

Transformações Geométricas

Translação:

$$\begin{bmatrix} x_T \\ y_T \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_T \\ y_T \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matriz de Translação: $T(\Delta_x, \Delta_y) = \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \end{bmatrix}$

Escalamento:

$$\begin{cases} x_S = x \times S_x \\ y_S = y \times S_y \end{cases} \Leftrightarrow \begin{bmatrix} x_S \\ y_S \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

• O escalameto define-se em relação à origem das coordenadas

Matriz de Escalamento: $S(S_x, S_y) = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$

Rotação:

$$\begin{cases} x = R \cos(\alpha) \\ y = R \sin(\alpha) \end{cases} \rightarrow \begin{cases} x_R = R \cos(\alpha + \beta) \\ y_R = R \sin(\alpha + \beta) \end{cases} \Leftrightarrow \begin{cases} x_R = R \cos(\alpha) \cos(\beta) - R \sin(\alpha) \sin(\beta) \\ y_R = R \sin(\alpha) \cos(\beta) + R \cos(\alpha) \sin(\beta) \end{cases}$$

$$\Leftrightarrow \begin{cases} x_R = x \cos(\beta) - y \sin(\beta) \\ y_R = x \sin(\beta) + y \cos(\beta) \end{cases} \Leftrightarrow \begin{bmatrix} x_R \\ y_R \end{bmatrix} = \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

• A rotação define-se em relação à origem das coordenadas

Matriz de Rotação: $R(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$

• A composição de transformações geométricas não é comutativa

Matriz de Translação: $T(\Delta_x, \Delta_y) = \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \\ 0 & 0 & 1 \end{bmatrix}$

Matriz de Escalamento: $S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Matriz de Rotação: $R(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

• O produto de matrizes é associativo, mas não comutativo

Como rodar relativamente a um ponto?

$$M = T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1)$$

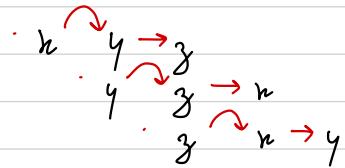
1. $T(-x_1, -y_1)$
2. $R(\theta)$
3. $T(x_1, y_1)$

Como refletir em relação a um eixo?

- $x \rightarrow S(1, -1)$
- $y \rightarrow S(-1, 1)$
- $y = x \rightarrow R(45^\circ) \cdot S(1, -1) \cdot R(-45^\circ)$
- $y = -x \rightarrow R(45^\circ) \cdot S(-1, 1) \cdot R(-45^\circ)$

• A matriz correspondente à transformação inversa existe: $M \cdot M^{-1} = I$

Regra da Mão Direita:



Matriz de Translação: $T(\Delta x, \Delta y, \Delta z) = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Matriz de Escalamento: $S(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Matriz de Rotação: $R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$\cdot R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$\cdot R_y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$



Modelos de Iluminação Locais

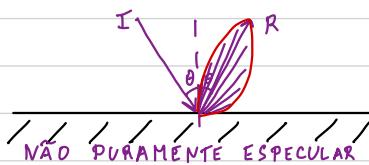
Reflexão Difusa: a luz reflete em todas as direções, com igual valor de intensidade



Reflexão Especular: fontes pontuais de luz produzem zonas sobre-iluminadas na superfície especular

PURAMENTE E.

Reflexão Especular + Difusa



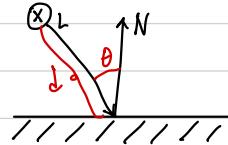
Iluminação Ambiente: $I = k_a I_a$

I: intensidade observada

k_a : coeficiente de reflexão ambiente (difusa) da face - $0 \leq k_a \leq 1$

I_a : constante em todas as direções

Reflexão Difusa: $I_d = k_d \frac{I_f}{d^2} \cos(\theta), \quad \cos(\theta) = N \cdot L$, $0 \leq \theta \leq \pi/2$



θ : ângulo de incidência da fonte de luz

N: normal à superfície (vetor unitário)

L: direção do raio de iluminação (raio incidente)

I_f : intensidade da fonte de luz

k_d : coeficiente de reflexão difusa

LEI DE LAMBERT

Atenuação Quadrática: $I = k_d \frac{I_f}{d^2} \cos(\theta)$

Atenuação Linear: $I = k_d \frac{I_f}{d} \cos(\theta)$

Nenhuma Atenuação: $I = k_d I_f \cos(\theta)$

Reflexão Especular / Modelo de Phong: $I_s = k_s \frac{I_f}{d^2} \cos^n(\alpha)$



R: direção de reflexão máxima

α: ângulo entre R e a direção do observador V

• A reflexão especular depende da posição do observador

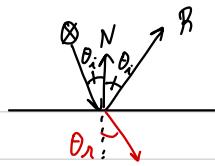


• Yendo V e R vetores unitários, $I_s = k_s \frac{I_f}{d^2} (V \cdot R)^n$

Modelo Elementar de Iluminação: $I = k_a |a| + f_l I_f (k_d \cos(\theta) + k_s \cos^n(\alpha))$

$$f_{ATT} = \begin{cases} 1/d^2, & \text{atenuação quadrática} \\ 1/d, & \text{atenuação linear} \\ 1, & \text{ausência de atenuação} \end{cases}$$

$$I = k_a |a| + f_l I_f (k_d \cos(\theta) + k_s \cos^n(\alpha))$$



Refração: $\text{ren}(\theta_r) = \frac{n_i}{n_r} \text{ ren}(\theta_i)$

n_i : índice de refração do ar

n_r : índice de refração do material

↳ razão entre a velocidade da luz no vazio e no material

$$\cos \theta = \vec{N} \cdot \vec{L}, \quad \vec{L} = \frac{\vec{P}_{\text{Luz}} - \vec{P}}{|\vec{P}_{\text{Luz}} - \vec{P}|}$$

$$\cos \alpha = \vec{R} \cdot \vec{V}, \quad \vec{V} = \frac{\vec{O}_{\text{obs}} - \vec{P}}{|\vec{O}_{\text{obs}} - \vec{P}|}$$

Como calcular \vec{R} ? $\vec{L} + \vec{R} = \vec{N} \cdot 2 |\vec{R}| \cos(\theta) \Leftrightarrow \vec{R} = 2 \cdot \vec{N} \cdot (\vec{N} \cdot \vec{L}) - \vec{L}$

↓ SIMPLIFICAÇÃO

$$R \cdot V \approx N \cdot H \rightarrow I = k_a l_a + k_d l_d (N \cdot L) + k_s l_s (N \cdot H)^n, \quad H = \frac{L + V}{|L + V|}$$

Fator de Atenuação: $f_{ATT} = \min \left(1, \frac{1}{k_c + k_d d + k_s d^2} \right)$

Cor: $I_2 = k_a I_{a2} D_{a2} + f_{ATT} I_{d2} \left(k_d D_{d2} \frac{\cos(\theta)}{NL} + k_s D_{s2} \frac{\cos^n(\alpha)}{RV} \right)$

