Computação Gráfica

Renderização em Tempo Real

Erivaldo Xavier de Lima Filho
desadoc@gmail.com

Computação Gráfica

- É a síntese de imagens por computador
- Consiste de vários passos:
  - Modelagem
  - Renderização
  - *Post-processing*
- A renderização pode ser:
  - *Offline*
  - *Real-Time*
- Foco da apresentação: renderização *real-time*

Computação Gráfica

- Aplicações:
  - Jogos
  - Simuladores militares
  - Filmes
  - Visualização Médica e Científica
  - Engenharia, Arquitetura
  - Dentre outros

Renderização *Offline*

- Mais utilizada para mídia não-interativa
- Implementada em CPU
- Alta qualidade
- Longos períodos de renderização
- *Raytracing*

Renderização *Offline*

Final Fantasy XIII

Renderização *Real-Time*

- Em geral, síntese que ocorre em frequência aceitável para o olho humano
- Imersiva
- Baixa qualidade
- Baixo tempo de resposta
- Dependência de hardware especializado
- Rasterização

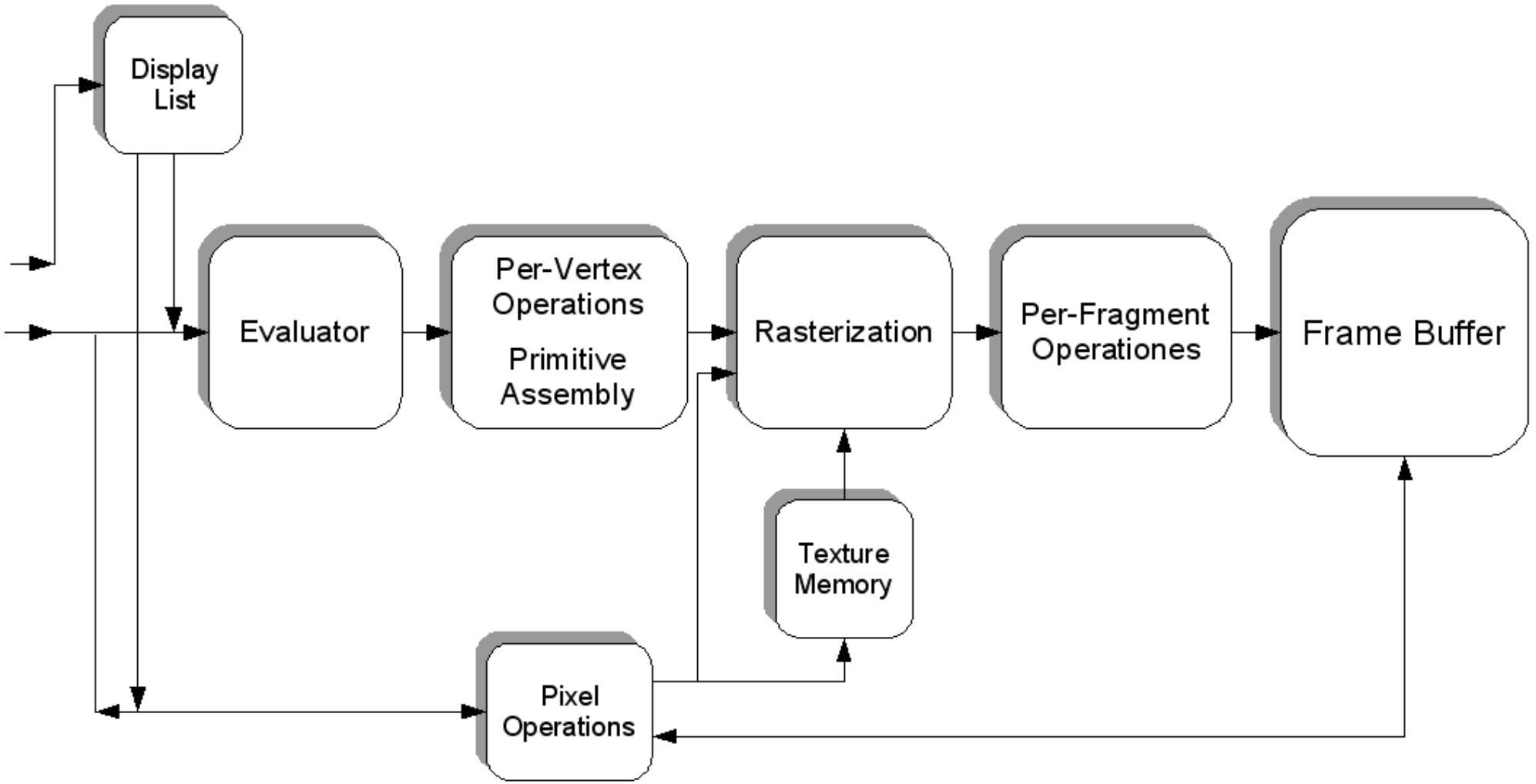
Renderização *Real-Time*

Crysis

Renderização *Real-Time*

A “Mágica” da Renderização *Real-Time*

- *Pipeline* de rasterização



Geometria

- Primitivas são a entrada da *pipeline*:
  - Pontos
  - Retas
  - e Triângulos

- Acompanhadas por muitos parâmetros:
  - Valores em ponto-flutuante
  - Matrizes
  - Texturas

Etapas do *Pipeline*

- Transformação da geometria
- *Culling* e *Clipping*
- Rasterização
- Texturização
- Sombreamento (*Shading)*
- *Depth*, *Scissors* e outros testes
- *Blend*

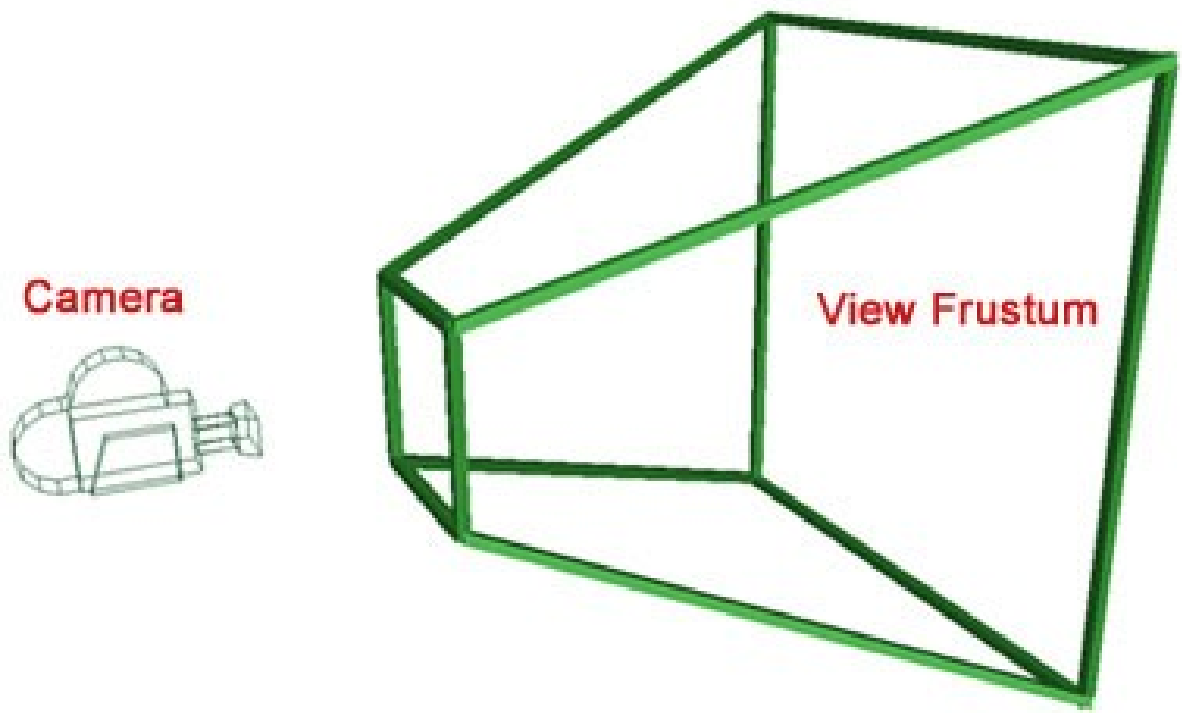
Os *Shaders*

- Vertex *Shader*
  - Transformação da geometria
- *Culling* e *Clipping*
- Rasterização
- Pixel *Shader*
  - Texturização
  - *Shading*
- *Depth*, *Scissor* e outros testes
- *Blend*

