 - S_i \end{cases}$

i) Se $d_i \geq 0 \Rightarrow$ tomamos T_i $(y_i = y_{i-1} + 1 = d_i + 2.(dy - dx)$

ii) Se $d_i < 0 \Rightarrow$ tomamos S_i $(y_i = d_i + 1 = d_i + 2.dy$

Para $(i = 1)$, como $(x_0, y_0) = (0,0) \Rightarrow$ $d_i = 2.dy - dx$, por 1

Obs.: O algoritmo acima funciona bem para linhas do primeiro octante. Como podemos generalizar que uma linha em outros octantes pode ser obtida à partir do primeiro é necessário apenas desenvolver transformações do primeiro para os demais octantes.

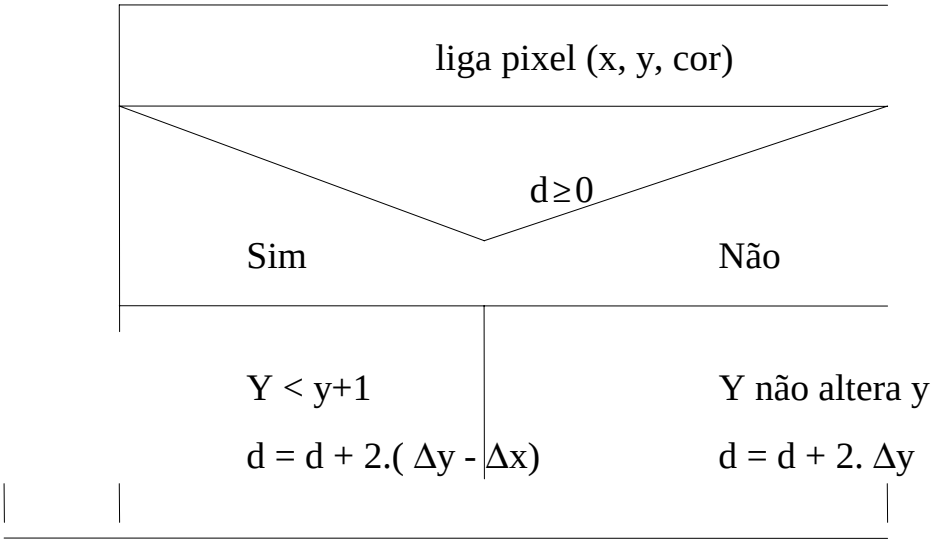
Descrição Algorítmica - quadro

considerando $(x_0, y_0) = (0, 0)$ $(i = 1)$:

$\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$

Delta = d = 2. $\Delta y - \Delta x$

para x de x_1 até x_2 faça:



3.1.3.2. Antialiasing

Como retas e arestas são contínuas, seu traçado em monitores do tipo "raster" causam certas imprecisões, porque tais dispositivos são discretos. O fenômeno é chamado de aliasing (serrilhado).

Soluções para antialiasing.

Aumentar a taxa de amostragem - com aumento da resolução do "RASTER"

Verificar quanto cada pixel se encontra dentro da linha a ser desenhada.

3.1.3.3.Algoritmo De Bresenham Modificado Com Antialiasing:

Obs: Linha $\Rightarrow$ (x_1,y_1) à (x_2,y_2)

I $\Rightarrow$ n° de intensidades disponíveis

$x = x_1$; $y = y_1$

$\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$

$m = I(\Delta y / \Delta x)$; $w = 1 - m$;

$e = I / 2$ (= 1 / 2 para intensidade I)

```
plot (x , y , m/2)
while (x < Δx)
        if  e < w them
                x = x + 1, e = e + m
        else
                x = x + 1 , y = y +1, e = e - m
        end if
        plot (x , y , e)
end while
finish
```

ALGORITMO GERAL DE BRESENHAM (utilização para todos os quadrantes)

-ver livro do Roger, pág. 40/41

3.2. GERAÇÃO DA CIRCUNFERÊNCIA

3.2.1. Equação paramétrica

A equação de circunferência (cônicas, de forma geral) enfrenta problemas similares geração de retas, adicionando-se a dificuldade de cálculo de funções trigonométricas. A primeira proposta de solução é dada pelo cálculo de pontos num período, usando as funções sen e cos:

$$0 \le t \le 2\pi \qquad \begin{cases} y = r \cdot \text{sen}(t) \\ x = r \cdot \cos(t) \end{cases}$$

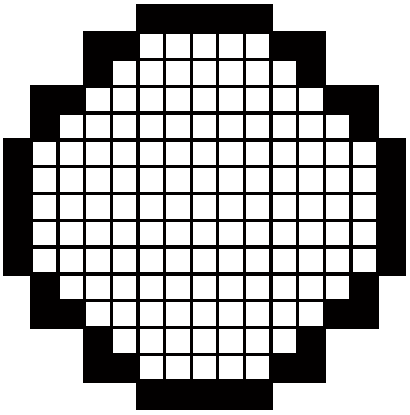
passo: t - mantém adequada a densidade dos pontos

$x \leq raio;$ y = 0

para t de 1 até 360 com passo “t”

pixel (x , y , cor)

$$x = r \cdot \cos(\tfrac{\pi \cdot t}{180})$$
$$y = r \cdot \text{sen}(\tfrac{\pi \cdot t}{180})$$

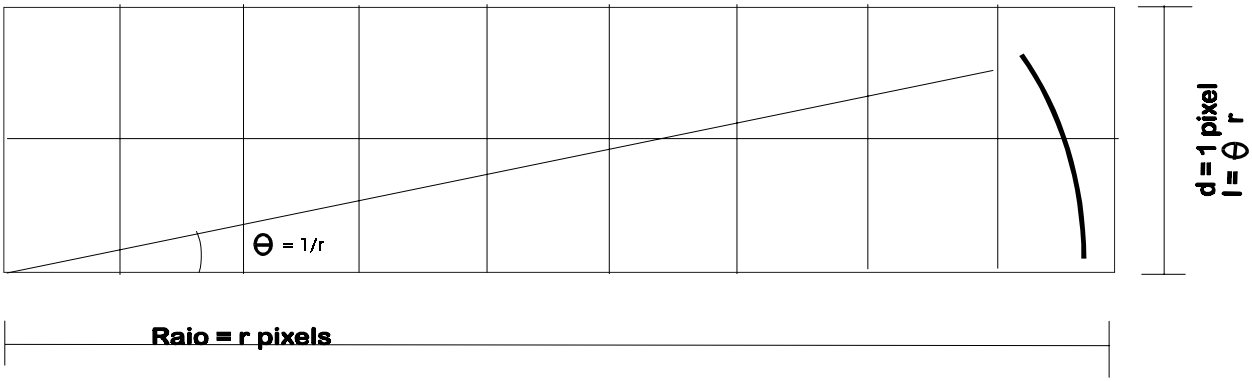


Problemas : - uso de funções trigonométricas.

- densidade dos pontos varia com o raio.

Algoritmo Incremental Com Simetria

base: deslocamento angular com incremento de 1 pixel (deslocamento em radianos - 1/r)



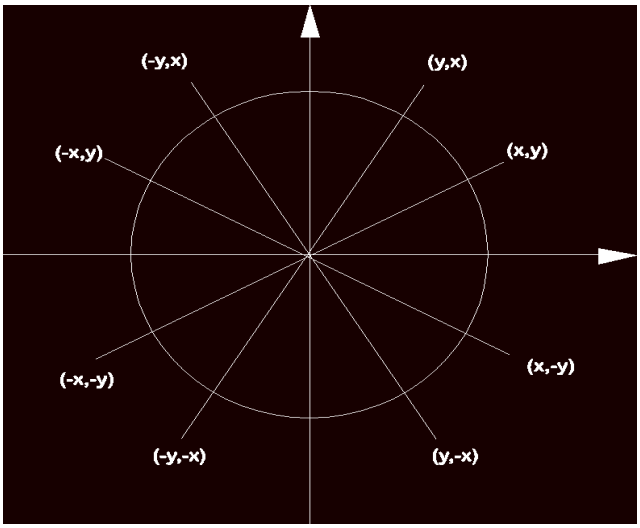
Uma propriedade importante que pode-se usar no algoritmo é a simetria da circunferência:

$$\cos(x) = \cos(\pi \pm x)$$

$$\cos(x) = \cos(-x)\ \pi$$

$$\text{sen}(x) = \text{sen}(\pi - x)$$

$$\text{sen}(x) = \text{sen}(\pi + x) = -\text{sen}(-x)$$



Problema: uso de funções trigonométricas o que reduz muito a eficiência do algoritmo.

Solução: algoritmo incremental com deslocamento angular constante e pequeno, com a rotação a partir de um ponto inicial:

$$\begin{cases} x_{n+1} = x_n \cdot \cos\theta + y_n \cdot \operatorname{sen}\theta \\ y_{n+1} = y_n \cdot \cos\theta - x_n \operatorname{sen}\theta \end{cases}$$

onde : cos θ e sen θ são valores fixos.

Problema: uso de xn e yn nas próximas iterações causa erros cumulativos

Solução: uso de números reais com função de arredondamento, calculando cada pixel.

3.2.1.1.ALGORITMO

$x \le r$; y = 0;

gera pontos sobre o eixo:

pixel (x , y , cor),

pixel (-x , y , cor),

pixel (x , -y , cor),

pixel (-x , -y , cor).

demais pontos $(0 \leq t \leq p / 4$ com passo $1 / r)$:

x = r cos t,

y = r sen t,

pixel (x , y , cor),

pixel (x , -y , cor),

pixel (-x , y , cor),

pixel (-x , -y , cor),

pixel (y , x , cor),

pixel (y , -x , cor),

pixel (-y , -x , cor).

3.2.2. Algoritmo de Bresenham

A solução dada por BRESENHAM também utiliza a noção de simetria, gerando o primeiro quadrante e os demais por simetria. Utiliza-se ainda um círculo centrado na origem. Começamos no ponto (O,R) e vamos diminuindo x exaustivamente, ou começamos no ponto (R,O) e diminuímos u exaustivamente.

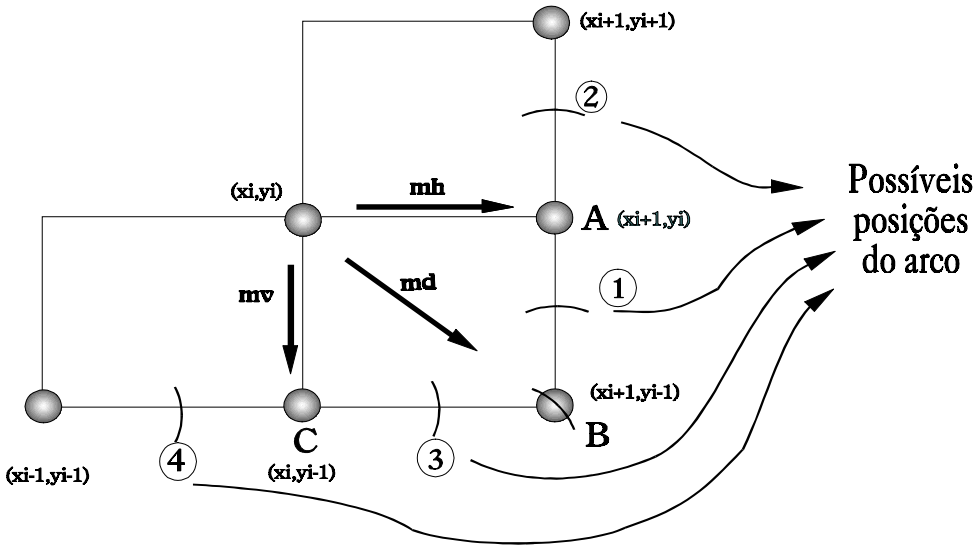
Escolha: (O,R).

Partindo de (O,R), gerando a circunferência no sentido horário, o algoritmo pode escolher entre 3 pontos diferentes, sendo que a escolha deve recair sobre aquele que está mais próximo do círculo ideal:

- horizontalmente para direita; (mh)

- diagonalmente para baixo à direita; (md)

- verticalmente para baixo; (mv)



$$\begin{cases} mh = \left| (x_i + 1)^2 + (y_i)^2 - R^2 \right| \\ md = \left| (x_i + 1)^2 + (y_i - 1)^2 - R^2 \right| \\ mv = \left| (x_i)^2 + (y_i - 1)^2 - R^2 \right| \end{cases}$$

O algoritmo escolhe o pixel que minimize o quadrado da distância entre um destes pixel e o círculo verdadeiro (mv, md, mh).