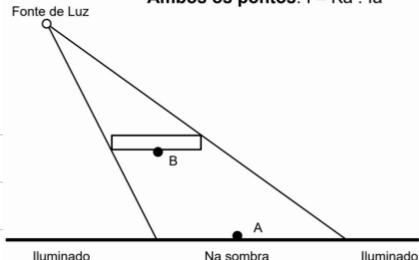
$$D_x = \begin{cases} D_R & - \text{vermelho} \\ D_G & - \text{verde} \\ D_B & - \text{azul} \end{cases}$$

λ : componente de cor

Atenuação da Iluminação: distância fonte-objeto

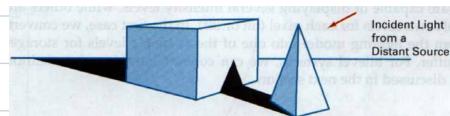
Atenuação Atmosférica: distância observador-objeto

Sombras



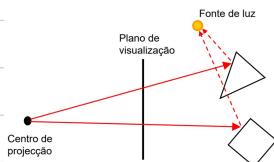
1. Atheron & Veiller:

- considerar a posição da fonte de luz como sendo a posição do observador e determinar as partes visíveis/iluminadas
- determinar as partes visíveis em relação ao observador



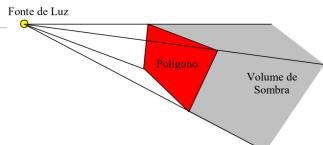
2. Ray Casting:

- emitiu um raio luminoso a partir do ponto de observação, através do centro de um pixel para "dentro" da cena - o ponto de interseção entre o raio e o objeto mais próximo define o objeto visível nesse pixel
- emitiu novo raio a partir do ponto de interseção para a fonte de luz - se interceptar algum objeto, então esse ponto está na sombra



3. Volumes de Sombras:

- partindo de uma fonte de luz, criar uma pirâmide de sombra por cada polígono encontrado em cena
- qualquer objeto ou parte dele que ali se encontre é declarado como estando em sombra



Sombreamento & Texturas

Olefivo: calcular a cor de cada ponto das suas superfícies visíveis

Fogia-Bruta: calcular a normal em cada ponto e aplica o modelo de iluminação pretendido

Sombreamento Constante (Shading): a cor calculada para um ponto do polígono é replicada para todos os pontos restantes do mesmo polígono

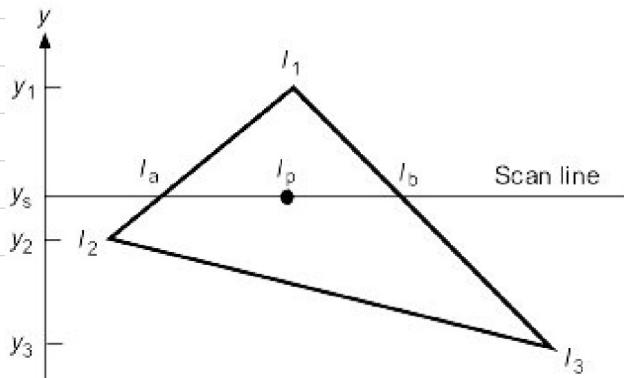
Condições:

1. fonte de luz no infinito \rightarrow NL constante em qualquer ponto do polígono
2. observador no infinito \rightarrow RV constante em qualquer ponto do polígono
3. a face é a própria superfície plana a modelar e não a aproximação de uma superfície curva

Sombreamento Interpolado (Smooth Shading): determinam a cor de um ponto por interpolação da cor definida nos vértices do polígono

Método de Gouraud:

1. Calcular a cor de cada vértice através do modelo de iluminação pretendido
2. Calcular a cor dos restantes pontos do polígono por interpolação bi-linear



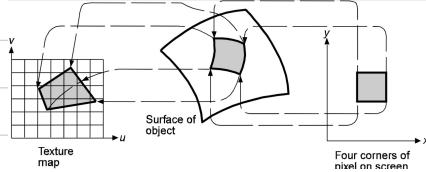
$$\left. \begin{aligned} I_a &= I_1 - (I_1 - I_2) \frac{y_1 - y_s}{y_1 - y_2} \\ I_b &= I_1 - (I_1 - I_3) \frac{y_1 - y_s}{y_1 - y_3} \\ I_p &= I_b - (I_b - I_a) \frac{x_b - x_p}{x_b - x_a} \end{aligned} \right\} \text{ARESTAS}$$

- Método de Phong: efetua a interpolação das normais em vez da cor
1. Para cada vértice da malha poligonal, calcular o vetor normal à superfície
 2.
 - a. As normais nas arestas são calculadas por interpolação linear das normais nos vértices
 - b. As normais ao longo dos pontos de uma linha de varredura obtêm-se por interpolação linear das normais nas arestas
 3. O modelo de iluminação local é aplicado em cada ponto
- O cálculo da iluminação em cada pixel exige o mapeamento inverso para coordenadas do objeto depois de determinada a normal: $C_{\text{obj}} = f(C_{\text{Ecrã}})$

Texturas: permitem obter detalhe visual sem aumentar o detalhe geométrico

Texel: pixel da imagem da textura

1. Mapeamento de Texturas (2D): a textura tem coordenadas normalizadas: $\begin{bmatrix} u, v \\ 0, 1 \end{bmatrix}$
- a. 4 cantos do pixel não mapeados na superfície (s, t)
 - b. 4 pontos (s, t) não mapeados no espaço da textura (u, v) — a cor resultante é extraída das cores dos texels incluídos na área resultante



2. Bump Mapping Textures: simulação de rugosidade sem aumento da geometria
 ↳ afeta-se a direção da normal para o cálculo da iluminação

3. Texturas 3D: evolução contínua no "interior" dos objetos
 ↳ função devolve cor em função das coordenadas espaciais (x, y, z)

Luz & Cor

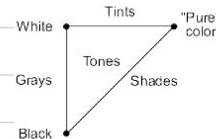
- LUZ**
- 1. Matriz: distingue entre as várias cores
 - 2. Saturação: distância da cor ao círculo de igual intensidade
- COR**
- 3. Intensidade: intensidade refletida
 - 4. Brilho: intensidade emitida

Tintos: pigmento branco + pigmento puro = menor saturação

Sombreados: pigmento preto + pigmento puro = menor intensidade

Tons: pigmento branco e preto + pigmento puro

Cinzentos: pigmento preto + pigmento branco



Espectro Eletrromagnético: vermelho ... violeta

$$c = \lambda f$$

Cromatância: definição da cor, independente da intensidade da luz
 $= \text{Cor} + \text{Saturação}$

Numa fonte de luz branca, todas as frequências estão igualmente presentes

$$\text{Sat} = \frac{E_D - E_W}{E_D}$$

E_D : energia dominante

E_W : energia para o branco

Quanto maior $E_D - E_W$, mais pura será a cor emitida

$E_W = 0$: pura 100%

$E_W = E_D$: pura 0% (branco)

Luminância: área abaixo da curva da energia total emitida

O modelo RGB deve-se às características dos cones do olho humano

$$\text{CINZENTO} = 21\% R + 72\% G + 7\% B$$

Não é possível obter todas as cores pela soma das componentes R, G, B

Modelo CIE: definem-se 3 cores primárias imaginárias X, Y, Z para substituir R, G, B — Y coincide com a função eficiência luminosa

