UFABC

Computação Gráfica

André Brandão

Aula 11

Pipeline Gráfico

Sumário

- Rasterização
- Operações antes e depois da rasterização
- Antialiasing simples
- Seleção de primitivas para eficiência

Introdução

- No Ray Tracing, cada pixel lança um raio para a cena para verificar qual será a cor final deste dado pixel.
- Na rasterização, por meio de cada objeto presente em uma cena, serão verificados os pixels que serão ocupados por esse objeto.
- A rasterização é um processo que segue uma renderização por ordem de objetos (*object-order rendering*).

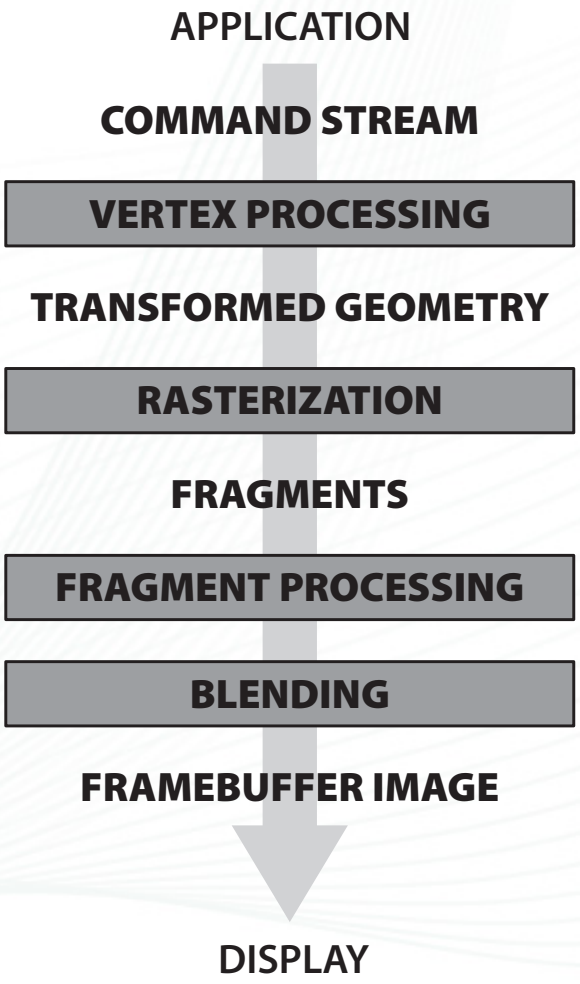
Introdução

- Quando nos referimos ao Pipeline Gráfico, fazemos referência ao processo de rasterização.
- Pode-se classificar o Pipeline Gráfico em:
  - Ligado ao Hardware: voltado a sistemas interativos em tempo real. Envolve conceitos de OpenGL e DirectX.
  - Ligado ao Software: voltado animações de alta qualidade, efeitos visuais em enormes cenas. Tomam muito tempo para a sua execução.

Introdução

- Porém, tanto um processo como o outro tem conceitos compartilhados. Aqui, verificaremos os **conceitos compartilhados entre os dois tipos de Pipeline Gráfico.**
- Assim, divide-se o Pipeline Gráfico em quatro etapas:
  - Processamento de Vértices (*Vertex Processing*)
  - Rasterização
  - Processamento de Fragmentos (*Fragment Processing*)
  - *Blending*