Hoje, existem mais tipos de *shader...*

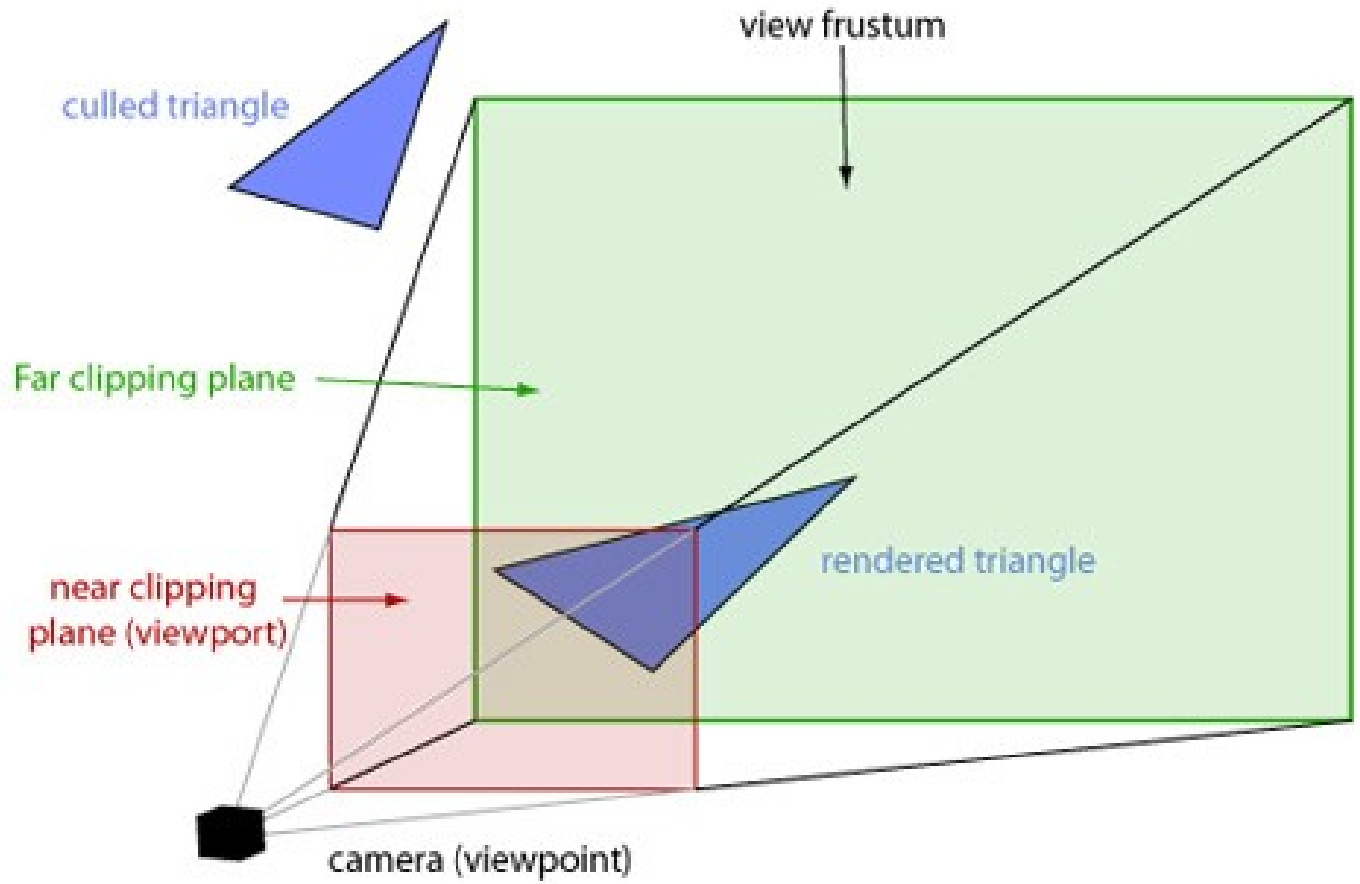
Transformação da Geometria

- A geometria é projetada no espaço de visualização da câmera



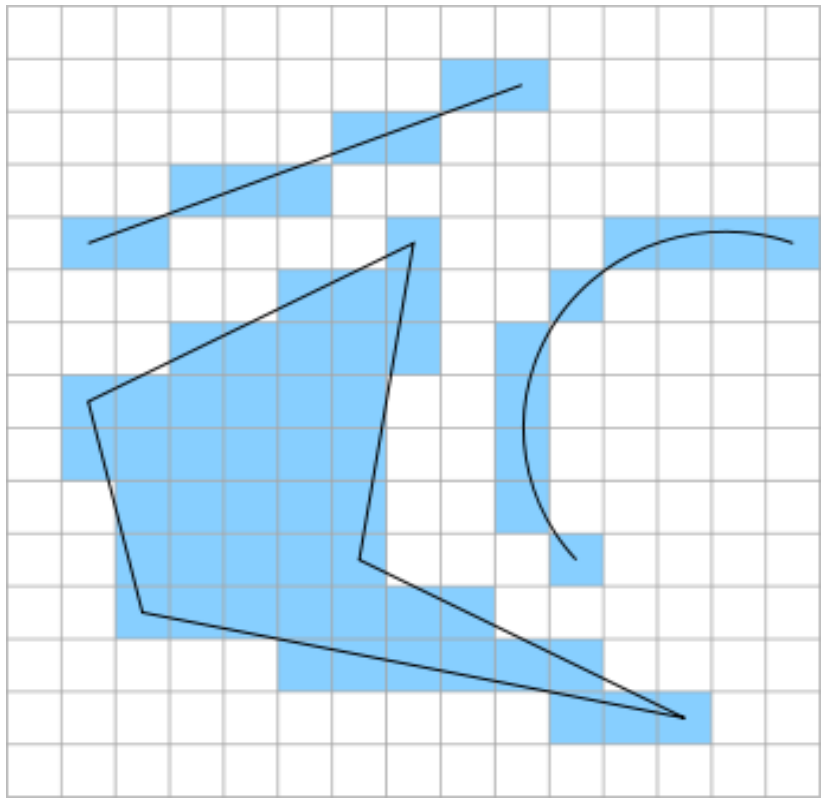
Clipping