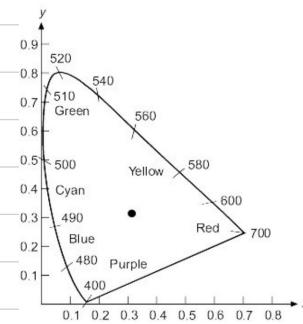
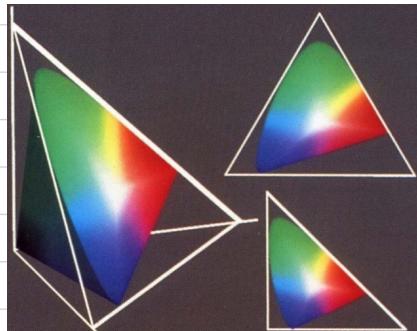
$$\text{Cor } C = xX + yY + zZ, \quad x + y + z = 1, \quad x = X / (X + Y + Z)$$

Os valores da Cromatância (x, y, z) dependem do comprimento de onda dominante e da saturação, mas não da luminância — não normalizados

$$x + y + z$$

x, y e Y definem completamente uma cor: $X = \frac{x}{y} Y, \quad Z = \frac{1-x-y}{y} Y$

Diagrama de Cromatância CIE: projeção do plano $X + Y + Z = 1$ em XY



x e y definem uma cor, nem contam com a luminância

Ponto: várias cores com a mesma cromatância, mas com diferentes luminâncias

Fronteira Curva: cores 100% puras

Centro: cor branca

• Seja C a cor branca

Como determinar o comprimento de onda dominante? Somando duas cores, a cor resultante encontra-se sobre o segmento que une as duas cores

Ex: $A = C + B \rightarrow B$ é o comprimento de onda dominante

Como determinar a pureza da cor? AC/BC é a pureza (%) da cor A

Quanto mais perto estiver A de C, mais luz branca estará incluída em A e menos pura (menos saturada) será essa cor

Cores Complementares: cores que somadas originam a cor branca

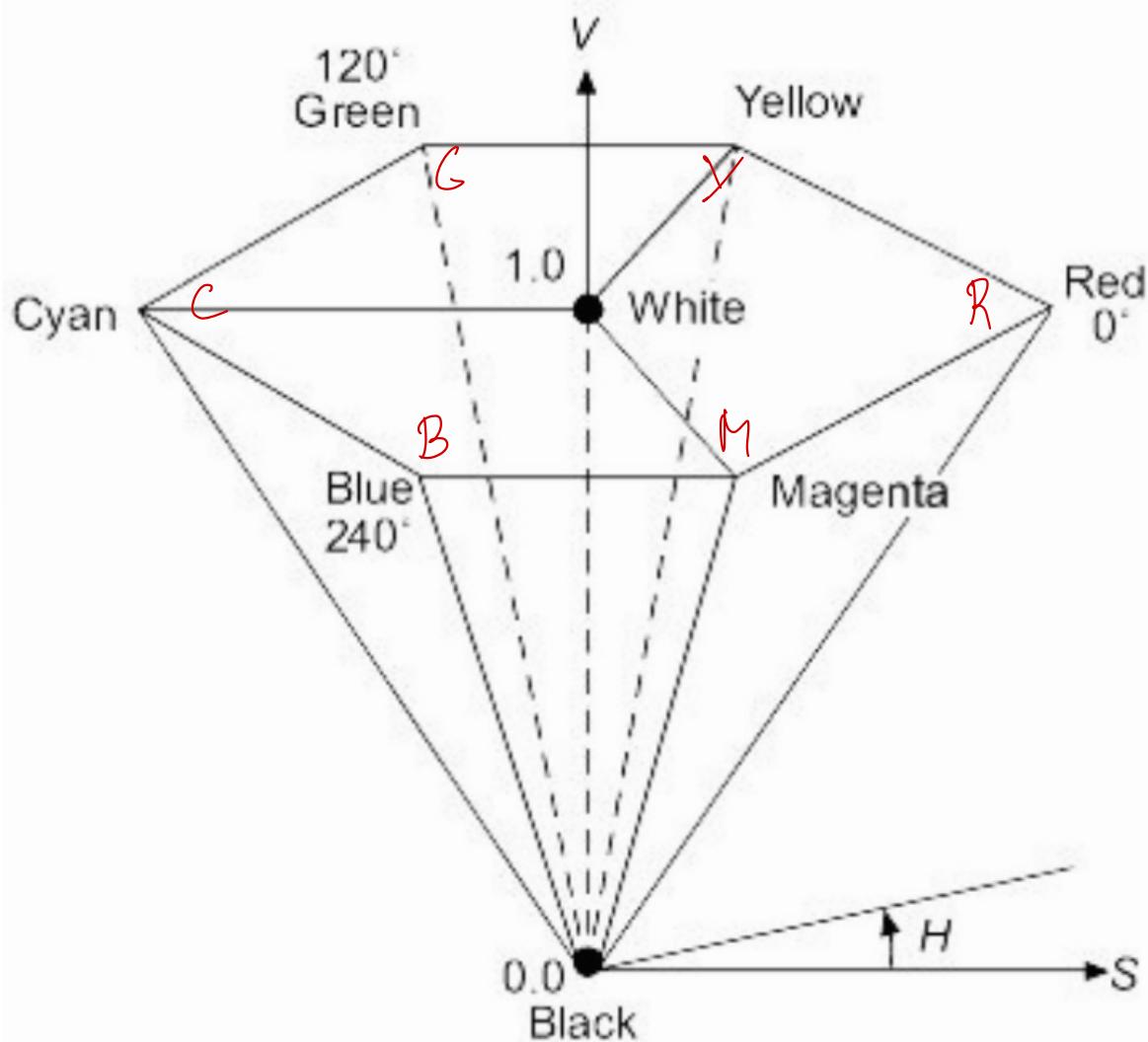
Cores Não Expectais: cores que não podem ser definidas por um comprimento de onda dominante - definido pelo complementar do comprimento de onda

Gama de Cores: as cores entre pontos não obtidos pela mistura das cores desses pontos - nenhum triângulo cobre todas as cores do Diagrama CIE

Modelo RGB: aditivo - no cubo RGB com preto em $(0,0,0)$ e branco em $(1,1,1)$, a diagonal principal representa os níveis de cinzento ($R=G=B$)
$$(R, G, B) + (C, M, Y) = (1, 1, 1)$$

Modelo CMY: subtrativo - as cores não especificadas pelo que é removido / subtraído à luz branca $(0,0,0)$, em vez de somado ao preto $(1,1,1)$, pelo que as cores Ciano, Magenta e Amarelo não são complementares a R, G, B

Modelo HSV: (Matiz, Saturação, Valor) - as cores mais brilhantes estão em $V=1$, o ângulo em torno do eixo vertical faz variar a matiz e a saturação varia de 0 (no centro) a 1 (na periferia do cone)