Etapas do Pipeline Gráfico



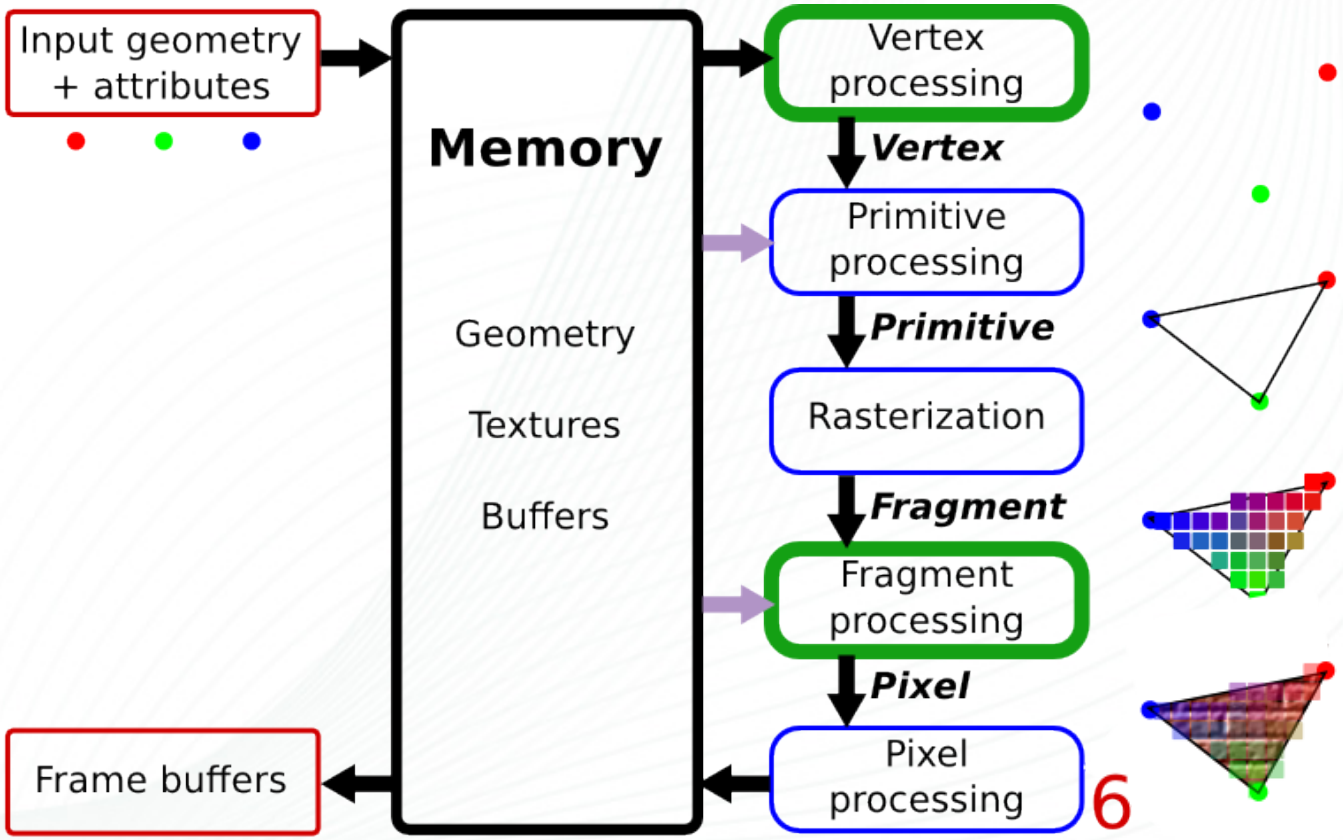
Introdução

- Objetos geométricos são inseridos no Pipeline Gráfico por meio de uma aplicação interativa ou a partir de um arquivo de descrição de cena. De um modo ou de outro, esses objetos são descritos por conjuntos de vértices.
- Os objetos geométricos passam pela etapa de *Vertex Processing* na qual possibilita a identificação das primitivas que compõem a cena.

Introdução

- As primitivas são identificadas na etapa de Rasterização, que possibilita a divisão de cada primitiva em fragmentos, um para cada pixel que forma a imagem.
- Cada fragmento passa a ser dividido pela quantidade de pixels, na etapa de *Fragment Processing.*
- Na etapa de *Blending,* os fragmentos são mapeados para os seus respectivos pixels no plano de projeção.