- Etapa de eliminação de geometria fora do volume de visualização



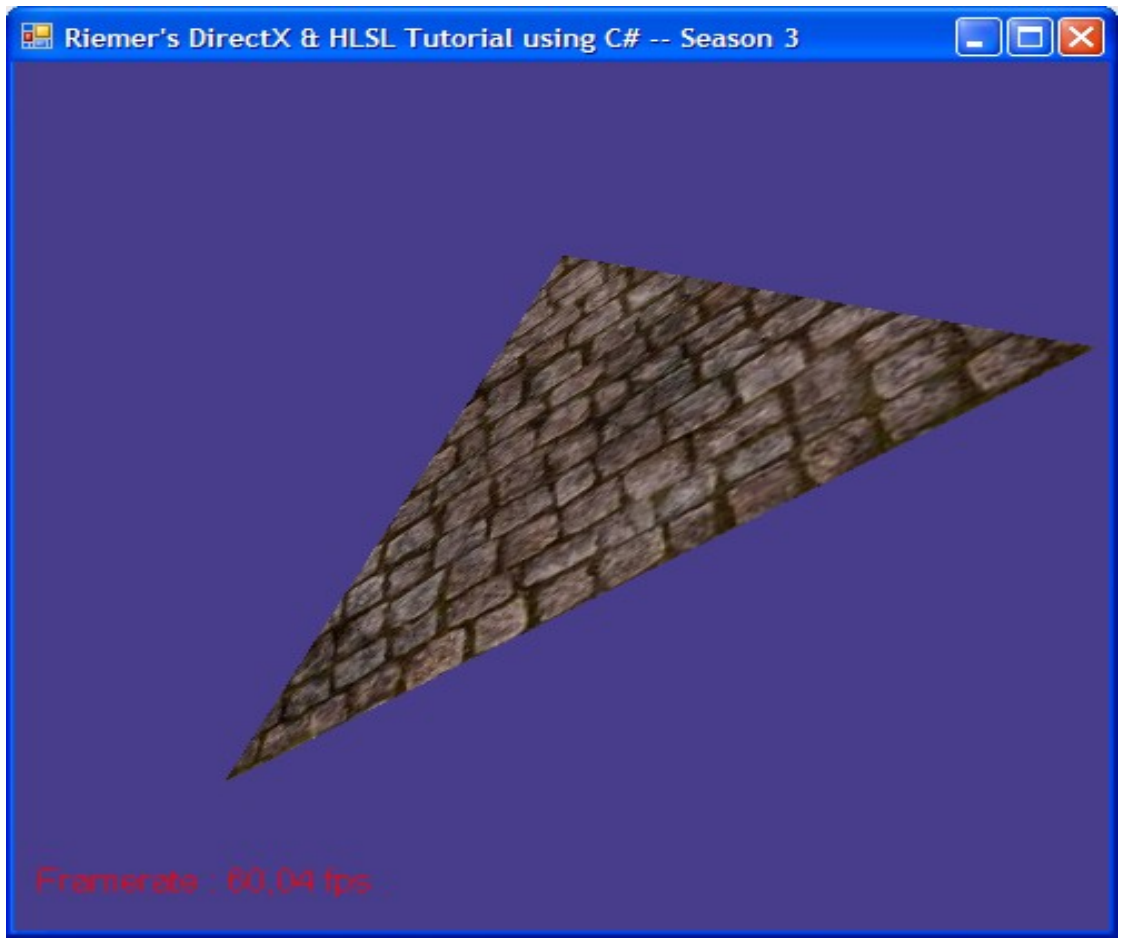
Rasterização

- Fragmentos são criados a partir da geometria
- Etapa importante para anti-*aliasing* da cena