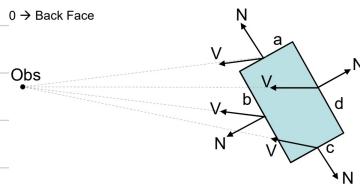
Visibilidade

Algoritmos no Espaço Objeto: comparar os objetos entre si de modo a selecionar a parte visível de cada um (não obstruídas por ele ou por outros objetos)

Algoritmos no Espaço Imagem: para cada pixel da imagem determinar qual o objetivo visível (mais perto do observador, atendendo aos raios de projeção)

"Backface Culling": para reduzir o número de polígonos a processar, só se enviam as "front faces" para o cálculo de visibilidade

$$\vec{N} \cdot \vec{V} = |\vec{N}| |\vec{V}| \cos(\alpha) \begin{cases} > 0 \rightarrow \text{front face} \\ < 0 \rightarrow \text{back face} \end{cases}$$



Determinar Linhas Visíveis: todas as arestas são testadas para produzir uma lista com os seus segmentos visíveis

Ao Volume: supõe-se que uma aresta pode ser oculta pelo volume de um objeto
A Aresta: o teste é efetuado aresta contra aresta, observando que a visibilidade de uma aresta goza de coerência, o que permite determinar a invisibilidade de uma aresta a partir da invisibilidade de outra aresta que posse com ela pontos em comum

Coerência de Aresta: uma aresta só altera a sua visibilidade onde se encontra por trás de uma aresta visível ou quando penetra numa face

Algoritmo de Roberts: cada aresta deve pertencer a uma face de um poliedro convexo - poliedros côncavos devem ser partidos em poliedros convexos

1. Backface Culling

2. Comparar as arestas restantes contra cada volume (poliedro) da cena, podendo ocorrer uma de quatro situações:



Algoritmo de A, L, G. e M.:

1. Backface Culling

2. Calcular a Quantitative Invisibility de um vértice para cada objeto

DI : número de polígonos entre o observador e o próprio ponto

aresta para ser trás de um polígrafo $\rightarrow DI + 1$

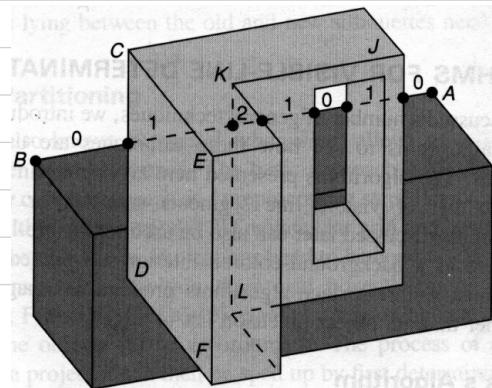
aresta deixa de ser oculta $\rightarrow DI - 1$

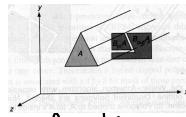
Quando se chega ao vértice final de uma aresta, o valor DI desse vértice é o valor inicial DI nas arestas que se iniciam nesse vértice

Contour Line: aresta partilhada por um polígono back-facing com outro front-facing ou um polígono front-facing isolado

O DI é alterado quando a aresta para ser trás de uma contour line

$DI = 0$: vértice visível





Algoritmo de A&W: requer um algoritmo capaz de efetuar clipping de um polígono côncavo com buracos contra outro polígono qualquer.

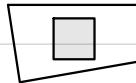
1. Ordenar os polígonos pela coordenada Z — o mais próximo do observador é selecionado como janela de corte
2. O polígono é usado para efetuar clipping dos restantes, resultando duas listas contendo os polígonos (ou parte deles) que são inteiros (inteiros) e exteiros (exteiros) à região de clipping
3. Os polígonos da lista inteiros são marcados como invisíveis
4. Passar aos polígonos seguinte da lista de exteiros e regressar a 2

Determinar Áreas Visíveis: 3 tipos de algoritmos

Cocriência de Área: um grupo de pixels adjacentes é habitualmente coextenso pela mesma face visível

Algoritmo de Warnock: orientado à área

1. Dividir sucessivamente a imagem projetada em 4 blocos retangulares de áreas iguais
2. Se uma área é considerada coerente, então é desenhada com os polígonos que contém; Senão, o procedimento é aplicado recursivamente



Cocriência Vertical: o conjunto de objetos visíveis determinado para uma linha de varrimento difere pouco do conjunto correspondente da linha anterior

¹ Algoritmo de Linha de Varrimento: orientado à linha

Tabela de Arestas: guarda informação de todas as arestas cuja projeção no plano de visualização não é horizontal - entradas por y crescente (coordenadas (x, z) ; altura $y_1 - y_0$ ou y_{MAX} ; incrementos $(\frac{\Delta x}{\Delta y}, \frac{\Delta z}{\Delta y})$; polígonos)

Tabela de Polígonos: guarda informação de todos os polígonos (coeficientes da equação do plano; cor; coordenada Z ; flag "in-out" = False)

Lista de Arestas Ativas: arestas ativas na linha de varrimento atual

Quando se verifica a sobreposição de polígonos, esses polígonos têm a flag "in-out" = True → Utilizando a equação do plano de cada polígono, determina-se a coordenada Z de cada um para saber qual o mais perto do observador

¹ Algoritmo Z-buffer: orientado ao pixel

1. Preencher com zeros o Z-Buffer e o frame buffer com a cor de fundo
- O maior valor de Z será o correspondente ao plano frontal de clipping
2. Percorrer cada polígono, por qualquer ordem
3. Se o ponto (x, y) do polígono atual estiver mais perto do observador do que o ponto atual do Z-Buffer, então o atual substitui o anterior

¹ Algoritmo Ray-Casting: orientado ao pixel - a superfície visível em cada pixel da imagem é determinada fazendo um raio de luz imaginário a partir do centro de projeção (observador), passando pelo centro do pixel para a cena. A cor em cada pixel é definida pela cor, no ponto de intersecção, do objeto interceptado mais próximo

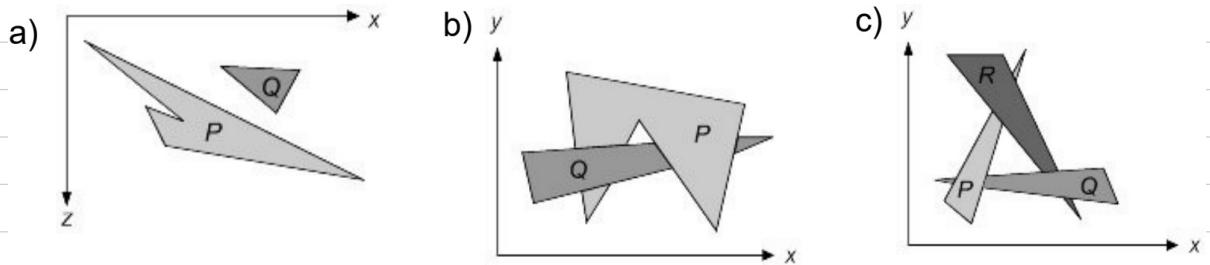
Algoritmos de Lista de Prioridades: começam no espaço objeto e acabam no espaço imagem

Oobjetivo: determinar a ordem de visibilidade para os objetos/polygones, assegurando, assim, que a imagem será corretamente criada se os objetos forem desenhados por cima da ordem.