Chamando Δi = diferença entre o quadrado da distância do pixel ao centro e o raio da circunferência, temos: $\Delta i = (x_i + 1)^2 + (y_i - 1)^2 + R^2$ (pixel diagonal)

1°) caso ($\Delta i < 0$) - O ponto B está no interior do círculo e deve-se escolher outro ponto: caso 1 ou 2.

Se $\delta = mh - md$ e $\delta = md - mv$ e

a) se $\delta \leq 0 \Rightarrow$ o ponto escolhido é o A;

b) se $\delta > 0 \Rightarrow$ o ponto escolhido é o B;

2°) caso ($\Delta i < 0$) - O ponto B está fora da circunferência (caso 3 ou 4)

a) se $\delta \leq 0 \Rightarrow$ o ponto escolhido é o B;

b) se $\delta > 0 \Rightarrow$ o ponto escolhido é o C;

4. PREENCHIMENTO DE ÁREAS

Terminais de varredura têm como vantagem (sobre terminais vetoriais) a capacidade de representação de áreas preenchidas (cheias). Há necessidade de um algoritmo para preenchimento de áreas do tipo polígono de circunferências.

4.1. Algoritmo de Varredura

Princípio: o contorno do polígono já está desenhado na tela com uma determinada cor, diferente daquela de fundo.

- necessita de um ponto interno deste polígono.

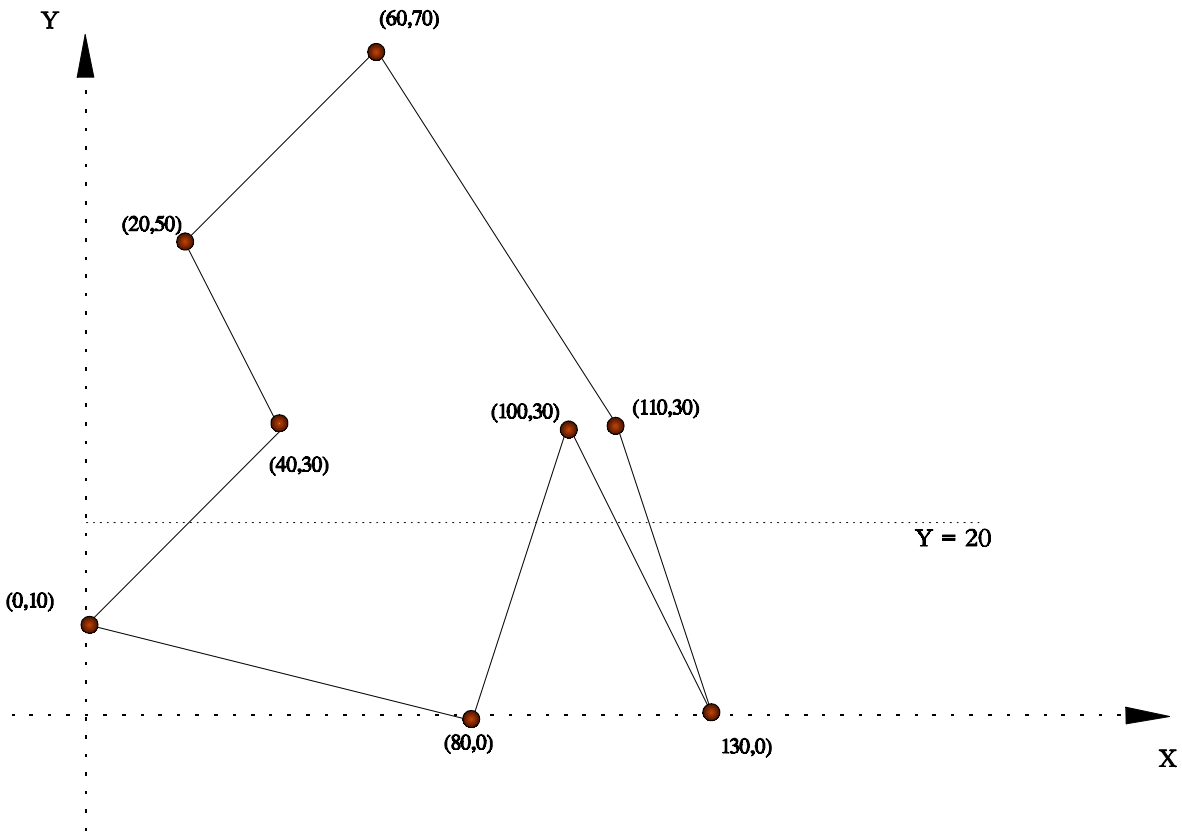
Descrição: A partir do ponto fornecido, muda a cor do pixel até encontrar um pixel de cor diferente da cor de fundo, seguindo as linhas de varredura; (encontrar uma aresta do polígono).

-Uma lista ligada armazena pontos que servem para continuar o algoritmo; tais pontos fazem o papel de ponto inicial, na iteração seguinte.

Obs.: o algoritmo se presta à preenchimento de qualquer área fechada.

4.2.Algoritmo de Análise Geométrica

Baseado na descrição geométrica (como, por exemplo, uma lista de vértices). Identifica os pontos externos do polígono e as interseções das arestas do polígono com as linhas de varredura.



1° passo: montar a tabela de lados:

LADO	Y_{min}	$Y_{máx}$	X para Y_{min}	1/m
1	0	10	80	-8,0
2	10	30	0	+2,0
3	30	50	40	-1,0
4	50	70	20	+2,0
5	30	70	110	-1,25
6	0	30	130	-0,67
7	0	30	130	-1,0
8	10	30	80	+0,67

(Armazenamento do polígono - 8 lados)

2° passo: Interseção com a linha de varredura

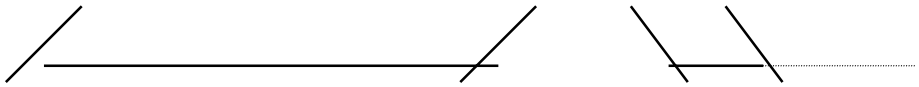
Vamos excluir os lados que estão acima da linha de varredura (y=20) $\Rightarrow$ ($Y_{min} \geq 20$) e os que estão abaixo da mesma ($Y_{máx} \leq 20$). Assim a análise será das arestas 2, 6, 7 e 8.

A interseção é dada pela fórmula:

$$x = \frac{1}{m} \cdot (Y_{varredura} - Y_{min}) + X_{Ymin} \quad \text{(Equação da reta)}$$

Observe: (aresta 7) x = -1.0 (20 - 0) + 130 = 110

3° passo: Ordenam-se os pontos e traça-se as linhas, tomando-as de 2 em 2. Ordenamento de x, com valores crescentes:



y=20

(x=20) (x=116,6) (x=110) (x=86,7)

(1) (2) (3) (4)

O preenchimento começa para x par e termina para x impar.

Problema: linha de varredura (y = 30) encontra um vértice

- A primeira interseção no ponto (40 , 30) $\Rightarrow$ 1 único ponto para mudar o estado de preenchimento.

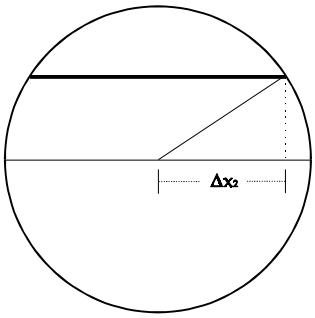
- A interseção (100 , 30) não deve mudar este estado, pois, o preenchimento deve continuar.

Solução: (100,30) deve ser representado na lista de interseções por um número par de pontos (dois ou mesmo nenhum), neste caso, os lados estão totalmente acima ou abaixo do vértice, as inclinações têm sinais diferentes.

4.3.Preenchimento De Circunferência

Quando o problema é uma circunferência, o preenchimento é simples: calcula-se facilmente a interseção dela com a linha de varredura:

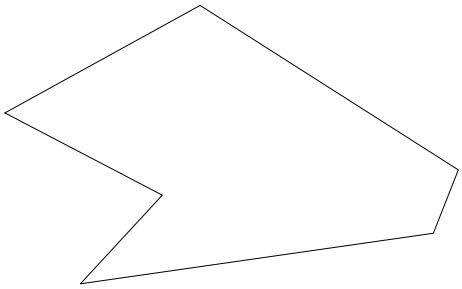
x (cálculo) = $R^2 - y^2$



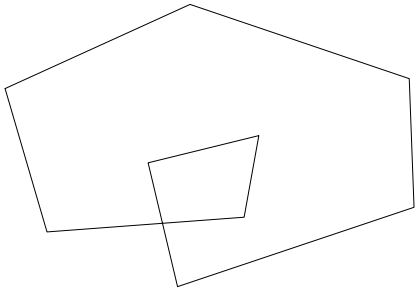
EXERCÍCIOS:

1) Preencher as seguintes áreas

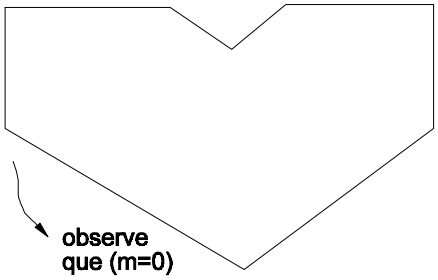
a)



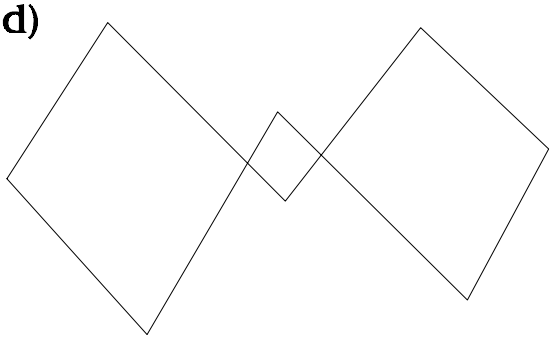
b)



c)

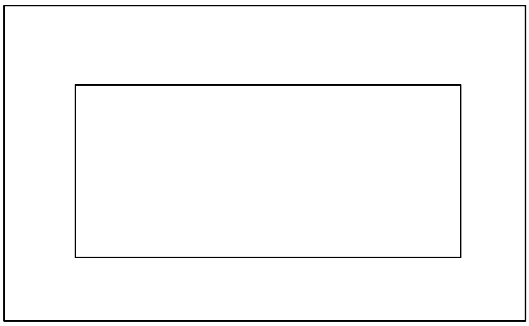


d)

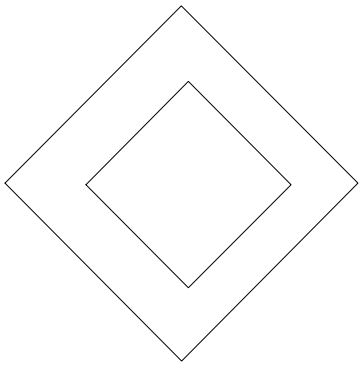


2) Estenda o algoritmo, de maneira a cobrir as seguintes áreas:

a)



b)



5. TRANSFORMAÇÕES GEOMÉTRICAS

São (alterações) operações matemáticas que permitem alterar uniformemente o aspecto de um desenho já armazenado no computador. Tais transformações permitem alterações uniformes de uma imagem definida sobre um sistema de coordenadas. Não há comprometimento da estrutura do desenho mas do aspecto que o mesmo assumirá. (mudança de orientação / escala).

Observação: é interessante notar que as operações de transformação de visualização (WC/NDC) São combinações de transformações de escala (altera valores das coordenadas de modo proporcional) e de translação.