Etapas do Pipeline Gráfico



http://romain.vergne.free.fr/teaching/IS/imgs03/pipeline-v2.png

Sumário

- **Rasterização**
- Operações antes e depois da rasterização
- Antialiasing simples
- Seleção de primitivas para eficiência

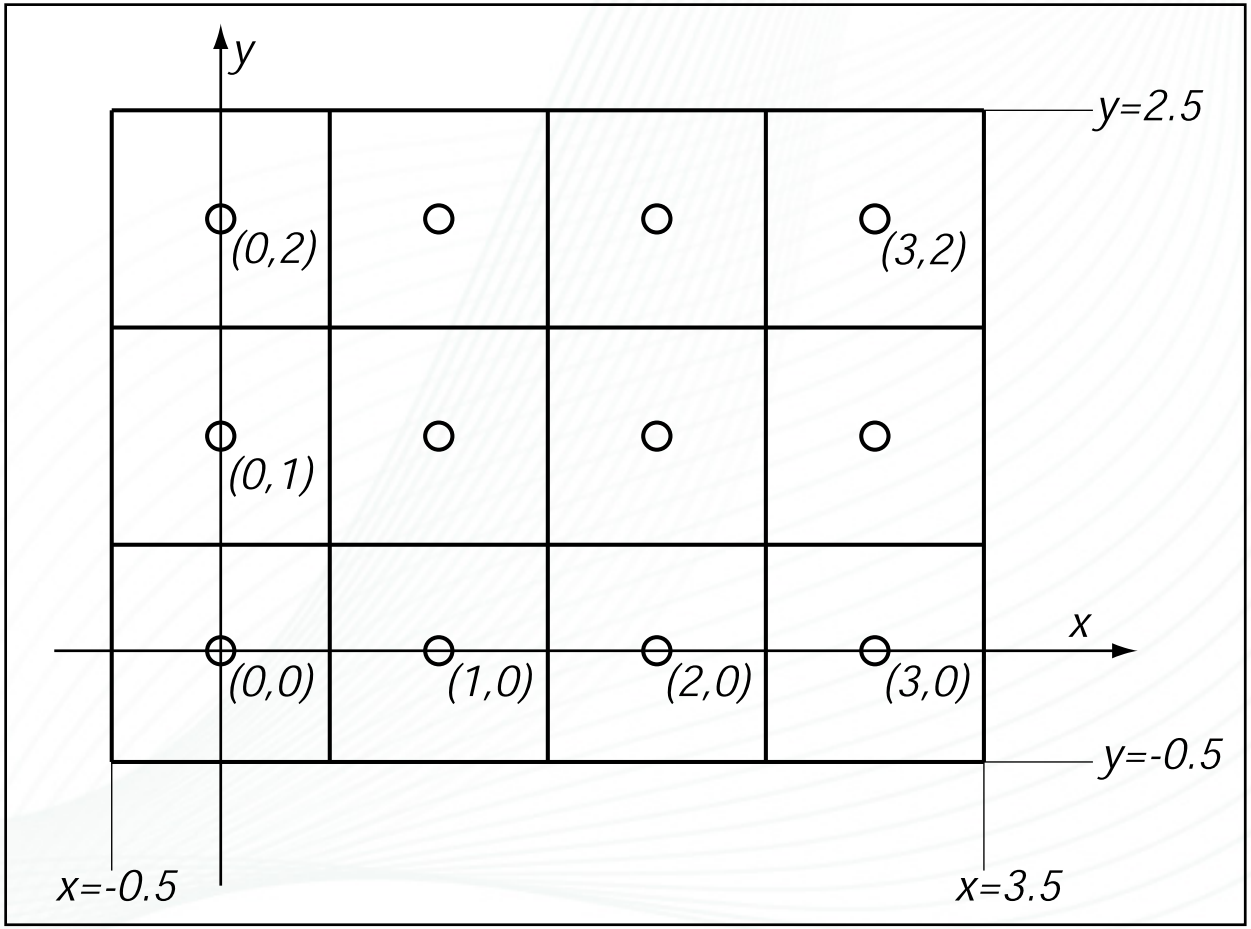
Rasterização

- A Rasterização é a etapa central na renderização por ordem de objetos (*object-order rendering*).
- Nesta etapa, existem duas tarefas: (1) enumerar os pixels que serão cobertos pelas primitivas e (2) interpolar valores nas primitivas.

Desenho de linhas

- Muitas APIs gráficas contêm funções para desenhar linhas, que recebem dois pontos e desenha uma linha entre eles.
- Por exemplo, ao enviar os pontos $(x_1,y_1) = (1, 1)$ e $(x_2,y_2) = (3, 2)$, uma API desenha uma linha entre os dois extremos.
- Um conjunto de pixels entre os dois pontos são preenchidos, de modo a desenhar, da maneira mais razoável possível, uma linha entre os dois pontos enviados à função.