1. Pintar as faces mais afastadas do observador
2. à medida que outras, mais próximas, vão sendo pintadas, occultam as anteriores

Algoritmo de Newell & Sancha: pintar os polygones por ordem decrescente da distância ao observador

1. Ordenar os polygones por ordem crescente de Z
2. Resolver qualquer ambiguidade na ordenação, nomeadamente se houver sobreposição de polygones na coordenada Z (pode ser necessário dividir polygones)
3. Pintar os polygones por ordem do mais afastado para o mais próximo



Pri-Processar: ordenar os polygones pela coordenada Z do vértice mais afastado

Processar: para o último polygone P da lista, verificar se existe algum polygone Q cujo maior Z seja mais afastado do que o menor Z de P e que esteja a ser obstaculo para P - se não existir, então P pode ser desenhado

Iluminação Global

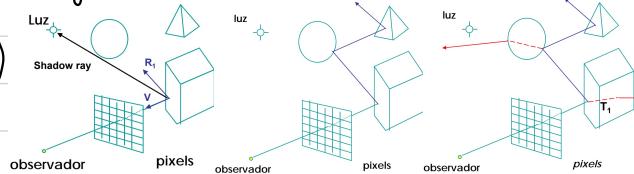
Iluminação Local: com ← fonte de luz

Iluminação Global: com ← fonte de luz + reflexões

Ray Tracing: depende da posição do observador

- O plano de visualização é discretizado em pontos de amostragem
- Faz-se passar, por cada ponto de amostragem, um raio luminoso que parte do observador em direção ao interior da cena
- O rastro de cada raio vai permitir somar as contribuições de reflexão entre faces próximas
- A intersecção do raio refletido com os restantes objetos é registada para obter as contribuições destes na iluminação do ponto
- Se os objetos forem (semi) transparentes é necessário considerar os raios transmitidos para o interior/exterior do objeto

$$\rightarrow I_{\text{inicial}} = I_{\text{local}} = k_a I_a + k_d I_f \text{ (N.L)}$$
$$\rightarrow I = I_{\text{local}} + k_r I_{\text{refletida}} + k_t I_{\text{transmitida}}$$



• Para cada pixel constrói-se uma árvore de intersecções - a cor final do pixel determina-se percorrendo a árvore das folhas para a raiz e calculando as contribuições de cada ramo de acordo com o modelo de reflexão

• sombras, reflexões e refrações fáceis; simula bem iluminação especular

• custo do cálculo de intersecções; não simula bem iluminação difusa

Como processar menos raios? item buffers; adaptative depth control; light buffers

Como processar menos intersecções? (organização hierárquica dos) volumes envolventes; divisão espacial em grelhas tridimensionais

Radiosity: não depende da posição do observador
→ só efetua o cálculo da iluminação, trabalhando no espaço objeto

1. Modela as interações de luz entre objetos e fontes de luz, sem considerar a posição do observador
2. Cria a imagem considerando o observador e efetua cálculo de visibilidade

- Considera-se que todas as superfícies podem produzir/emitir luz
- Assume-se que os processos de emissão e reflexão são difusos ideais
- Necesitam-se das faces discretizadas em patches para garantir que a radiosidade se mantém constante na área correspondente a um patch

Radiosidade: energia expelida de um patch, por unidade de tempo e de área

$$B_i = E_i + \rho_i \sum_j (F_{j-i} B_j A_j / A_i)$$

B_i : radiosidade, energia expelida do patch i em W/m^2

E_i : emissão de luz (auto-emisiva) pelo patch i

ρ_i : reflectividade, percentagem da energia incidente que é refletida pelo patch i

F_{j-i} : fator de forma, percentagem de energia que abandona o patch j e atinge i

Em ambientes difusos: $A_i F_{i-j} = A_j F_{j-i}$ 

$$B_i = E_i + \rho_i \sum_j B_j F_{j-i}$$

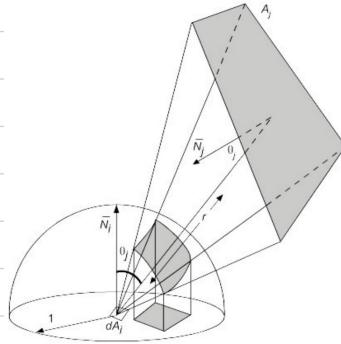
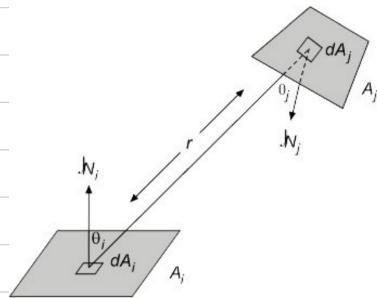
Criação da Imagem:

1. Obter a radiosidade para cada patch
2. Definir a posição do observador
3. Aplicar um algoritmo de visibilidade
4. Calcular a radiosidade dos vértices de cada polígono
5. Aplicar a interpolação de cor

Fator de Forma: F_{ij} representa a fração/percentagem da energia total expelida pelo patch i que atinge o patch j , tomando em consideração a forma, orientação relativa e distância entre ambos os patches, bem como os obstáculos que obstruem o caminho.

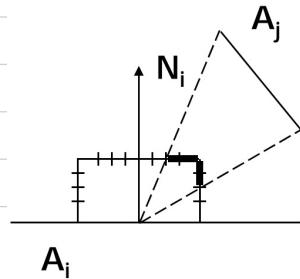
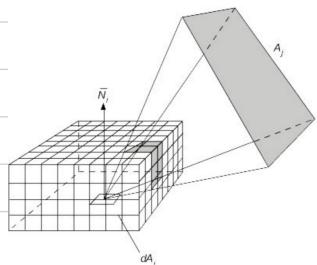
Como calcular?

$$F_{ij} = \frac{1}{A_i} \iint \frac{\cos(\theta_i) \cos(\theta_j)}{r^2} H_{ij} dA_j A_i$$



Como simplificar? $F_{ij} = \sum F_g$

1. Projetar A_{ij} no hemisfério, registrando os quadrados que não cobrem.
2. Para cada quadrado, registrar quais são os patches visíveis A_j e a sua distância.
3. Guardar apenas a patch mais próxima, porque as outras serão invisíveis.
4. Calcular fatores de forma elementares F_g para cada quadrado q do hemisfério.
5. Somar todas as contribuições das células cobertas pelo patch j .