3 tipos fundamentais
de transformação :

ESCALA
TRANSLAÇÃO
ROTAÇÃO (ao redor da origem)

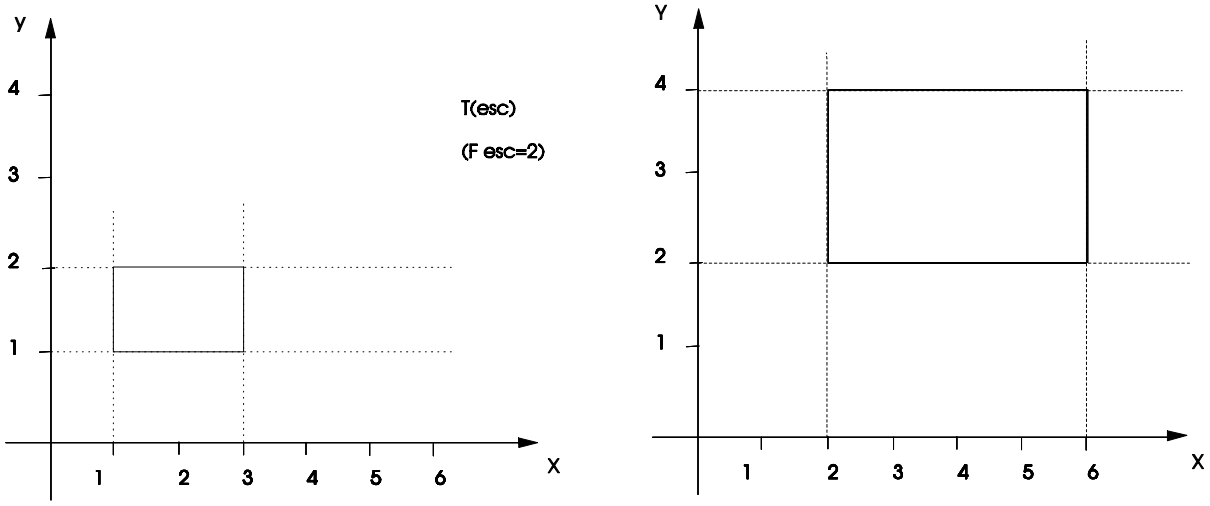
5.1. Transformação De Escala

Multiplicação de todas as coordenadas que definem o desenho por fatores de escala não nulos.

$$\text{BIDIMENSIONAL} \begin{cases} \bullet \text{ Fator de escala horizontal - X} \\ \bullet \text{ Fator de escala vertical - Y} \end{cases}$$

$$\begin{cases} x' = Ex \cdot x \\ y' = Ey \cdot y \end{cases} \qquad \text{ou} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Ex & 0 \\ 0 & Ey \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\text{obs:} \begin{cases} E > 1 \Rightarrow \text{Ampliacao da imagem} \\ 0 < E < 1 \Rightarrow \text{reducao da imagem} \\ E < 0 \Rightarrow \text{Espelhamento} \end{cases}$$



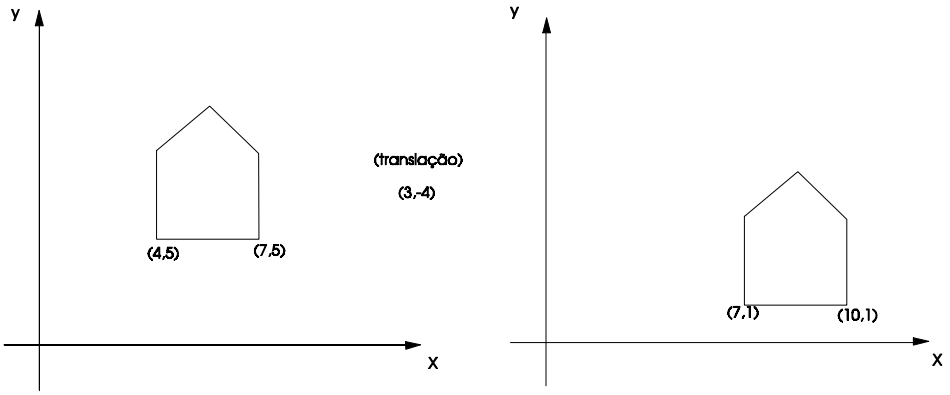
Obs: fatores Ex e Ey iguais $\Rightarrow$ semelhança com o original

II. Translação: movimentação da figura para outra posição no sistema de coordenadas. Todos os pontos da imagem são deslocados de uma mesma distância em relação a posição anterior.

$$\begin{cases} x' = x + Tx \\ y' = y + Ty \end{cases} \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Tx \\ Ty \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

P (x, y) ; P'(x', y') { P'= T+P }

obs.: suponha uma linha constituída por um número muito grande de pontos, este processo pode consumir muito tempo. A frente, definiremos translação com somente pontos iniciais e finais de linha.



5.2. Rotação Em Torno Da Origem:

Movimentação de uma figura para uma outra posição, de forma que todos os pontos da imagem mantenham a mesma distância da origem que possuíam

antes da transformação. Rotação em torno de um ponto qualquer: transformações de translação e de rotação em torno da origem.

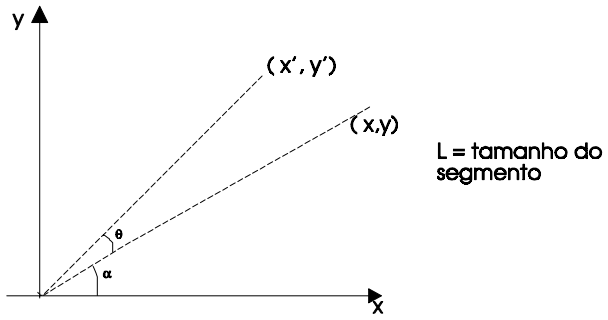
5.2.1. DEFINIÇÕES MATEMÁTICAS

$$\begin{cases} x' = x \cdot \cos\ \theta - y \cdot \operatorname{sen}\theta & \text{(I)} \\ y' = y \cdot \cos\theta + x \cdot \operatorname{sen}\theta & \text{(II)} \end{cases}$$

θ : ângulo tomado no sentido anti-horário

em outro sentido : $\begin{cases} \cos(-\theta) = \cos\theta \\ \operatorname{sen}(-\theta) = -\operatorname{sen}\theta \end{cases}$ $\Rightarrow$ Modificar equação I e II

Demonstração:



$L = \sqrt{x^2 + y^2}$, $\cos\alpha = \frac{x}{L}$; $\operatorname{sen}\alpha = \frac{y}{L}$;

mas L é a distância de (x', y') à origem também, temos:

$L = \sqrt{x^2 + y^2}$ e $\cos(\theta + \alpha = \frac{x'}{L}$; $\operatorname{sen}(\theta + \alpha) = \frac{y'}{L}$

como $\begin{cases}\cos(a+b)=\cos a\cdot\cos b-\text{sen}a\cdot\text{sen}b\\ \text{sen}(a+b)=\text{sen}a\cdot\cos b+\text{sen}b\cdot\cos a\end{cases}$, temos:

$$\begin{cases}\dfrac{x'}{L}=\cos\theta\cdot\cos\alpha-\text{sen}\,\theta\cdot\text{sen}\,\alpha\\ \dfrac{y'}{L}=\text{sen}\,\theta\cdot\cos\alpha+\text{sen}\,\alpha\cdot\cos\theta\end{cases}\quad\therefore\quad\begin{aligned}x'&=L\cdot\cos\theta\cdot\cos\alpha-L\cdot\text{sen}\,\theta\cdot\text{sen}\,\alpha\\ y'&=L\cdot\text{sen}\,\theta\cdot\cos\alpha+L\cdot\text{sen}\,\alpha\cdot\cos\theta\end{aligned}$$

5.3. Coordenadas Homogêneas e Matriz De Transformação

Podemos facilitar o cômputo de valores e transformar com uso de matrizes para rotação, escala e translação.

$$\begin{aligned}P'&=T+P &\Rightarrow\ &\text{Translacao}\\ \text{P}'&=\text{S}\cdot\text{P} &\Rightarrow\ &\text{Escala}\\ \text{P}'&=\text{R}\cdot\text{P} &\Rightarrow\ &\text{Rotacao}\end{aligned}$$

Se os pontos são tratados com coordenadas homogêneas, todos as transformações podem ser tratadas com multiplicações. Em coordenadas homogêneas, um ponto é tratado com um vetor de 3 valores (tripla) ⇒ (x, y, w). O ponto transformado é representado por (x', y', w'). W é colocado para dar consistência nos cálculos.

A. Matriz de Escala

$$E = \begin{bmatrix} Ex & 0 & 0 \\ 0 & Ey & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} 6 & 4 & w \end{bmatrix}$$

B. Matriz de Translação: no caso P'= T·P

inverter

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix} \begin{bmatrix} x & y & 1 \end{bmatrix}$$

Exemplo:

$$\begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} 3 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 \end{bmatrix}$$

ignorado ↵

C) **MATRIZ DE ROTAÇÃO**

$$\begin{cases} x' = x \cdot \cos\theta - y \operatorname{sen}\theta \\ y' = y \cdot \cos\theta + x \cdot \operatorname{sen}\theta \end{cases} \Rightarrow$$

$$\text{matriz de rotacao} = \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ -\operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R$$

$$\begin{bmatrix} x' & y' & w \end{bmatrix} = \begin{bmatrix} x & y & w \end{bmatrix} \cdot R \quad \text{ou no caso: } P' = R(\theta) \,.\, P$$

$$R = \begin{bmatrix} \cos\theta & -\text{sen}\theta & 0 \\ \text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.4. Combinação Das Transformações

Combinação de várias transformações $\Rightarrow$ espelhamento

Facilidade: uso de matrizes de transformação

Supondo:

$$(x, y) \xrightarrow{\text{transf. com uso de } M_1} (x_1, y_1) \xrightarrow{\text{transf. com uso de } M_2} (x_2, y_2)$$

existe M_3 que chega direto à (x_2, y_2)

Existe a matriz que faz diretamente a conversão e tal matriz é dada por m1.m2.

demonstração:

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix}.m_1 \qquad (1)$$

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} x_2 & y_1 & 1 \end{bmatrix}.m_2 \qquad (2)$$

(2) em (1) :

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} = \left(\begin{bmatrix} x & y & 1 \end{bmatrix}, m_1 \right).m_2$$, pela propriedade associativa :

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix}.m_3$$

Obs. : O produto de matrizes não é comutativo:

$m_1.m_2 \neq m_2.m_1$ Assim , a ordem na qual as transformações são aplicadas influi no resultado final. A vantagem do uso de uma única matriz é um ganho de eficiência.

Considerando, por exemplo, a rotação de um objeto sobre um ponto qualquer P_1. Transformamos este processo em 3 outros:

- transladamos de P_1 para a origem;
- rotacionamos;
- transladamos da origem para P_1.

A transformação seria dada por $\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix}.T_T$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix}.\begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ -\operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} = T_T$$

$$T_T = \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ -\operatorname{sen}\theta & \cos\theta & 0 \\ x_1(1-\cos\theta) + y_1.\operatorname{sen}\theta & y_1(1-\cos\theta) - x_1\operatorname{sen}\theta & 1 \end{bmatrix}$$

Obs. : A composição das transformações pela matriz de multiplicação ilustra a facilidade proporcionada pelo uso de coordenadas homogêneas.

Caso 2: Multiplicação por escala, seguida de translação

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} . \begin{bmatrix} Ex & 0 & 0 \\ 0 & Ey & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} = T_{Tz}$$

$$T_{Tz} = \begin{bmatrix} Ex & 0 & 0 \\ 0 & Ey & 0 \\ x_1(1-Ex) & y_1(1-Ey) & 1 \end{bmatrix}$$

Seqüência:

- translada-se o ponto de referência para a origem;
- promove-se a rotação em torno da origem;
- translada-se da origem para o ponto de referência.

5.5. Espelhamento:

Produção de imagens simétricas com uso de transformações de escala com fatores negativos

Espelhamento
- c/ relação a **x** => Ex = 1 ; Ey = -1
- c/ relação a **y** => Ex = -1 ; Ey = 1
- c/ relação a **x** e **y** => Ex = -1 ; Ey = -1

5.6. Espelhamento em relação a uma reta qualquer:

a) Traslada-se a figura de modo que um dos pontos da reta de simetria vá para a origem;

b) Rotaciona-se a figura até que a reta de simetria se torne paralela à um dos eixos do sistema de coordenadas;

c) Espelha-se a figura em relação ao eixo que, neste instante, coincide com a reta de simetria. Caso, seja o eixo Y, usa-se Ex = -1; Ey - 1;

d) Rotaciona-se a figura em um ângulo oposto ao aplicado em **b** -de forma a retornar a reta de simetria à sua posição original- ;

e) Transladar a figura com constantes de deslocamento opostas às aplicadas em **a** , de modo a voltar a reta de simetria à sua posição .

5.7. Considerações sobre eficiência:

Uma combinação de matrizes de rotação (R), escala (S) e transformação (T) pode produzir uma matriz de forma:

$$\begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix}$$ sendo, a obtenção de $(x_1 \;\; y_1)$ da por:

$$\begin{bmatrix} x_1 & y_1 & z \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} a & b & 0 \\ d & e & 0 \\ c & f & 1 \end{bmatrix}$$

chegando a $\begin{cases} x_1 = ax + by + c \\ y_1 = dx + ey + f \end{cases}$, (9 multiplicações e 6 adições)

Podemos exigir menos operações aritméticas com uso de

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{(3X2)}$$ => (menos operações)

5.8. TRANSFORMAÇÕES EM 3 DIMENSÕES:

A idéia básica consiste em trabalhar com algoritmos e estrutura de dados que representem tridimensionalmente a imagem com conversão de coordenadas tridimensionais para projeção bidimensionais, no memento da representação de imagem.