Desenho de linhas



Desenho de linhas

- O desenho de linhas é baseado em equações da reta.
- Temos dois tipos de equações:
  - Equações implícitas
  - Equações paramétricas

Equações implícitas

- A primeira coisa a se fazer é encontrar a equação implícita para a linha no formato:

$$y = mx + b$$

- A equação pode ser encontrada da seguinte forma:

$$f(x,y) = (y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0 = 0$$

Equações implícitas

$$f(x,y) = (y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0 = 0$$

- Assumimos que $x_0 \leq x_1$. Se esse não for o caso, invertemos a ordem dos argumentos.
- O fator m é dado por:

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

- O pressuposto fundamental é que será desenhada uma linha, com menor espessura possível, que não contenha lacunas.
- Uma conexão diagonal que passa por dois pixels não é considerada uma lacuna.

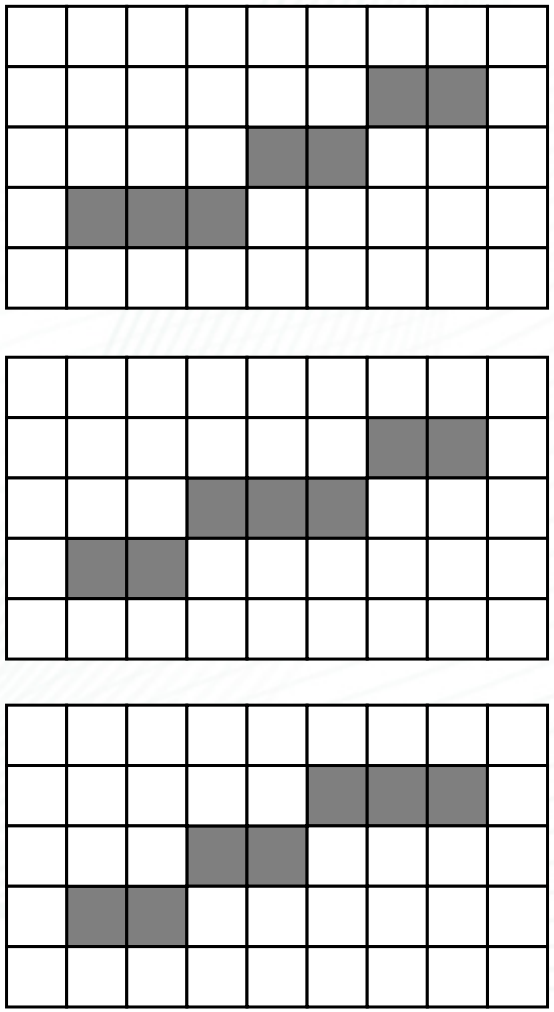
Equações implícitas

- **Vamos considerar o caso em que $m \in (0, 1]$, onde a linha é mais horizontal do que vertical.**
- Quando temos diagonal que passa por dois pixels, desenhamos a linha da esquerda para a direita. Então, teremos duas possibilidades:
  - Desenhar a linha a partir da mesma altura do pixel a esquerda, ou;
  - Desenhar a linha a partir de um pixel acima do pixel mais a esquerda.
- Nos dois casos, sempre haverá um pixel em cada coluna entre os dois extremos.
- Não pintar nenhum pixel, neste caso, resulta em uma lacuna e, se forem pintados dois pixels, teremos uma espessura muito grossa.

Equações implícitas

- Sempre haverá um pixel em cada coluna que separa os extremos. Se houver dois pixels “pintados” entre os extremos, a linha fica com espessura maior do que esperada.
- Porém, podemos “pintar” dois pixels na mesma linha, para o caso que consideramos, que é mais horizontal do que vertical.
- Considere o exemplo dos extremos (1,1) e (7,3). A figura, a seguir, ilustra três soluções possíveis para ilustrar a reta.

Desenho de linhas



Desenho de linhas

- No caso que discutimos, onde $m \in (0, 1]$, estabelecemos o pixel mais à esquerda e o número da coluna (o valor da coordenada x) do pixel mais à direita, então executa-se um processo iterativo no sentido horizontal, afim de estabelecer os valores de y da linha que será desenhada.