Curvas e Superfícies

Representação de Superfícies: permitem descrever objetos através das faces

Malha Poligonal: coleção de arestas, vértices e polígonos interligados de modo que cada aresta é apenas ligada por um ou dois polígonos

- Uma aresta liga dois vértices
- Um polígono é definido por uma sequência fechada de arestas
- Uma aresta é ligada a um ou dois polígonos (adjacentes)
- Um vértice é partilhado por pelo menos duas arestas
- Todas as arestas fazem parte de algum polígono

1. Representação Explícita: cada polígono é representado por uma lista de coordenadas dos vértices que o constituem — uma aresta é definida por dois vértices consecutivos, fechando o polígono

2. Representação por Apontadores para Lista de Vértices: cada polígono é representado por uma lista de índices/apontadores para uma lista de vértices

3. Representação por Apontadores para Lista de Areias: cada polígono é representado por uma lista de apontadores para uma lista de arestas, na qual cada aresta aparece numa única vez — por sua vez, cada aresta aponta para os dois vértices que a definem e guarda também quais os polígonos a que pertence $E = (V_1, V_2, P_1, P_2)$

4. Solução de Baumgart: cada vértice tem um apontador para uma das arestas que incide nesse vértice e cada aresta apresenta um apontador para a próxima aresta que incide nesse vértice

Curvas Cílicas

- A representação por linha poligonal é uma aproximação de primeira ordem
- Geralmente, utilizam-se polinómios de grau 3

Forma Paramétrica: $x = f_x(t)$, $y = f_y(t)$

Explícita: $y = f(x)$

Implicita: $f(x, y) = 0$

Forma Geral

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases}, \quad 0 \leq t \leq 1$$

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = \underbrace{\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}}_T \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} = T C$$

Continuidade: junção num ponto
Suavidade: mesmo declive na junção

A continuidade e a suavidade não garantidas fazendo coincidir as derivadas/tangentes das curvas nos pontos de junção

$$\frac{\partial Q(t)}{\partial t} = \begin{pmatrix} \frac{\partial x(t)}{\partial t} & \frac{\partial y(t)}{\partial t} & \frac{\partial z(t)}{\partial t} \end{pmatrix} = \frac{\partial (CT)}{\partial t} = C \frac{\partial T}{\partial t} = C \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}$$

G: continuidade geométrica

C: continuidade paramétrica

Tipos de Continuidade:

1. G^0 : as curvas só se juntam num ponto
2. G^1 : a direção dos dois vetores tangentes é igual
3. C^1 : os dois vetores tangentes no ponto de junção têm a mesma direção e amplitude (primeira derivada igual)
4. C^n : as curvas têm no ponto de junção todas as derivadas iguais até à ordem n

- C^1 — velocidade contínua de um objeto ao longo da curva
- C^2 — aceleração contínua de um objeto ao longo da curva
- A continuidade paramétrica é mais restritiva do que a continuidade geométrica (p.e. $C^1 \rightarrow G^1$)

Tipos de Curvas:

1. Curvas de Hermite: 2 pontos extremos e os vetores tangentes nesses pontos
2. Curvas de Bézier: 2 pontos extremos e 2 pontos que controlam os vetores tangentes nesses extremos
3. Curvas Splines

$$\text{Curva } Q(t) = [x(t) \ y(t) \ z(t)] = T C = T M G = B G$$

$$T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

Matriz T

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

Matriz de Base

tipo de curva

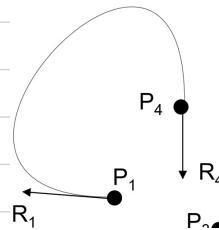
$$G = \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

Vetor Geométrico
geometria da curva

Curvas de Hermite

Vetor Geométrico

$$G = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$



$$\begin{aligned} \cdot Q(t) &= 1^1 M_H G_H = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} M_H G_H \\ \cdot Q'(t) &= 1^1 M_H G_H = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} M_H G_H \end{aligned} \rightarrow \begin{cases} Q(0) = [0\ 0\ 0\ 1] M_H G_H = P_1 \\ Q(1) = [1\ 1\ 1\ 1] M_H G_H = P_4 \\ Q'(0) = [0\ 0\ 1\ 0] M_H G_H = R_1 \\ Q'(1) = [3\ 2\ 1\ 0] M_H G_H = R_4 \end{cases}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = G_H \rightarrow M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

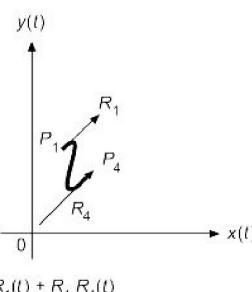
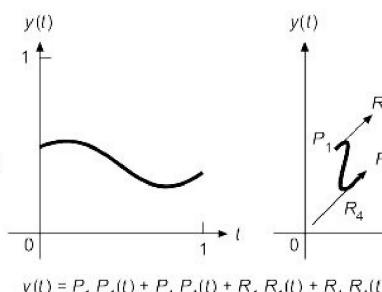
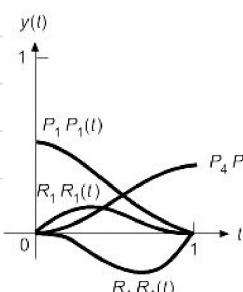
$\underbrace{\hspace{10em}}_{M_H}$

$$Q(t) = 1^1 M_H G_H = B_H G_H = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Continuidade
na Juncão

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} \rightarrow \begin{bmatrix} P_4 \\ P_3 \\ kR_4 \\ R_3 \end{bmatrix}$$

- $G^1 \rightarrow k > 0$
- $C^1 \rightarrow k = 1$



$$y(t) = P_1 P_1(t) + P_4 P_4(t) + R_1 R_1(t) + R_4 R_4(t)$$

Curvas de Bézier

Vector Geométrico

$$G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$



$$\begin{aligned} R_1 &= Q'(0) = 3(P_2 - P_1) \\ R_4 &= Q'(1) = 3(P_4 - P_3) \end{aligned}$$

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$G_H = \underbrace{M_{HB}}_{\text{Matriz de transformação}} \times G_B$$