- Ponto => representado por 3 coordenadas (x,y,z) e por 4 coordenadas no caso de coordenadas homogêneas: Matrizes 4x4

Representação matricial : (W_x , W_y , W_z , W) ; $W \neq 0$. Se $W \neq 1$ => W é dividido nas 3 primeiras coordenadas homogêneas para obter a coordenada cartesiana do ponto (x,y,z) .

5.8.1. Escala:

multiplicação de cada uma das coordenadas pelo fator de escala correspondente. Efeito: aproximação ou afastamento do ponto em relação à origem do sistema, proporcionalmente, em cada eixo, aos fatores de escala;

$$\text{Matriz: } \begin{bmatrix} Ex & 0 & 0 & 0 \\ 0 & Ey & 0 & 0 \\ 0 & 0 & Ez & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ e, portanto:}$$

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} . \begin{bmatrix} Ex & 0 & 0 & 0 \\ 0 & Ey & 0 & 0 \\ 0 & 0 & Ez & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.8.2. Translação:

A seguir a matriz homogênea representa a translação em 3 dimensões:

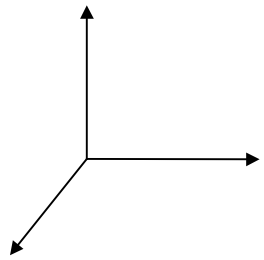
$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$ onde T_x , T_y , T_z representam as constantes de deslocamento em cada eixo. $\begin{bmatrix} x' & y' & z' & w \end{bmatrix} = \begin{bmatrix} x & y & z & w \end{bmatrix}.T_3$

5.8.3. Rotação:

Quando trabalha-se com 3 dimensões deve-se definir qual o plano sobre o qual a rotação deve ser realizada, ou seja, qual o eixo em torno de qual se procederá a rotação.

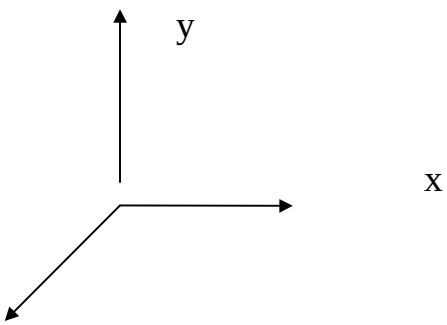
Se: plano de rotação por eixo **x** e **y,** seja eixo de rotação z:

$$R_z = \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



A rotação em torno de eixo **x** seria:

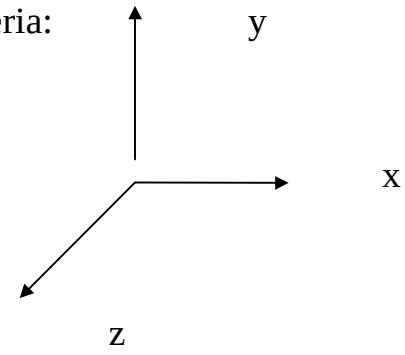
$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \text{sen}\,\alpha & 0 \\ 0 & -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



z

E, finalmente, em torno do eixo y seria:

$$Ry = \begin{bmatrix} \cos\theta & 0 & -\operatorname{sen}\theta & 0 \\ 0 & 1 & 0 & 1 \\ \operatorname{sen}\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



5.8.4. Rotação em torno de um eixo arbitrário**:**

Sejam (a, b, c) e (a', b', c') os pontos que determinam os eixos:

PASSOS:

1) Translação de forma a fazer o eixo de rotação passar pela origem (T_x = -a; Ty = -b; T_z = -c);

2) Rotação em torno de eixo **x**, de forma que o eixo de rotação fique no plano xz; (angulo θ);

3) Nova rotação agora em torno de eixo y (angulo β) até que o eixo de rotação coincida com o eixo **z**;

4) Rotação em torno de eixo **z** com o ângulo desejado (α) ;

5) Rotação de - β em torno do eixo y;

6) Rotação de $-\theta$ em torno do eixo **x**;

7) Translação com (T_x = a; T_y = b; Tz = c). inversa

Resumo

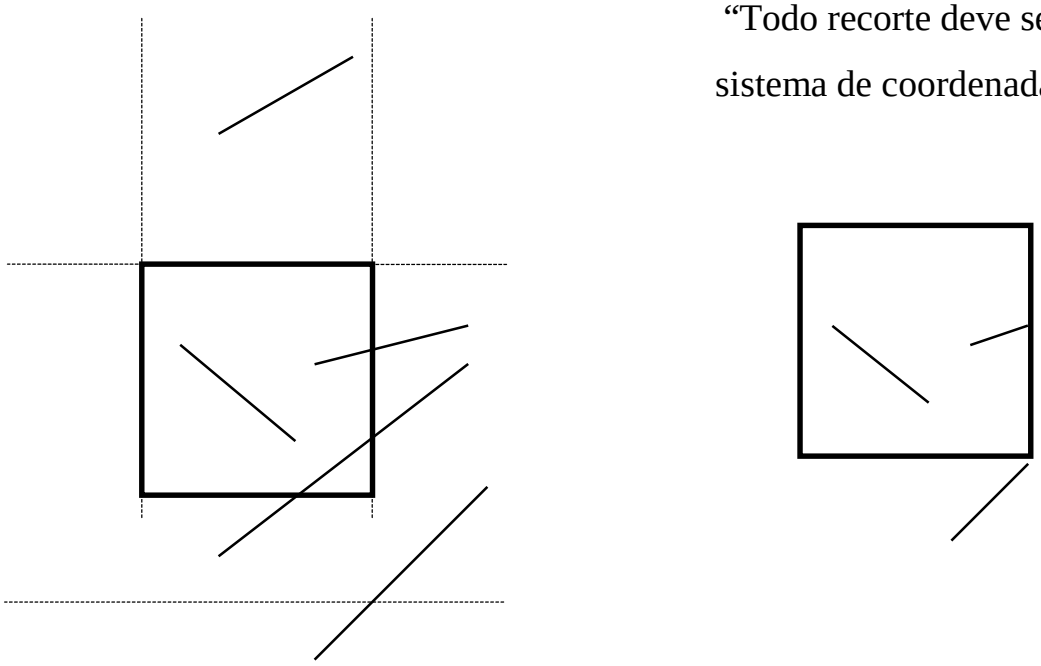
$$\text{Novo\_Ponto} = \text{Ponto\_velho}.\ T.R_x.R_y.R_\alpha.R_y^{-1}.R_x^{-1}.T^{-1}$$

Exercício para implementar:

Apresente as transformações simples aplicadas à um triângulo qualquer definido nos eixos x, y, z.

Apresente um esquema no qual pelo menos 2 transformações foram aplicadas à uma figura qualquer.

6. ALGORITMO DE RECORTE (CLIPPING):



"Todo recorte deve ser efetuado no sistema de coordenadas de usuário"

JANELA VIEWPORT

fig. 1: Recorte de pontos, linhas, caracteres

6.1. Pontos:

Processo rápido e muito simples. O ponto que deve ser apresentado na viewport é aquele para o qual as inequações abaixo são satisfeitas.

$x_{min} \leq x \leq x_{max}$ e $y_{min} \leq y \leq y_{max}$

O ponto que não satisfaz as quatro inequações acima, ou uma delas qualquer não deve ser apresentado.

6.2. Linhas:

Exige mais cálculos e testes do que o processo anterior. Embora seja necessário considerar apenas as partes finais da linha e não uma infinita quantidade de pontos.

No desenho acima a linha EF, Cujos pontos finais se encontram dentro dos limites da janela é aceita trivialmente.

Linhas GH e IJ => dever ser cortadas.

6.2.1. 1ª solução:

Checagem por meio da equação paramétrica envolvendo os limites da janela e a própria linha.

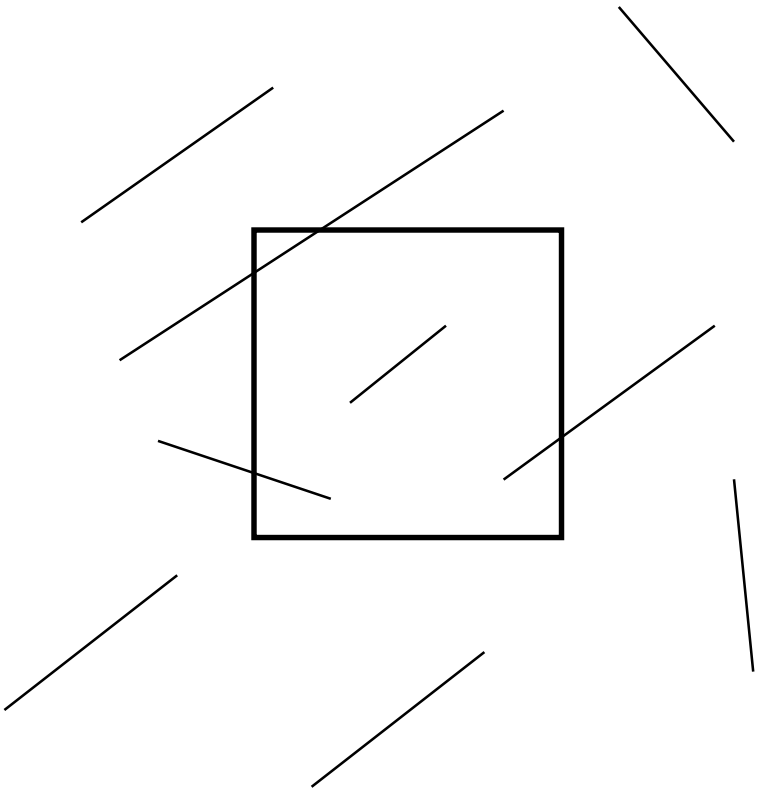
Consiste num processo de grande quantidade de cálculos e teste e não é muito eficiente. Num display típico, centenas ou milhares de linhas de linha podem ser encontradas.

Um algoritmo eficiente deve promover alguns testes iniciais na linha de forma a determinar se cálculos de interseção são realmente necessários . Inicialmente, o par de pontos finais pode ser checado para observar se ambos pertencem à janela, fazendo a linha ser aceita trivialmente. Ao mesmo tempo, a linha pode ser trivialmente rejeitada, por testes simples também, como no caso da linha CD, que apre-

senta $y>y_{max}$(da janela). Podem ser por exemplo, rejeitados pontos que estão abaixo de y_{min} e à esquerda de x_{min} ou à direita de x_{max}.

6.2.2. Algoritmo de Cohen-Sutherland:

Projetado para identificar eficientemente que linhas podem ser trivialmente aceitas ou rejeitadas checando regiões. Cálculos de interseção são necessários apenas para linhas na qual os casos acima falharem.



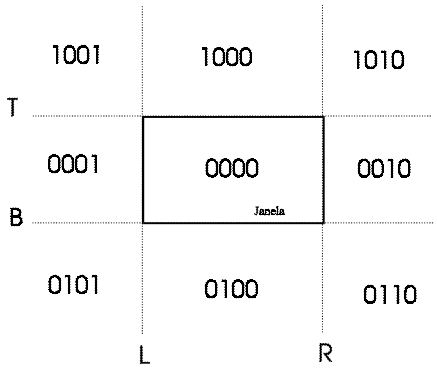
1° bit - o ponto está à esquerda da janela - b0 - P(4)

2° bit - o ponto está à direita da janela - b1 - P(3)

3° bit - o ponto está à abaixo da janela - b2 - P(2)

4° bit - o ponto está à acima da janela - b3 - P(1)

(b3 b2 b1 b0)



1° passo: Associar códigos aos pontos finais usando a regra:

$$\begin{cases} \text{se } x_1 < X_L \Rightarrow P_{code(4)} = 1; \text{ do contrario } P_{code(4)} = 0 \\ \text{se } x_1 > X_L \Rightarrow P_{code(3)} = 1; \text{ do contrario } P_{code(3)} = 0 \end{cases}$$

$$\begin{cases} \text{se } y_1 < Y_T \Rightarrow P_{code(1)} = 1; \text{ do contrario } P_{code(1)} = 0 \\ \text{se } y_1 > Y_B \Rightarrow P_{code(2)} = 1; \text{ do contrario } P_{code(2)} = 0 \end{cases}$$

2° verificar se é totalmente visível:

- Se os dois códigos associados as duas extremidades do segmento de reta forem zero ⇒ a linha é totalmente visível.

$soma_1 := 0$; $soma_2 := 0$;
for i = 1 to 4
    $soma_1 := soma_1 + P1CODE(i)$
    $soma_2 := soma_2 + P2CODE(i)$
if $soma_1 = 0$ and $soma_2 = 0$ them
linha totalmente visível

3° passo: verificar se a linha é invisível, se encontrando totalmente à direita, ou à esquerda, ou acima ou abaixo.