Algoritmo para o desenho de linha na equação implícita

$y = y_0$

for $x = x_0$ to x_1 **do**

    draw(x, y)

    if (some condition) **then**

        $y = y + 1$

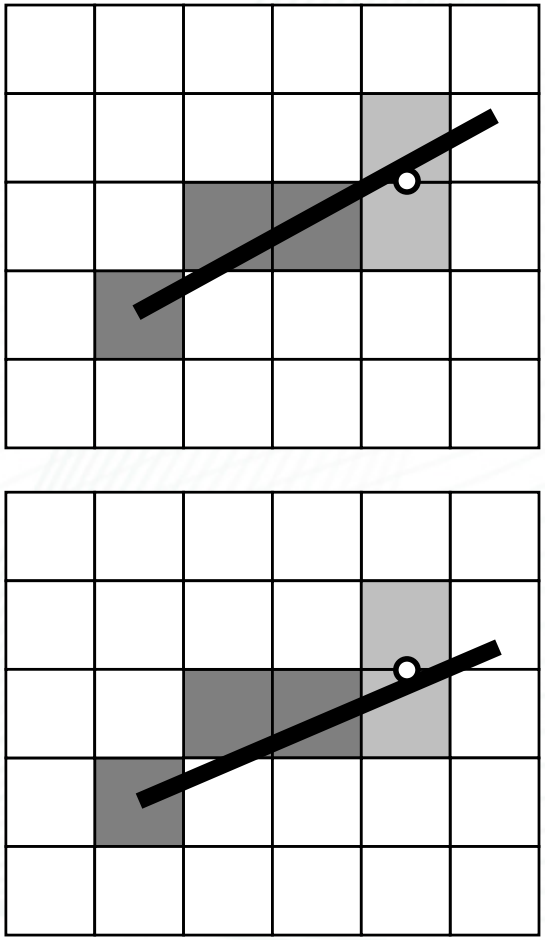
Desenho de linhas

- No algoritmo, poderíamos traduzir, em palavras, para algo como “continhe a pintar pixels, da esquerda para a direita, e, algumas vezes, mova-se para cima, na direção do eixo y, enquanto pinta os pixels”.
- A questão, aqui, é estebalecer maneiras eficientes para definir a decisão presente na condição “if”.

Desenho de linhas

- Uma maneira eficiente de definir a condição “if” é pelo algoritmo do ponto médio entre os pixels candidatos a serem pintados.
- Ao tracejarmos uma reta que liga dois pontos, observamos os pontos candidatos $(x + 1, y)$ e $(x + 1, y + 1)$. O ponto médio entre os dois candidatos é $(x + 1, y + 0.5)$.
- Se a linha passar abaixo do ponto médio, então o pixel pintado é o de menor coordenada y. Caso contrário, será pintado o pixel de maior coordenada y.

Ponto médio



Desenho de linhas

- Para melhor avaliação, deve-se verificar o valor de $f(x+1, y+0.5)$ na equação

$$f(x,y) = (y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0 = 0$$

- Podemos fazer a análise de acordo com o fator de y, que é $(x_1 - x_0)y$. Lembre que o fator de y é considerado positivo, pois $x_1 > x_0$. Assim, a medida que o fator de y cresce, a reta passará acima do ponto médio. Assim…