Uma curva de Bézier está sempre circunscrita ao quadrilátero

$$Q(t) = 1 M_H G_H = 1 M_H M_{HB} G_B = 1 M_B G_B$$

$$M_B = M_H M_{HB} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

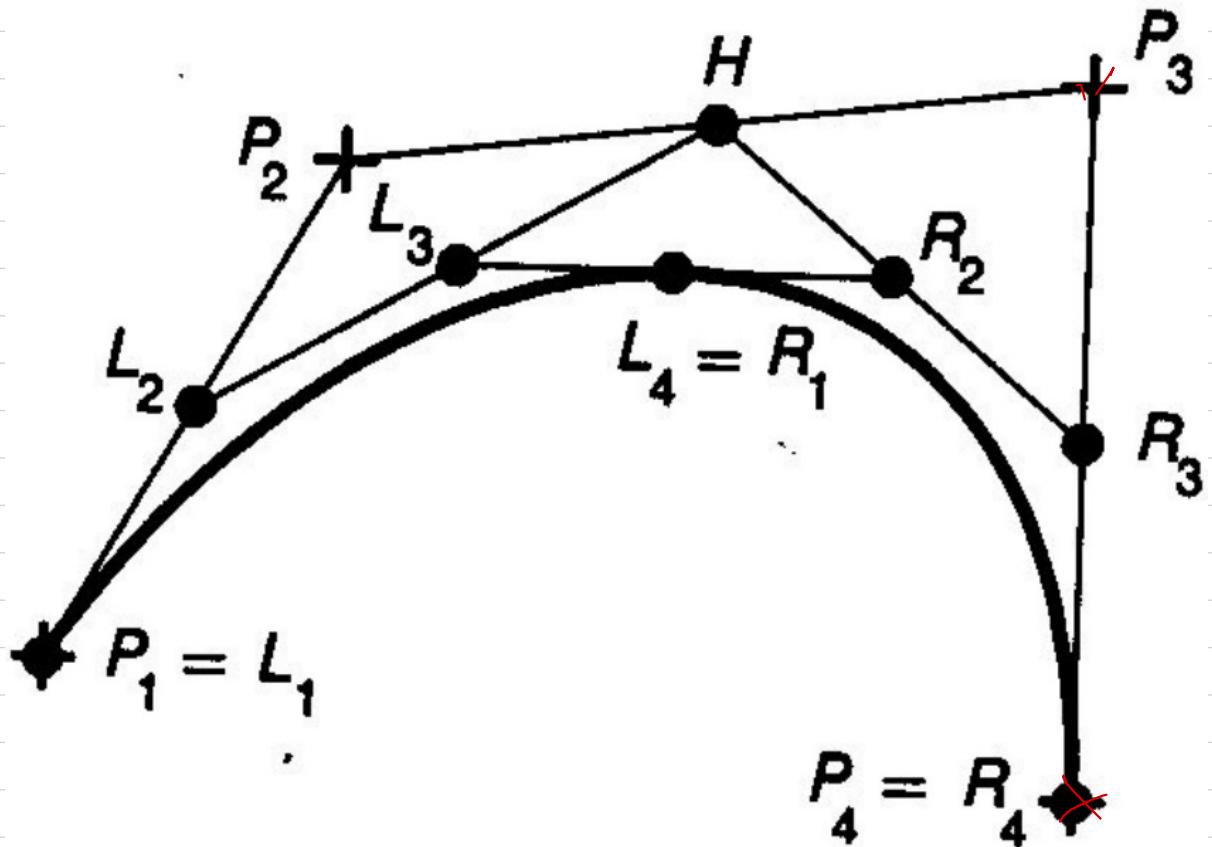
- Como $Q(0) = P_1$ e $Q(1) = P_4$, a curva passa de P_1 para P_4
- A soma de $Q(t)$ é 1 em qualquer ponto
- "Convex Hull" é o polígono quadrilátero que circunscreve a curva

Continuidade C¹: $P_4 - P_3 = k(P_5 - P_4)$, $k > 0$ — P_3, P_4, P_5 colineares

Continuidade C¹: $P_4 - P_3 = P_5 - P_4$ ($k = 1$)

Desenho de Curvas Cúbicas:

1. Força Bruta: avaliar $x(t)$, $y(t)$ e $z(t)$ para valores incrementais de t entre 0 e 1, sendo $f(t) = at^3 + bt^2 + ct + d = ((at+b)t+c)t + d$
2. Algoritmo de Casteljau: efetuar a subdivisão recursiva da curva, parando apenas quando a curva em questão é suficientemente "plana" para poder ser aproximada por um segmento de reta (o seu "convex hull") ou quando os quatro pontos de controlo estão no mesmo pixel



Superfícies Curvas

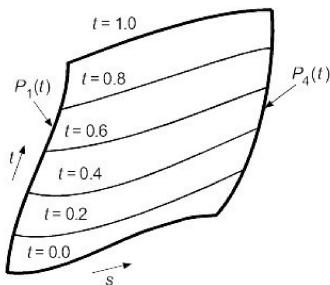
As superfícies cúbicas não são uma generalização das curvas cúbicas

$$Q(t) = 1 \text{ M G} \rightsquigarrow Q(s) = S \text{ M G}$$

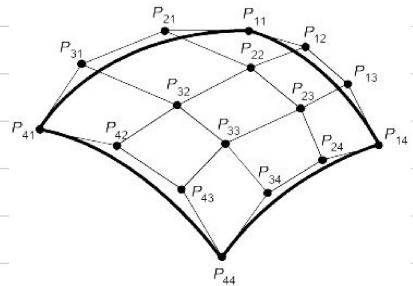
$$Q(s, t) = S \text{ M } G(t) = S \text{ M } \begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix}$$

A matriz geométrica tem 16 pontos de controlo

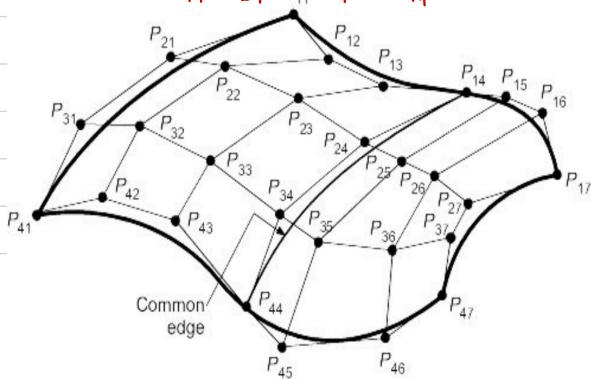
HERMITE



BÉZIER



Continuidade C^0 e G^0 : os pontos de controlo de fronteira devem coincidir



Modelação de Sólidos

Modelação de Sólidos: permite distinguir entre o interior, o exterior e a superfície de um objeto 3D e calcular propriedades dos objetos que dependem dessa distinção

Características/Propriedades:

1. Domínio de representação suficientemente abrangente
2. Não ambígua e única
3. Precisa/Exata
4. Impossibilidade de criar objetos inválidos
5. Fechada
6. Compacta

Operações Booleanas: união; intersecção; diferença

Tipos de Representações:

1. Por Instanciação de Primitivas: conjunto especificado de sólidos 3D úteis
2. Por Varrimento: o deslocamento de um objeto segundo uma trajetória define outro objeto - translação; escalamento; rotação
3. Pela Fronteira: descrição por vértices, arestas e faces da superfície de fronteira (malha poligonal fechada)

2-manifolds: os pontos vizinhos de um qualquer ponto da fronteira estão num disco - cada aresta é partilhada por duas faces

Poliedro: sólido delimitado por um conjunto de polígonos cujas arestas pertencem a dois polígonos (2-manifolds)

Fórmula de Euler: $V - E + F = 2$

↓ CONDIÇÕES ADICIONAIS

1. Cada aresta liga dois vértices e é partilhada por duas faces
2. Pelo menos três arestas encontram-se no mesmo vértice

Generalizações da Fórmula de Euler: $V - E + F - H = 2 (C - G)$

H: número de buracos nas faces

G: número de buracos que atravessam o objeto

C: número de partes do objeto

4. por Decomposição Espacial: decomposição num conjunto de sólidos mais primitivos do que o original, adjacentes e que não se interseccionam

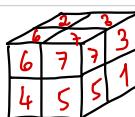
a. Decomposição Celular: existe um conjunto de células primitivas paramétricas

b. Enumeração da Ocupação Espacial: células idênticas e de igual dimensão (voxels) não colocadas numa grelha regular e o objeto é codificado por uma lista única de células ocupadas

c. Dches: o espaço 3D é dividido em octantes com número de nós proporcional à superfície do objeto

d. Árvores Binárias de Partição do Espaço: o espaço é sucessivamente dividido por um plano arbitrário, associado a um nó interno da árvore com dois apontadores (um para dentro e outro para fora do polígono) — se um sub-espaco é homogéneo, deixa de ser dividido