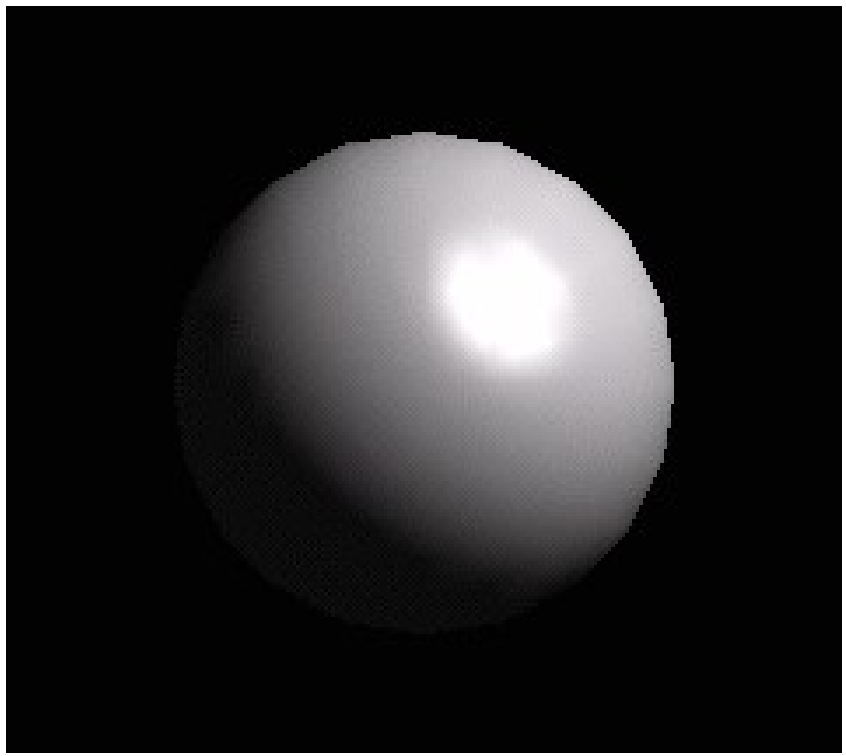
Texturização

- Acesso à memória, usualmente uma imagem, para obter parâmetro do fragmento



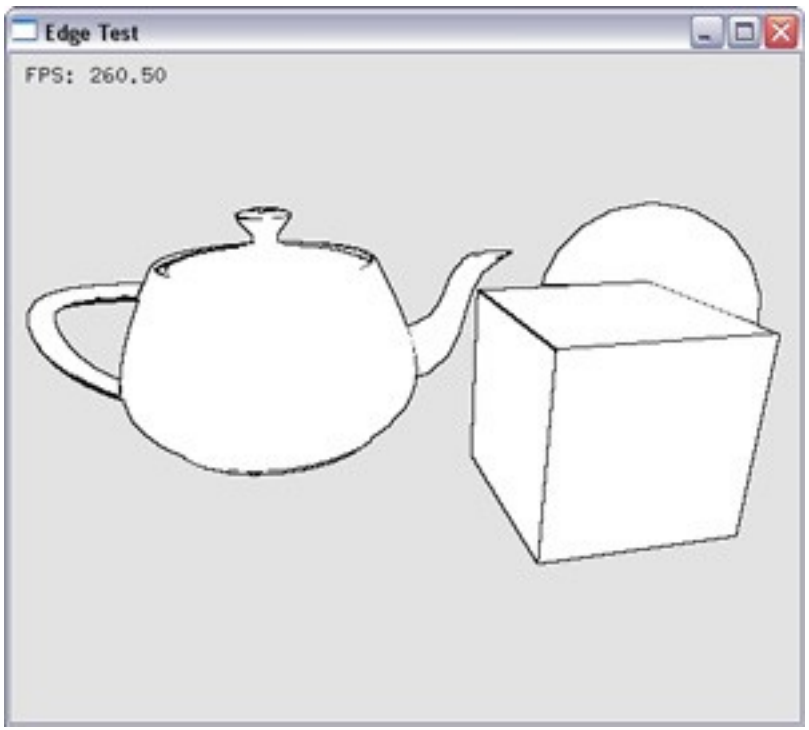
Shading

- Cálculo de iluminação do fragmento por fontes de luz
- Basicamente, uma multiplicação escalar de vetores



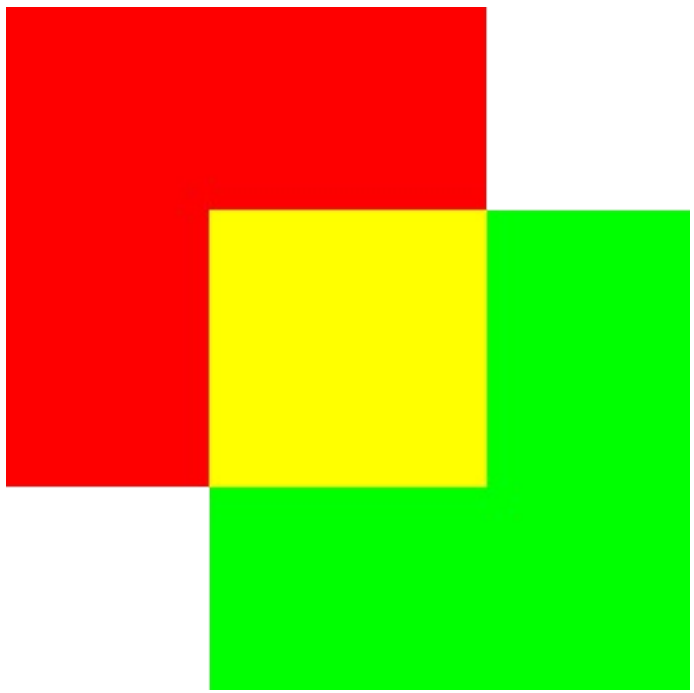
Depth e *Scissor Tests*

- Determinam se o fragmento está oculto por outro ou deve ser ignorado



Blending

- Finalmente, o resultado é colocado no *buffer*/tela
- Pode substituir um antigo, ser somado, etc.



Programando Gráficos

- *Pipeline* Fixa
  - Antigamente: a *pipeline* era fixa e o programador apenas a controlava dentro do programa na máquina *host*
  - API como DirectX e OpenGL permitem controlá-la
  - Grande limitação de visuais possíveis

- Pipeline Customizável
  - Atualmente: algumas etapas podem ser especificadas em linguagens de programação
  - DirectX e OpenGL permitem a criação destes programas especiais

- Agora somos responsáveis pelo o que acontece dentro e fora da GPU! Yeah!

Exemplo de *Vertex* e Pixel *Shaders*

```