Algoritmo para o desenho de linha na equação implícita

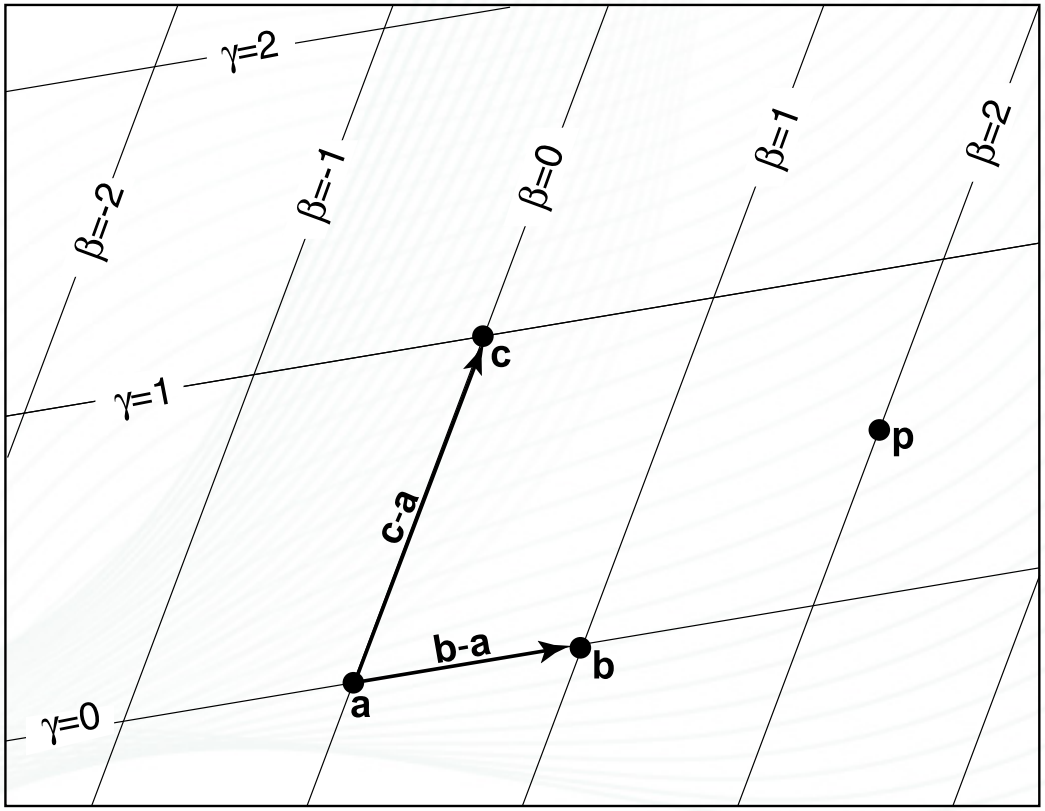
$y = y_0$
$d = f(x_0 + 1, y_0 + 0.5)$
for $x = x_0$ to x_1 **do**
    draw(x, y)
    if $d < 0$ **then**
        $y = y + 1$
        $d = d + (x_1 - x_0) + (y_0 - y_1)$
    else
        $d = d + (y_0 - y_1)$

Rasterização de triângulos

- Triângulos podem ser definidos por pontos e/ou coordenadas dos seus vértices.
- Um triângulo, ao ser definido pelos seus vértices $p_0 = (x_0, y_0), p_1 = (x_1, y_1) e\ p_2 = (x_2, y_2)$ podemos interpolar as cores dos pixels que constituem o triângulo por meio de coordenadas baricêntricas.
- Nas coordenadas baricêntricas, ao levarmos em consideração os vetores que compõem o triângulo, teremos:

Coordenadas baricêntricas

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$



Coordenadas baricêntricas

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

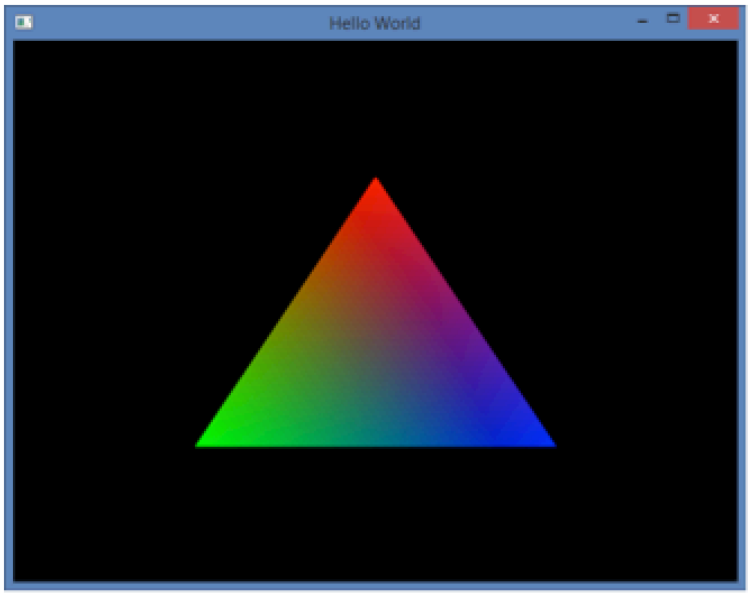
$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