5. Geometria Sólida Construtiva: árvore cujos nós internos não operações booleanas e transformações geométricas e as folhas são primitivas simples



Desenho de Linhas

Segmentos de Reta: $y = mx + b$,

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$m < 1$: x avança sempre 1; y mantém-se ou incrementa-se

$m > 1$: y avança sempre 1; x mantém-se ou incrementa-se

$$\Rightarrow y_{i+1} = m x_{i+1} + b = m(x_i + \Delta x) + b = y_i + m \Delta x \xrightarrow{\Delta x=1} y_{i+1} = y_i + m$$

Algoritmo ($m < 1$):

1. Incrementar x de 1 em cada passo, partindo do ponto mais à esquerda
2. $y_{i+1} = y_i + m$
3. Desenhar o ponto $(x_{i+1}, \text{round}(y_{i+1}))$ — pixel mais próximo da reta

ALGORITMO MIDPOINT

$$(0, 0) \rightarrow (a, b) \rightarrow y = mx \Leftrightarrow y = \frac{b}{a}x \Leftrightarrow f(x, y) = bx - ay = 0$$

$$P = (x_p, y_p)$$

$$Q = (x_p + 1, y_p + \frac{1}{2} + e)$$

$$E = (x_p + 1, y_p)$$

$$M = (x_p + 1, y_p + \frac{1}{2})$$

$$NE = (x_p + 1, y_p + 1)$$

$$e = Q - M$$

$e > 0$: o próximo ponto é NE
 $e < 0$: o próximo ponto é E

$$\left. \begin{array}{l} \\ \end{array} \right\} \text{erro} < \frac{1}{2}$$

$$\begin{aligned} Q \in r &\Leftrightarrow f(Q) = 0 \Leftrightarrow f(x_p + 1, y_p + \frac{1}{2} + e) = 0 \Leftrightarrow b(x_p + 1) - a(y_p + \frac{1}{2} + e) = 0 \\ &\Leftrightarrow b(x_p + 1) - a(y_p + \frac{1}{2}) - ae = 0 \Leftrightarrow b(x_p + 1) - a(y_p + \frac{1}{2}) = ae \Leftrightarrow f(x_p + 1, y_p + \frac{1}{2}) = ae \\ &\Leftrightarrow f(M) = ae \end{aligned}$$

$$a > 0 \rightarrow \text{sign}(e) = \text{sign}(ae) = \text{sign}(f(M)) = \text{sign}(f(x_p+1, y_p + 1/2)) = d_p$$

Genericamente: $d_{i+1} = f(x_{i+1}, y_{i+1} + 1/2)$

$\begin{cases} d_i > 0 \rightarrow \text{movimento para NE} \rightarrow x_{i+1} = x_i + 1, y_{i+1} = y_i + 1 \rightarrow d_{i+1} = d_i + b - a \\ d_i < 0 \rightarrow \text{movimento para E} \rightarrow x_{i+1} = x_i + 1, y_{i+1} = y_i \rightarrow d_{i+1} = d_i + b \end{cases}$

$\cdot d_0 = f(A) = f(x_0 + 1, y_0 + 1/2) = f(0 + 1, 0 + 1/2) = f(1, 1/2) = b - a/2$

Conclusão: multiplicar tudo por 2 para obter valores inteiros

$\rightarrow D_0 = 2(b - a)$

$\rightarrow D_{i+1} = D_i + 2(b - a)$, quando o movimento é para NE

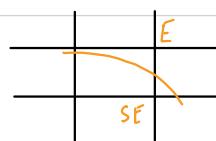
$\rightarrow D_{i+1} = D_i + 2b$, quando o movimento é para E

Circunferências: $d_p = f(x_p + 1, y_p - 1/2) = (x_p + 1)^2 + (y_p - 1/2)^2 - r^2$

$\cdot d_i > 0$: movimento para SE $\rightarrow x_{i+1} = x_i + 1, y_{i+1} = y_i - 1 \rightarrow d_{i+1} = d_i + 2x_i - 2y_i + 5$
 $\cdot d_i < 0$: movimento para E $\rightarrow x_{i+1} = x_i + 1, y_{i+1} = y_i \rightarrow d_{i+1} = d_i + 2x_i + 3$

$\cdot d_0 = f(D+1, R - 1/2) = f(1, R - 1/2) = 1 + (R - 1/2)^2 - R^2 = 1 + R^2 - R + 1/4 - R^2 = 5/4 - R$

⚠ Considerando tudo no 1º octante



Preenchimento de Regiões

Algoritmos de Preenchimento de Regiões:

1. Preenchimento segundo contorno existente

a. Por difusão - "flood fill"

i. Limitado por contorno: problema da fronteira incompleta

ii. Limitado ao interior da região: para substituir uma cor por outra

b. Por análise de contorno - "boundary algorithm"

2. Preenchimento por varredura segundo descrição de contorno - Scan conversion

a. Algoritmo da lista de pontos de fronteira ordenados

b. Algoritmo da lista de arestas ativas

Conectividade 4: -|-

Conectividade 8: *

"Boundary Algorithm": trabalha linha a linha e não sobreengrena a filha

1. Parte do ponto inicial, situado no interior, que começa por ser colocado na filha

2. Se filha vazia, então termina; Senão, retira um ponto da filha

3. A partir desse ponto, preenche na horizontal, primeiro para a direita e depois para a esquerda, até encontrar o contorno, registrando as extremidades X_L e X_R

4. Na linha imediatamente abaixo, procura, entre X_L e X_R , os novos ^{pontos} de fato, colocando-os na filha

5. Repete 4, para a linha imediatamente acima

6. Volta para 2

Algoritmo da Lista de Pontos de fronteira Ordenados: determina os pontos de intersecção das arestas com as linhas de varredura do ecrã e ordena-os

1. Determinar os Pontos de Fronteira
2. Ordenar os Pontos de Fronteira (y, z)
3. Preencher os segmentos horizontais formados por pares de pontos consecutivos

Como melhora? Construir uma lista ordenada de pontos para cada valor de Y

Algoritmo da Lista de Arestas Ativas: só as arestas ativas são usadas no preenchimento de uma linha de varredura

Vértice Simples: a aresta sai da lista na linha anterior $\rightarrow \text{long } Y = Y_2 - Y_1$

Vértice Duplo: a aresta sai da lista na linha seguinte $\rightarrow \text{long } Y = Y_2 - Y_1 + 1$

1. Constituir a Tabela de Arestas — $\text{long } Y$ é a altura da aresta
2. Percorrer as linhas de varredura para construir a Lista de Arestas Ativas, ordenando-a pelos valores de X
3. Agrupar AEL aos pares de arestas, segundo os valores de X
4. Para cada aresta, decrementar o valor de $\text{long } Y$ e atualizar a lista
5. Voltar para 2

A Tabela de Arestas regista as arestas que entram na AEL em cada linha

A análise é efectuada de cima para baixo e da esquerda para a direita

$$\{ X, DX, \text{Long } Y \}$$