Inter : = 0
for i = 1 to 4
    Inter : = Inter + integer ((P1. CODE(i) + P2. CODE(i)) / 2)
    If Inter <> 0 then linha invisível
                else next i

Ex:
- linha KL P1 (0 0 1 0) ; P2(0 1 1 0) ⇒ OK - linha invisivel
- linha GH P1 (0 0 0 1) ; P2 (1 0 0 0) ⇒ Falha
- linha IJ_1 P1 (1 0 0 1) ; P2 (1 0 0 0) ⇒ OK - linha invisivel

4° passo: Verificar se um dos pontos está dentro da janela.

$Soma_1 = 0$ ou $Soma_2 = 0$

Se estiver, basta tomar uma intersecção

$$Xc \leq Xc' \leq Xd$$
$$Xe \leq Xf' \leq Xf$$

5° passo: A linha é parcialmente visível ou totalmente invisível para o primeiro passo, devem ser calculadas as interseções.

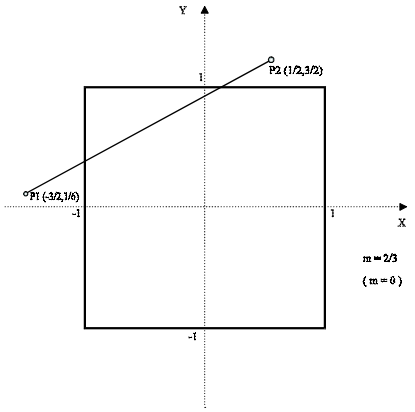
Esquerda: X_L, $Y = M^* (X_L - X_1) + Y_1$; $M \neq 0$;

Direita: X_R, $Y = M^* (X_R - X_1) + Y_1$; $M \neq 0$;

Top: Y_T, $X = X_1 + 1/M \,.\, (Y_T - Y_1)$; $M \neq 0$;

Botton: Y_B, $X = X_1 + 1/M \,.\, (Y_B - Y_1)$; $M \neq 0$;

Exemplo:



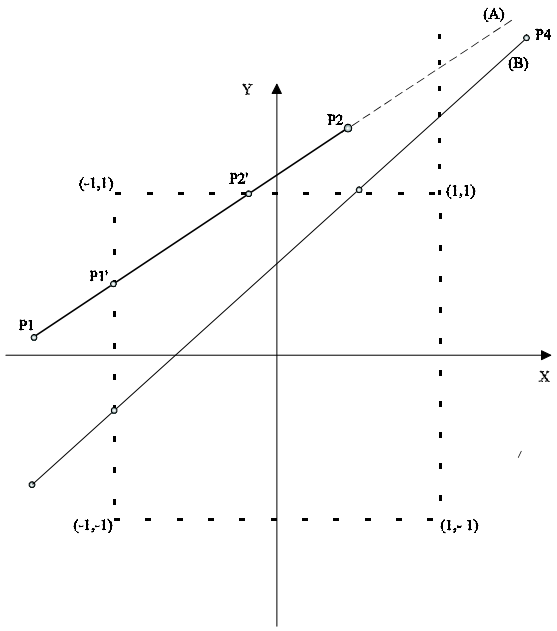
Esquerda: $X= -1 \Rightarrow Y = 2/3. [-1 - (-3/2)] + 1/6 = ½ \{Y_B \leq Y = ½ \leq Y_T\}$

Direita: $X= 1 \Rightarrow Y = 2/3. [1 - (-3/2)] + 1/6 = 11/6$ {FORA}

Top: $Y= 1 \Rightarrow X = -3/2 + 3/2. [1 - 1/6] = -1/4 \{X_L \leq -1/4 \leq X_R\}$

Botton: $Y= -1 \Rightarrow X = -3/2 + 3/2. [1 - 1/6] = -13/4$ {FORA}

Exemplo: Prove a eficiência do algoritmo anterior para os casos A e B abaixo:



1/6)

P1 (-3/2 ,

P2 (1/2 , 3/2)

P_1' (-1 , 1/2)

6.2.3. ALGORITMO DA SUBDIVISÃO DO PONTO MÉDIO

-MIDPOINT SUBDIVISION ALGORITHM-

O algoritmo anterior necessitava de cálculo para obter a interseção da linha com os limites da janela. O cálculo direto pode ser evitado com utilização de busca binária, dividindo a linha em pontos médios. Este algoritmo, que é um caso particular do anterior foi proposto por Sproull e Sutherland.

Sua implementação em software é mais lenta que o processo de utilização do cálculo direto da interseção da linha com as bordas da janela. No entanto, sua implementação em hardware é mais rápida por podermos utilizar arquitetura paralela e divisão e multiplicação por 2 é muito rápido.

Divisão por 2, com bits:

n° 6 = 0110 $\Rightarrow$ deslocando 1 bit para a direita, temos o n° dividido por 2 $\Rightarrow$ 0011 (em decimal = 3)

Novamente, um teste inicial é efetuado para detectar linhas trivialmente invisíveis ou normalmente visíveis. Linhas para as quais o teste inicial falha são subdivididas em 2 partes iguais.

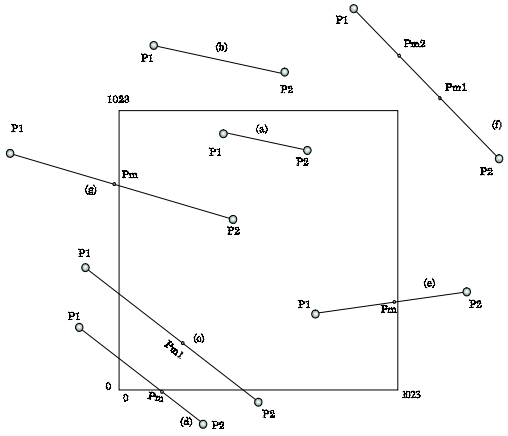
$$X_m = \frac{X_1 + X_2}{2} \quad ; \quad Y_m = \frac{Y_1 + Y_2}{2}$$

O teste é aplicado em cada uma das metades até que a interseção com as bordas da janela é obtida ou o comprimento da parte decorrente da subdivisão é infinitesimal. A visibilidade do ponto é então determinada.

(busca logarítmica)

ILUSTRAÇÃO DA TÉCNICA

Seja a janela abaixo e as linhas conforme esboçadas: $\Rightarrow$



Linha ‘f’: P1-1000; P2-0010 $\Rightarrow$ necessário checar

Subdivisão $\Rightarrow$ P_{m1} -1010 P2-0010 $\Rightarrow$ trivialmente rejeitado

Subdivisão $\Rightarrow$ P_{m1} - 1010 P1-1000 $\Rightarrow$ trivialmente rejeitado

Subdivisões sucessivas levam a rejeitar completamente a linha

Linha ‘c’: P1 (0001) P2 (0100) $\Rightarrow$ necessário checar ponto médio P_{m1} (0001) $\Rightarrow$ mesmo resultado para ambos os lados

* Analisando P_{m1} P2 inicialmente:

- dividimos em P_{m2} $\Rightarrow$ $\begin{cases} P_{m1}(0000) \\ \\ P_{m2}(0000) \end{cases}$ $\Rightarrow$ totalmente visivel

$\begin{cases} P_{m2}(0000) \\ \\ P_2(0100) \end{cases}$ $\Rightarrow$ parcialmente visivel

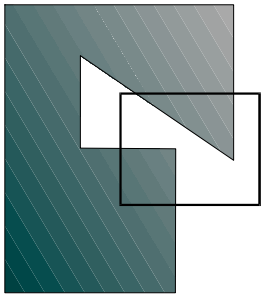
$\Rightarrow$ Traça-se P_{m1} P_{m2}; continuamos a divisão de P_{m2} P_{m2}

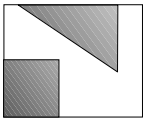
Por divisões sucessivas, segmentos menores poderão ser traçados e a interseção com a janela é obtida.

O segmento P_{m1} P_1 é então analisados da mesma forma. O problema de traçar pequenos segmentos é a ineficiência. O ideal é obter os dois pontos e traçar a linha que está entre eles (dentro da janela, é claro).

6.3. Recorte de polígonos:

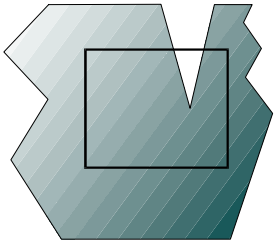
Em algumas aplicações, há necessidade de promover o recorte de polígonos, cujos vértices estão armazenados numa estrutura de dados qualquer. Para serem exibidos, os polígonos devem primeiro passar por uma operação de transformação de visualização e depois por um processo de recorte até serem convertidos nas coordenadas do equipamento. O algoritmo que recorta polígonos deve prever diferentes casos, tais como:

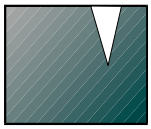




Polígono côncavo é recortado em 2

polígonos separados e distintos.





6.3.1. Algoritmos de Sutherland Hodgman

Estratégia de solução de uma série de problemas simples e idênticos que resolvem o problema quando combinados:

Solução: Recortar uma polígono através do recorte de suas laterais (áreas que tocam os limites da janela)

Serão feitos 4 recortes:

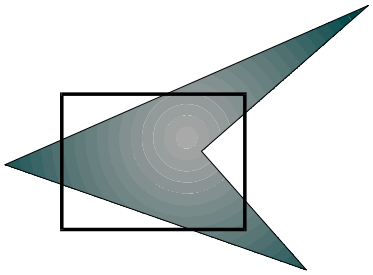
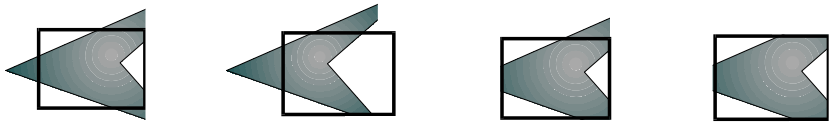


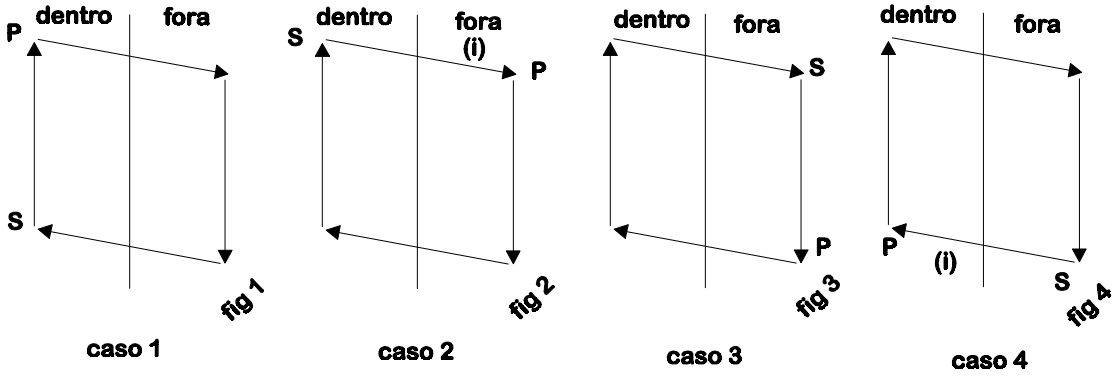
Figura: Recorte de polígono,

lado a lado:



- Com o polígono de lados definidos pelos vértices: U_1, U_2, U_3, ... U_n
- para cada lado observa-se a relação entre vértices sucessivos e as janelas (limites)

lados definidos pelos vértices da lista de saídas serão apresentados na tela. Casos possíveis:



CASO 1: dois vértices é adicionado à lista de saídas (p, no caso)

CASO 2: o ponto "i" de interseção é tratado como um vértice de saída (a ser traçado)

CASO 3: os dois vértices são descartados.

CASO 4: os dois pontos "i" e "p" são colocados na lista de vértices de saída.

Algoritmo destinado a obter as interseções

- O primeiro ponto não é colocado na lista de saídas (s, da fig. 1, anterior), já que o mesmo é o vértice inicial e já se encontra na mesma, pois o processo é seqüencial.
- para linhas do polígono totalmente invisível, nenhum ponto é adicionado à lista de saídas (caso 3).
- para casos 2 e 4, é necessário calcular as interseções
- para um dado vértice, se necessário calcular se o mesmo está dentro ou fora de uma janela, temos uma função que aplica um teste baseado em produto vetorial, como:

$$P_1 . P_2 \quad X \quad P_1 . P_3 = X$$

$$P_1 . P_2 \quad X \quad P_1 . P_4 = \quad .$$

Faz-se o produto vetorial de $P_1 . P_2$ X $P_1 . P_3$, caso o mesmo resultado em vetor entrando (módulo negativo) o ponto está do lado de dentro da janela, se o mesmo for positivo, o ponto está do lado de fora da janela.