$$\alpha + \beta + \gamma = 1$$

Rasterização de triângulos

- Assim, se os vértices têm cores $c_0, c_1\ e\ c_2$, a cor c em **um** dado ponto do triângulo é:

$$c = \alpha c_0 + \beta c_1 + \gamma c_2$$



Rasterização de triângulos

- Então, para pintar cada ponto que faz parte do triângulo, ao utilizar coordenadas baricêntricas, podemos definir um algoritmo.

Algoritmo simplificado de rasterização de triângulo

for all x **do**
  for all y **do**
    compute (α, β, γ) for (x, y)
    if $(\alpha \in [0, 1]$ and $\beta \in [0, 1]$ and $\gamma \in [0, 1])$ **then**
      $\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$
      drawpixel (x, y) with color $\mathbf{c}$

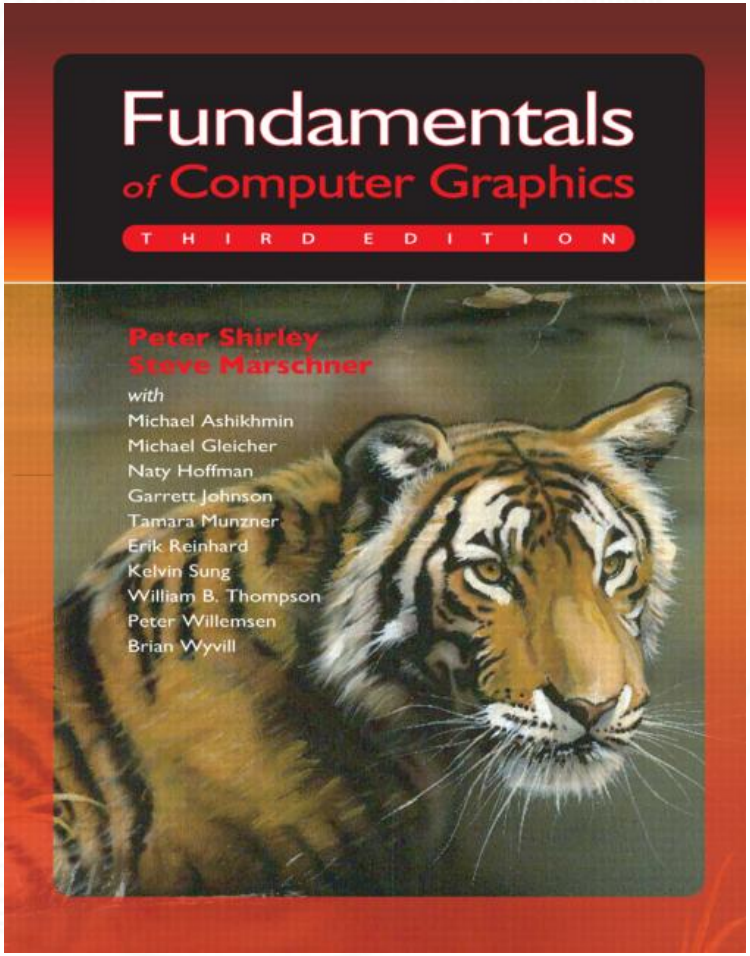
Sumário

- ~~Rasterização~~
- **Operações antes e depois da rasterização**
- Antialiasing simples
- Seleção de primitivas para eficiência

Aula de hoje

Shirley, Peter, Michael Ashikhmin, and Steve Marschner. Fundamentals of computer graphics. CRC Press, 3rd Edition, 2009.

- **Capítulo 8**



Reorganização das aulas

- Devido à paralização realizada no dia 24/10, as próximas aulas serão reorganizadas.
- Em breve, a nova organização das próximas aulas ficará disponível no TIDIA.

Fim da Aula 11

André Luiz Brandão