float4x4 worldViewProj : WorldViewProjection;

struct VertexInput {
float4 pos : POSITION;
};

struct VertexOutput {
float4 pos : POSITION;
float4 color : COLOR;
};

VertexOutput MyVertexShader(VertexInput input)
{

VertexOutput output = (VertexOutput) 0;
output.pos = mul(input.pos, worldViewProj);
output.color = float4(1, 1, 1, 1);

return output;
}

float4 MyPixelShader(VertexOutput in) : COLOR0
{
return in.color;
}

technique MyTechnique {
pass P0 {
VertexShader = compile vs_2_0 MyVertexShader();
PixelShader = compile ps_2_0 MyPixelShader();
}
}
```

Perguntas?

Obrigado!

Referências

http://www.chadvernon.com/blog/tutorials/managed-directx-2/frustum-culling/

http://dispatchevent.org/calebjohnston/papervision3d-introduction-p1/

http://commons.wikimedia.org/wiki/File:Rasterization_bw.svg

http://www.riemers.net/eng/Tutorials/DirectX/Csharp/Series3/Textured_Triangle.php

http://www.justadventure.com/articles/3D/3DGraphicsTrens.shtm

http://dwightdesign.com/portfolio/programs/

http://www.gehacktes.net/2010/01/alphablending-with-opengl/

http://fourrivers.innerweaver.com/shaders-part-3