O módulo do produto vetorial de dois vetores $V(V_x, V_y)$ e $W (W_x, W_y)$ é um vetor cuja magnitude é dada por $Vx\ W_y - V_y\ W_x$. Se este n° for positivo $\Rightarrow$ ponto fora. Se este n° for negativo $\Rightarrow$ ponto dentro.

6.3.2 **Algoritmo de Weiter-Atherton**

O algoritmo anteriormente discutido requer uma região de recorte convexa. Em contexto de tratamento de linhas escondidas, muitas regiões côncavas aparecem.

O método proposto por W.A. é mais complexo e mais poderoso que o método anterior. Ele é capaz de recortar um polígono côncavo com "buracos" interiores definindo um outro polígono côncavo.

Polígono a ser recortado ⇒ *polígono sujeito*

Polígono que é definido na região recortada ⇒ *polígono a ser recortado*

Região de recorte : *polígono recorte*

O algoritmo descreve ambos, polígono sujeito e polígono recorte por uma lista circular de vértices e recorta o polígono sujeito por traçar em voltas das laterais do polígono sujeito na direção horária até que a interseção com o polígono recorte seja encontrada. Se a fronteira entre no polígono recorte o algoritmo procede ao longo dos limites do polígono sujeito. Se a fronteira deixa o polígono recorte, o algoritmo faz um retorno à direita e segue o polígono recorte no sentido horário até encontrar a interseção de onde partiu.

7. PROJEÇÕES

Definição: enxergamos em 2 dimensões ⇒ não há muita necessidade de tratamento em 3 dimensões. Logo, modelos tridimensionais devem ser convertidos em imagens bidemensionais, esta é a finalidade do algoritmo de projeção. Projeção é conversão genérica de entidades de uma dada dimensão para outra de menor ordem.

Nosso estudo. Projeção: 3D⇒2D (interesse nas diversas aplicações compugráficas)

7.1. INTRODUÇÃO: TIPOS DE PROJEÇÃO (3D⇒ 2D)

O que determina a projeção são 2 elementos:

* plano de projeção ⇒ quadro
* centro de projeção ⇒ ponto de vista (c)

Há diversas técnicas para projetar 3D⇒2D:

- projeção paralela; casos particulares
- projeção axionométrica; do espelhamento
- projeção perspectiva; (+ interesse)

A projeção de um corpo fica definida pelo conjunto de pontos do quadro obtidos da sua interseção com as linhas visuais - projetoras - que saem de **o** e atingem **c.**

É claro que o tamanho e a forma da figura projetada dependem da forma de posicionamento do objeto em relação ao plano de projeção e ao centro. Observe que a projetora é sempre uma linha reta (logo, são projeções geométricas) como o quadro é um plano, logo, são projeções planas.

Classificação em relação ao centro de projeção

a) **Projeção Cônica-perspectiva:** o centro de projeção é um ponto próprio, em coordenadas finitas no sistema tridimensional. Esta projeção deforma a figura, diminuindo os objetos mais distantes e distorcendo os ângulos.

b) **Projeção cilíndricas-paralela:** tem um ponto impróprio como centro de projeção - isto é; as linhas visuais encontram-se no infinito. Mantém a proporcionalidade da figura.

7.2. Transformações de projeção :

A projeção de um objeto pode ser formulada como uma transformação como as vistas anteriormente. Desta forma, pode ser representada por uma matriz 4x4 que, aplicada à um ponto do espaço obtenha um ponto do plano.

Considerações para obter as matrizes de transformação:

- O objeto a ser projetado deve estar descrito em relação a um sistema de coordenadas tal que as direções principais do mesmo coincidam com os eixos do sistema;

Caso não seja possível definir os eixos principais de maneira clara, usa-se um paralelepípedo envoltório.

- O plano de projeção é um plano vertical, colocado perpendicularmente ao eixo z do sistema de coordenadas do objeto;
- O objeto está modelado por um conjunto de pontos convenientemente.

OBS.: havendo mais de um objeto em cena é necessário uma conversão entre os sistemas de coordenadas do objeto e da cena. Os pontos de cada objeto devem ser convertidos para o sistema global por uma transformação de mudança de base, antes de se efetuar as transformações de projeção:

7.3. PROJEÇÕES CÔNICAS

i) **do ponto**: basta ligá-lo ao centro de projeção e obter a interseção da reta com o plano de projeção.

ii) **da reta:** idem ao anterior para dois pontos da reta

7.3.1. Cálculo da transformação de perspectiva:

Por semelhança de triângulos, podemos encontrar:

$$\frac{Y'_p}{d} = \frac{Y_p}{Z_R}; \qquad Y'_p = \frac{Y_p . d}{Z_R} = \frac{Y_p}{Z_R / d}$$

$$\frac{X'_p}{d} = \frac{X_p}{Z_p} \quad \Rightarrow \quad X'_p = \frac{X_p . d}{Z_p} = \frac{X_p}{Z_p / d}$$

Destas relações obtêm-se a matriz de transformação da perspectiva cônica:

Obs: Se o centro de projeção for (0,0,d) teríamos:

$$P_{con} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad X = \frac{\frac{X}{Z+d}}{d} \; ; \quad Y' = \frac{\frac{Y}{Z+d}}{d}$$

Assim

$$[Xp\ Yp\ Zp\ 1] . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{matrix} [Xp\ Yp\ Zp\ \ Zp/d] \\ \left[\frac{Xp}{Zp/d} \quad \frac{Yp}{Zp/d} \quad d \quad 1\right] \end{matrix}$$

Exemplo: Seja o objeto definido pelas tabelas abaixo: e plano de projeção Z=Z0, simule a projeção do mesmo na tela de um computador:

PONTO			
1	0	0	0
2	0	0	0
3	0	0	5
4	0	0	0
5	0	0	0

lado (1, P1, P2)
lado (2, P1, P3)
lado (3, P1, P4)
lado (4, P1, P5)
lado (5, P2, P3)
lado (6, P2, P4)
lado (7, P3, P5)
lado (8, P4, P5)

Poderíamos ter perspectivas definitivas sobre 2 e 3 pontos de projeção. Dadas por:

$$\text{Pontos de projecao} \begin{cases} x = r_1 \\ y = r_2 \end{cases} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1/r_1 \\ 0 & 1 & 0 & 1/r_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{Pontos de projecao} \begin{cases} x = r_1 \\ y = r_2 \\ z = r_3 \end{cases} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 1/r_1 \\ 0 & 1 & 0 & 1/r_2 \\ 0 & 0 & 1 & 1/r_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Anomalias da perspectiva

Perspectiva $\Rightarrow$ gera anomalias que aumentam o realismo em termos de profundidade, mas alteram as medidas e formas reais.

1. Encurtamento perspectivo: aumentando a distância do objeto ao centro de projeção menor parece ser;
2. Pontos de fuga: as projeções são categorizadas pelo número de pontos de fuga principais (n° de eixos que o plano de projeção corta). Se a projeção é com 1 ponto de fuga principal então o plano de projeção corta o eixo z e linhas paralelas aos eixos x e y não convergem.
3. Confusão visual: objetos situados atrás do centro de projeção são projetados no plano de projeção de cima para baixo e de trás para frente, conforme:

4. Distorção topológica: Fenômeno pela qual um segmento de reta que une um ponto situado à frente do observador com um ponto situado à sua retaguarda é efetivamente projetado segundo uma linha quebrada de comprimento infinito. A causa é o fato de que pontos do plano que contem o ponto central da projeção são projetados no infinito pela transformação perspectiva.

OBS: IMPORTANTE: Há possibilidade da existência de um, dois ou três pontos de fuga principais, se o plano de projeção é perpendicular (e a normal ao plano, paralela) e um, dois ou três eixos.

7.3.2. Projeções Cilíndricas (ou paralelas)

Há 2 tipos de projeções paralelas, baseadas na relação entre a *direção de projeção* e a normal ao plano de projeção:

i) Proj. Ortogonal: a direção de projeção é a mesma direção da normal ao plano de projeção;

ii) Proj. Oblíqua: as direções citadas acima não são as mesmas.

7.3.3.PROJEÇÕES OBLÍQUAS

- Fornecem sensação espacial e permitem medidas
- Neste caso, a direção de projeção não forma 90° com o plano de projeção, mas o plano de projeção é paralelo a um dos 3 eixos.
- Geralmente: faz-se uma face paralela ao plano de projeção, normalmente a que contém mais detalhes. Esta face projeta-se em sua verdadeira grandeza, de forma a evitar a deformação das formas circulares desta face.

DEFINIÇÕES MATEMÁTICAS:

Seja o cubo unitário ao lado que dever ser projetado no plano xy. A matriz para projeção oblíqua pode ser escrita em função de x e l.

Note que (0, 0, 1) pode ser projetado em xy como:

$$(l.\cos\alpha,\ l.\sin\alpha),$$

levando o outro ponto no espaco dado por

$$P'(l.\cos\alpha,\ l.\sin\alpha,\ \theta)$$

Como a linha projetora, que não é perpendicular ao plano de projeção, deve passar por P e P' e as demais serão paralelas à ela, chegamos à:

Considerando um ponto genérico (x, y, z): sua projeção é dada por (x_p, y_p, θ). Com estes 2 pontos e considerando o vetor PP', que define a reta projetora, temos, como equação simétrica da reta:

$$\frac{x - x_p}{l.\cos\alpha} = \frac{y - y_p}{l.\sin\alpha} = \frac{z}{-1};$$ destas relacoes, temos

$$\frac{x - x_p}{l.\cos\alpha} = -z \quad \Rightarrow \quad x_p = x + z.l\cos\alpha$$

$$\text{e} \quad y_p = y + z.l\sin\alpha$$

e chegamos à matriz da projeção oblíquo:

$$Pobl. = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l.\cos\alpha & l.\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obs.:

- se l = 1 e $\alpha = 45°$ ($\beta = 45°$, também) => projeção cavaleira (cavalier)
- se l = 1/2 e $\alpha = 45°$ [(β = arctg 2) (aprox: 63,4°)] a projeção projeção é dita gabinete (cabinet)

Proj. cavaleira: a projeção de uma linha perpendicular ao plano de projeção é de mesmo comprimento que a linha em si.

Exemplos:

Proj. cabinet: direção de projeção forma aprox. 63,4° com o plano de projeção.

1. Para um cabo de aresta de 8 cm, encontre a perspectiva cavaleira do mesmo, supondo o mesmo com um dos vértices na origem do sistema de coordenadas (x, y, z)
2. Obtenha a projeção “cabinet” para o cubo acima.

8. TRATAMENTO DE LINHAS / SUPERFÍCIES ESCONDIDAS

8.1. INTRODUÇÃO: Profundidade da perspectiva

A operação de remoção de linhas ou superfícies escondidas requer uma transformação perspectiva com propriedades especiais. É necessário avaliar a profundidade de cada ponto na vista perspectiva para tomar decisões sobre que pontos / superfícies da imagem devem ser omitidas na representação.

Sistema 3D - x_s, y_s, z_s { x_s e y_s => localização da vista perspectiva do ponto

{z_s => retém a informação de profundidade

Se para apresentar em 2D, despreza-se z_s e utilizam-se as outras 2 coordenadas sem modificação. Esta representação corresponde à uma projeção ortogonal do sistema de coordenadas da tela.

Ilusões de profundidade

A ilusão do cubo-Necker:

8.2. Classificação de Algoritmos

Os algoritmos para remoção de linhas / superfícies escondidas são classificados de acordo com o tratamento utilizado:

- object-space: diretamente com as definições do objeto;
- imagem-space: com suas imagens projetadas.

Método object-space: compara objetos e partes dos objetos entre si, para definir as partes que deveriam ser rotulados como invisíveis. Cálculos geométricos com maior precisão mais usados no tratamento de linhas escondidas.

Método image-space: a visibilidade é decidida ponto a ponto em cada posição do pixel sobre o plano de projeção mais usados no tratamento de superfícies escondidas.

3. Introdução:

Dado um objeto 3D e uma especificação de visualização definindo o tipo de projeção, nós podemos definir que limites e superfícies do objeto são visíveis para o centro de projeção (p/ proj. perspectiva) ou ao longo da direção de projeção (p/ proj. paralela).

Vamos observar que a idéia fundamental é simples, se a implementação é mais simples, então requer uma grande demanda de tempo computacional, outros algoritmos mais complexos podem ser estruturados.

2 propostas são elaboradas para tal solução.

IMPORTANTE: os algoritmos devem fazer cada passo o mais eficiente possível.

8.3. ALGORITMO DEPTH-BUFFER:

8.3.1. INTRODUÇÃO:

Trata-se de um algoritmo dos mais simples, sendo o mais simples da classe dos métodos image-space. Este algoritmo é algumas vezes chamado de z-buffer. Para cada ponto da tela gráfica, nós guardamos um registro da profundidade do objeto. Adicionalmente, é armazenada a intensidade que deverá ser usada para apresentar o objeto. O algoritmo requer 2 arrays, uma para intensidade e uma para profundidade, cada uma das quais é indexada pelas coordenadas do pixel (x,y).

8.3.2. ALGORITMO:

1. Para todo pixel da tela, faça depth [x,y]=1.0 e intensidade [x,y]= valor de fundo da tela (background);
2. Para cada polígono, encontre os pixel (x,y) que estão associados aos limites do polígono quando projetados na tela.

   Para cada um destes pixel:

   a. Calcular o valor da profundidade z do polígono para posição (x,y);

   b. Se z < depth [x,y] => coloque z (x,y) no z-buffer ou melhor, faça depth [x,y] = z e faça intensidade (x,y) igual à intensidade do polígono, caso contrário:

se z > depth [x,y] => não tome nenhuma ação

Depois de processados todos os polígonos, a array “intensidade” conterá a solução

Observe a figura abaixo:

Caso z < depth (x,y) o polígono está mais próximo para o observador que outros já gravados para este pixel.

Se z>depth[x,y], o polígono já gravado para (x,y) está situado mais perto do observador que o novo polígono e nenhuma ação e requerida.

c) comentários:

O algoritmo tem a desvantagem de requerer um grande espaço para o z-buffer [ou depth-buffer], mas, é muito simples de ser implementado. A performance do algoritmo tende a ser constante, o número de pixel coberto por cada polígono decresce na medida em que o número de polígonos inserido no volume aumenta.

Um outro importante detalhe do algoritmo é que ele usa o sistema de coordenadas da tela. Antes do passo 1, todos os polígonos da cena são transformados no sistema de coordenadas da tela, já descritos anteriormente.

d) Cálculo de z[x,y]

Pelo fato de haver um registro da equação do plano para cada polígono no sistema de coordenadas da tela, para calcular z[x,y] basta resolver a equação do plano:

$$Ax + By + Cz + D = 0$$

E o valor da variável z é dado por :

$$Z = \frac{- D - Ax - By}{C} \quad \text{(equ. do plano - 1)}$$

e) Limitações do método:

O método não é muito utilizado na prática por causa do enorme tamanho das áreas “profundidade” e “intensidade” (2 arrays). Num equipamento de 500 x 500 pixels teríamos 250 000 locações para cada array.

Para redução da área acima, a imagem pode ser dividida em imagens menores, por exemplo, no caso acima, dividiríamos em 100 trechos, cada uma com 50x50 pixels. Caímos para arrays com 2500 elementos, mas o tempo de execução cresce por que cada polígono é processado várias vezes.

8.4. ALGORITMOS DE CONEXÃO DE LINHAS DE VARREDURA

8.4.1. Introdução

Tais algoritmos resolvem o problema de linhas escondidas por meio de uma linha de varredura por vez, processando linhas de varredura da parte superior para inferior da tela. O algoritmo examina uma série de janelas na tela, cada janela é uma linha de varredura alta e tão larga quanto a (janela) tela. Novamente, serão re-querida duas arrays: intensidade e profundidade.

8.4.2. Algoritmo:

Para cada linha de varredura, faça :

1 - Para todo pixel da linha de varredura, faça depth[x,y]= 1,0 e intensidade[x,y]= valor de fundo da tela - background;

2 - Para cada polígono na cena, encontre todos os pixels na atual linha de varredura “y” que corta o polígono:

a - Calcule a profundidade z do polígono no (x,y)

b - se z <profundidade (x), faça depth[x] = z e intensidade [x] para a intensidade correspondente ao polígono.

3 - Após todos polígonos terem sido considerados, os valores contidos na array “intensidade” representam a solução.

Na figura abaixo, é apresentada a projeção de 2 triângulos no plano x-y; Linhas escondidas são apresentadas como detalhadas:

A tabela de dados, conforme visto anteriormente, é desenvolvida e armazenada de forma a conduzir o processo de análise das faces. Para a linha de varredura y=α, tal tabela contém somente AC e BC, nesta ordem. As faces devem ser processadas da esquerda para a direita.

No caso da linha “ B “ a ordem é AB, AC, FA e FE. A linha de varredura encontra 2 polígonos mas a varredura está dentro (in) somente em um polígono por vez.

No caso da linha de varredura y = γ, na tabela de polígonos o flag “in-lout” que é inicializado por “falso” é colocado para “verdadeiro”. A linha de varredura prossegue encontrando a outra face, DE. Neste ponto, o flag para DEF torna-se “verdadeiro”, mas a varredura está “dentro” de 2 polígonos. É necessário então decidir qual destes polígonos está mais próximo do observado. Esta determinação é feita com uso da equação do plano para ambos os polígonos para Z, y = γ e x dado pela interseção de y = γ com a face DE. O valor de x está efetivamente na tabela de lados. No nosso exemplo, DEF tem um “ Z ” menor e portanto é visível. Portanto, o sombreamento para DEF é usado para esconder a linha BC, tornando o flag para ABC como falso, já que alimentação linha de varredura está dentro somente do polígono DEF, obtendo agora encontro com a face FE.

8.5. Algoritmo de subdivisão de área

8.5.1. Introdução:

A idéia básica deste algoritmo é, utilizando-se de uma área de estudo (ou visualização), decidir, conforme fizemos com recorte, se um (ou uma parte de um) ou mais polígonos estão contidos na mesma área. A área pode ser particionada em áreas menores e a decisão lógica é tomada por recursão aplicada à cada uma das pequenas áreas. Com áreas cada vez menores, menores e menores polígonos sobrepõem-se em cada área.

Há 4 casos possíveis de relação entre áreas de interesse e polígonos:

É obvio que polígonos disjuntos não são de interesse para o estudo; nossos problemas deverão ocorrer para os casos b, c, a acima.

Casos:

1. Todos polígonos disjuntos à área => valor do pixel = fundo de tela;
2. Somente 1 polígono contido / interceptado => solução é completar o restante da área para a cor do fundo;
3. Há somente 1 polígono que transborda a área => toda a área é colocada na cor do polígono;
4. Há mais de um polígono contido, interceptado ou que transborde e pelo menos um deles transborda. É promovido um teste para detectar se o que transborda esta a frente dos demais, analisando a coordenada z de cada um dos planos.

É obvio que os 3 primeiros casos acima são de solução muito simples. Nos casos da hipótese 4, onde mais de um polígono pertencem à mesma área, necessitamos olhar a coordenada do plano na qual está contido e ver qual delas está mais próxima do observador. Se, no entanto, os planos dos 2 polígonos se interceptam, será necessário uma subdivisão para análise posterior, conforme:

8.6.Algoritmos do tipo object-space

8.6.1. POLÍGONOS

i) Equação do plano: baseado na equação do plano que contém as faces do objetos à ser exibido. Observe

Equação do plano: Ax + By + Cz + D = O, onde:

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2);$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2);$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2);$$

$D \Rightarrow$ retirado da equacao

N= (A, B, C)

Aplicando a equação do plano para coordenadas do observador, temos:

$$A.x_0 + By_0 + Cz_0 \begin{cases} < 0 \Rightarrow \text{fora} \Rightarrow \text{exibe} \\ > 0 \Rightarrow \text{dentro} \Rightarrow \text{esconde} \\ = 0 \Rightarrow \text{fronteira} \end{cases}$$

8.6.2. Por produto escalar:

- descrever as arestas de cada face no sentido anti-horário para quem está fora do objeto;
- obtém o vetor *u* fazendo o produto vetorial das arestas;

  $u.v < 0$ => sentidos opostos => exibe

  $u.v > 0$ => sentidos iguais => esconde

9. SOMBREAMENTO

Em seguida a remoção de sup/linhas escondida é importante, no processo de criação de imagens, sombrear as superfícies visíveis.

9.1. Introdução:

Reflexão difusa e luz ambiente

Todas superfícies visíveis promovem reflexão difusa da luz, espalhando luz igualmente em todas as direções. para todas as superfícies, lei dos cosenos de Lambert relata a quantidade de luz refletida em função da luz incidente e da superfície em questão.

A iluminação difusa (Id) é dada por:

$Id = Ip.\ Kd.\ \cos\theta \quad \left(0 \le \theta \le \pi/2\right)$

onde: Ip = intensidade do ponto de luz incidente;

Kd = coeficiente de reflexão difusa; (varia de 0 a 1)

θ = ângulo entre a direção L para o ponto de luz incidente e a normal à superfície.

ou ainda, por produto vetorial:

Id = Ip. Kd.(*L.N*)

Considerando a luz ambiente, teríamos uma nova equação:

Id = Ia. Ka + Ip. Kd. (*L.N*)

onde

Ia => intensidade de luz ambiente

Ka => quanto da luz ambiente é refletida pela superfície.

Obs.: A fonte de luz é assumida convenientemente como sendo coincidente com os olhos do observador, logo, não há sombras; logicamente, os raios de luz que colidem com a superfície serão todos paralelos.

Problema: 2 superfícies de mesma cor paralelas e uma mais próxima ao examinados que a outra, como seus vetores normais são iguais, a iluminação nas superfícies será a mesma e as mesmas serão indistinguíveis.

Solução: A energia luminosa diminui com o inversos do quadrado da distância (d = distância da luz (ida e volta)) chamado tal distância de r, temos:

$$Id = Ia.Ka + Ip.\ Kp\ (L.N)/\ R^2$$

Mas se $R \rightarrow \infty$ => problemas de simulação => hipoteticamente, os objetos seriam capazes de refletir apenas a luz ambiente. Para obtenção de efeitos mais realísticos, a equação acima pode ser melhorada para:

$$Id = \frac{Ia.Ka + I.p.\ Kd.\ (L.N)}{(r+K)}$$

onde K => constante e r => distância do ponto de vista à superfície em questão.

Considerando superfícies coloridas, teríamos equações diferentes para cores diferentes: Azul, vermelho e amarelo, com a tripla (Kdc, Kdm, Kdy) definindo as constantes de reflexão para cada cor.

Para o amarelo, teríamos Kdy = 1.0, Kdc = 0.0, Kdm = 0.0

fazendo o amarelo ser refletido e as demais serem absolvidas.

Para uma dada componente:

$$Id = Iac.Kac + \frac{I.pc.\ Kdc.\ (L.N)}{(r+K)}$$ para a componente Azul (CYAN)

9.2. REFLEXÃO ESPECULAR

Este tipo de reflexão ocorre em algumas superfícies brilhantes. Ex: iluminação da superfície de uma maçã => brilho intenso é causado por reflexão especular, fazendo, inclusive não perceber a real cor da maçã, o vermelho, o resto da maçã é visto por reflexão difusa.

Para uma superfície que reflete perfeitamente a luz o ângulo de reflexão é igual para o ângulo de incidência. Assim, somente o observador colocado nesta linha verá reflexão especular. Ou seja, para reflexão especular, o ângulo α deve ser zero, assim, v e R coincidem. a intensidade da reflexão especular é maior na direção de R e diminui rapidamente na medida em que α aumenta.

Modelo de Phong:

Intensidade da reflexão especular $\cong \cos^n\alpha$, onde "n" identifica o tipo da superfície (função da luz refletida)

$Is = Il.\ w(i,\lambda)\ \cos^n\alpha.$

$w(i,\lambda)$ = curva de reflectância

i => ângulo de incidência

λ => comprimento de onda

n grande => distribuição de luz da forma de metais

n pequeno => superfícies não metálicas

* Função de distribuição espacial para luz refletida. especularmente.

Reflexão especular é direcional, depende do ângulo de incidência da luz. Incidência perpendicular de luz pode causar apenas reflexão especular. Ex: para materiais não-metálicos a refletância pode ser apenas 4%, para materiais metálicos pode ser 80%.

Combinando resultados com o modelo anterior, temos o modelo de iluminação dado por:

$$I = Ia.Ka + \frac{Ip.}{r+K} [Kd.\cos\theta + w(i.,\lambda) \cos^n\alpha]$$

$$I = Ia.Ka + \frac{Ip.}{r+K} [Kd.\cos\theta + Ks.\cos^n\alpha]$$

ou ainda:

$$Ia.Ka + \frac{Ip.}{r+K} [Kd.(L.N) + Ks.(R.V)^n]$$

O cálculo de L.N pode ser dado por:

$$L.N = \frac{n.L}{|n|.|l|}$$

9.3. CONSIDERAÇÕES

Se a fonte de luz está no infinito, L.N é constante para um dado polígono, enquanto o produto R.V varia ao longo do polígono. Para superfícies bicúbicas, ou para fonte de luz que não está no infinito, tanto L.N como R.V variam ao longo da superfície. Assim, tais produtos vetoriais deverão ser calculados para cada ponto numa linha de varredura.

O método Torrance- Sparrow (adota tal modelo) é um modelo baseado na superfície de reflexão, em contraste com o modelo empírico proposto por Phong ($\cos^n\alpha$).

A superfície é assumida como uma coleção de faces microscópicas, cada uma como um perfeito refletor. A orientação de cada faceta é dada por função de distribuição de probabilidade Gausiana. A geometria de cada faceta e a direção da luz determinam a intensidade e a direção da reflexão especular.

a => totalmente exposta a luz;

b => escura;

c => parcialmente escura;

e => totalmente exposta a luz, mas alguns raios sofrem desvio na superfície “f”

Blim adaptou o modelo Torrance-Sparrow para computação gráfica, dando detalhes matemáticos e comparando com o modelo de Phong. Os dois modelos diferem consideravelmente no caso de ângulos de incidência em torno de 70°, quando o modelo de Sparrow apresenta mais reflexão especular da luz.

Ref.: livro Foley - pag 580

9.4. SOMBREAMENTO DE MALHAS POLIGONAIS

* 3 métodos para sombrear objetos definido por malhas poligonais:

i) sombreamento constante;
ii) sombreamento com interpolação da intensidade;
iii) sombreamento com interpolação do vetor normal. (aumento da dificuldade)

Em cada um deles, deve-se fazer a opção por um dos dois modelos de iluminação proposto anteriormente.

9.4.1. Sombreamento constante:

uma superfície curva seria representada por um conjunto de superfícies planares que pode ser sombreada com intensidades constantes de iluminação em cada superfície, desde que os planos que subdividem a superfície sejam suficientemente pe-

quenos. Esta aproximação pode gerar como efeito, exibições razoavelmente boas em muitos casos, especialmente se a curvatura muda gradualmente e a fonte de luz e o observador estão longe da superfície, decorre disto que algumas considerações são necessárias:

1. A fonte de luz está no infinito;
2. O observador também está no infinito;
3. O polígono que representa a atual superfícies que está sendo modelada, não é uma aproximação para uma superfície curva.

Se qualquer uma das 2 considerações iniciais é verdadeira, então a média L e V pode ser usada, talvez calculada no centro do polígono.

A consideração final é de efeito mais substancial que as outras duas na obtenção da imagem resultante. O efeito é que cada faceta do polígono visível da superfície aproximada é distinguível.

Com este método, a intensidade é calculada num ponto interior de cada polígono e a superfície (e a face planar) é sombreada com a intensidade calculada.

Se a orientação entre planos adjacentes muda abruptamente a diferença entre as intensidades das superfícies podem produzir um efeito áspero e irreal, sendo um efeito evidentemente visível => “EFFECT MACH BAND”

9.4.2. Interpolação da intensidade:

de forma a eliminar a descontinuidade da intensidade, propõe-se o sombreamento com interpolação da intensidade, ou, sombreamento de “Gouraud”.

Tal processo é constituído de 4 passos:

a) calcula-se os vetores normais às superfícies;

b) as normais dos vértices são calculadas, usando-se a média das normais de todas as superfícies que compartilham o mesmo vértice:

c) as intensidades dos vértices são encontradas, usando o vetor normal do vértice com um modelo de iluminação desejada;

d) cada polígono é sombreado pela interpolação linear das intensidades dos vértices ao longo da linha de varredura, como mostra a figura abaixo. Para cada face, será armazenado a intensidade inicial e a mudança do valor de y ocasionará a mudança de intensidade.

$$Ia = I_1 \frac{y_v - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_v}{y_1 - y_2}$$

$$Ib = I_1 \frac{y_v - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_v}{y_1 - y_3}$$

$$Ip = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$

Interpolação

Obs.: O resultado da aplicação do método de Gouraud é uma melhora sensível na sensação “irreal” passada pelo método anterior, mas, ainda causa sensação de “mach band”. livro ROGER pág. 323/324

9.4.3.Interpolação do vetor normal / “Phong”:

O procedimento proposto por Phong soluciona muitos problemas do sombreamento de Gouraud. Enquanto Gouraud interpola valores de intensidade ao longo de uma linha de varredura, o processo de Phong interpola o vetor normal à superfície ao longo da linha de varredura e ao longo das arestas. O modelo de iluminação é então aplicado à cada pixel, usando a normal interpolada para determinar a intensidade. Com esta técnica, reflexões especulares parecem mais reais, diminuindo os problemas “Mach Band” e aumentado grandemente o custo computacional.

- ***9.5. Técnica do traçado de raios (Ray Tracing)***

Proposta que resolve os problemas das duas técnicas anteriores, mas, exige grande quantidade de cálculos levando a inviabilidade no caso de máquinas lentas. Seu resultado no entanto é visível na forma de imagens belíssimas que são obtidas com seus cálculos.

vantagem: pode ser aplicado à curvas e poliedros

Técnica:

Projeta-se um raio virtual de luz que, partindo do ponto do observador passa por um pixel na tela e pelo conjunto de objetos que estão sendo visualizados.

- Devem ser computadas as interseções do raio com cada objeto que o mesmo atravessa;

- Se o raio não intercepta nenhum, a cor do pixel será a cor de fundo;

- Se mais de um objeto for interceptado, seleciona-se o mais próximo do observador, este será o ponto visível e os demais serão invisíveis (resolve o problema de linhas escondidas) neste ponto, calcula-se a intensidade refletida da superfície mais próxima do observador, empregando as fórmulas já propostas, deste cálculo resultará o valor da tonalidade do pixel correspondente.

(repete-se o passo para cada pixel na tela)

Destaque: processo com bons resultados.

10. SOMBRAS

Técnicas semelhantes à remoção de linhas escondidas, só que algoritmos de remoção de linhas escondidas determinam que superfícies devem ser vistas para um dado ponto de vista, aqui, determinamos que superfícies podem ser vistas por uma dada fonte de luz e para um ponto de vista, a mesma não é sombreada. A mesma lógica aplica-se à diversas fontes de luz.

O sombreamento contribui sensivelmente para dar realismo à cena por aumentar a sensação de profundidade, sendo também importante nas simulações.

Sombras : - sombra - (CG só usa esta)

- penumbra

Fonte de luz no infinito: projeção ortográfica é usada para determinar as sombras;

Fonte de luz a uma distância finita, mas fora do campo visual do observador: usa-se a projeção perspectiva;

Fonte de luz a uma distância finita, mas no campo visual do observador: o espaço deve ser dividido em setores e as sombras obtidas por setor, separadamente.

11. FORMAS E MODELOS GEOMÉTRICOS

11.1. Introdução:

Modelação: descrição de objetos e de imagem de maneira à permitir sua visualização.

Muitas aplicações de CG envolvem a representação de objetos em 3D. Em alguns casos, uma grande quantidade de triplas (x,y,z) são necessárias para descrever as superfícies do objeto em questão. Em alguns casos, no entanto, o objeto pode ser definido por combinação de superfícies matemáticas que, são suficientemente simples, a ponto de poderem ser implementadas computacionalmente.

Formas simples para primitivas: pontos, segmentos de reta, linhas poligonais, polígonos, poliedros.

Em alguns casos, uma coleção de linhas é insuficiente para descrever os objetos, já que linhas não definem superfícies e superfícies são necessárias para cálculos de superfícies escondidas, volumes etc...

Assim, formas geométricas mais complexas são usadas na modelagem de objetos, tais como: arcos de curvas, superfícies curvas, superfícies quádricas.

11.2. Formas Geométricas Simples

- Pontos e segmentos de retas:

  especificação dada por $P_1(x_1,y_1,z_1)$ e $P_2(x_2,y_2,z_2)$

- Linhas poligonais

  cadeia de segmentos de reta adjacentes. É especificada por uma seqüência de nós-vértices (P_0 ,P_1 , ... , P_n) que definem os segmentos de reta.

- Polígonos : linha poligonal fechada.

  arestas :

  chama-se polígono plano ao polígono cujos vértices pertencem a um mesmo plano.

11.3. Malhas Poligonais

Uma malha poligonal é uma coleção de vértices, arestas e polígonos, vértices são unidos por arestas e polígonos constituem uma seqüência de vértices e arestas. Uma malha pode ser representada por diferentes maneiras, com vantagens e desvantagens particulares.

São elas:

11.3.1.**Lista explícita de vértices**

$V = \{P_0, P_1, P_2 \ldots, P_n\}$ e os pontos $P_i(x_i, y_i, z_i)$

são os vértices da malhas poligonal, armazenada na ordem em que seriam encontrados por uma travessia do modelo.

Obs.: representação útil para polígonos isolados, mas ineficiente para uma malha poligonal completa, já que diversos vértices são partilhados mais de uma vez:

11.3.2.Lista de polígonos :

Cada vértice é armazenado uma única vez na lista de vértices $v = (P_0, \ldots, P_n)$ e cada polígono é definido por ponteiros ou índices para uma lista de vértices.

Obs.: de novo, quando da representação do objeto, arestas partilhadas são desenhadas diversas vezes.

11.3.3.Lista explícita de arestas:

há uma lista de vértices e cada vértice é armazenado uma única vez e uma lista de arestas, com cada aresta armazenada também uma única vez. Um elemento da lista d arestas aponta dois elementos da lista de vértices.

Uma polígono é então definido por uma lista de ponteiros para lista de arestas.

forma mais consistente, sendo mais fácil de ser checada, por conter mais informações.

Obs. finais:

As malhas poligonais são usadas em aplicações de engenharia, por serem fáceis de construir, mas, para construção de modelos mais realistas exigem um número muito elevado de polígonos para produzir ilusão de curvatura.

11.4. SUPERFÍCIES CURVAS

Permitem nível de modelagem mais elevado;

- formas de modelagem { segmentos de superficies curvas; modelagem solida
- maneiras de modelação { aditiva (uniao); subtrativa (escultura)

11.4.1. Modelação de Curvas

dados n+1 pontos: $P_0\,(x_0,y_0)$, ... , $P_n\,(x_n,y_n)$ pretende-se determinar a curva que corresponde a formato definida por estes pontos.

podemos querer que:

- A curva passe pelos pontos (interpolação)
- A curva se aproxime dos pontos: (aproximação)

A solução destes problemas passa por definir métodos de construção de curvas a partir de segmentos de curvas, geralmente modelados como linhas poligonais.

Na modelagem de uma curva f(x) usando segmentos de curvas, procura-se representar a curva como uma som de segmentos *i(x), designados funções de base.

Assim:

Obs.: escolhe-se convenientemente estas funções de base, geralmente associadas a funções polinomiais.

Polinômio de grau n:

Polinômios segmentados contínuos:

Q(x) é um polinômio segmentado contínuo de grau n. É definido como um conjunto de K polinômios $q_i(x)$, cada um de grau n e (K+1) nós t_0 , t_1 , ... , t_k de modo que:

para $t_i \leq x \leq t_{i+1} \Rightarrow Q(x) = q_i(x)$ e $i = 0 ; ... ; k - 1$

forçando os polinômios a coincidirem por nós.

ex.:

Obs.: Polinômios de grau elevado não são muito úteis para modelação de curvas. (Preferência: grau 3)

11.4.2.Funções polinomiais de base

Descreve-se a seguir algumas funções polinomiais de base dados:

$P_0 (x_0 , y_0) , ... , P_n (x_n , y_n) \Rightarrow n + 1$ pontos dados

$t_0 , t_1 , ... , t_n \Rightarrow$ nós $\Rightarrow$ números reais

- Polinômios de Lagrange
- Polinômios cúbicos de Hermite

A forma de polinômios cúbicos de Hermite é determinada pelos pontos finais e pelas tangentes aos pontos finais.

Figuras:

B-Splines:

Para uma sequência de nós $t_0 , t_1 , ... , t_n$ os polinômios $B_{i,n}$ são dados por:

Em particular , o polinômio cúbico B-spline $B_{i,3}$ é não nulo no intervalo $[t_i , t_{i+4}]$

Para nós não repetidos, temos:

$t_i \neq t_{i+1 n}$, o polinômio B-spline é zero nos extremos, ou seja, t_i e t_{i+n+1}.

- Polinômios de Bernstein

Os polinômios de Bernstein de grau “n” no intervalo [0,1] são definidos como:

Os polinômios cúbicos de Berstein serão: