CGGT

Computer Graphics and Game Technology Group

Desenvolvimento de shaders

Alessandro Ribeiro da Silva

28/09/2007

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Conteúdo

- **Apresentação**
- **Introdução a shaders**
- **Linguagens**
- **Ferramentas**
- **Exemplos**
- **Considerações finais**

Apresentação

- **Grupo CGGT criado no segundo semestre de 2006**
  - **Voltado para estudos nas áreas:**
    - **Computação gráfica**
    - **Jogos**
  - **Vários temas abordados**
    - **Deste a criação de jogos até questões mercadológicas**

Apresentação

Participe do grupo através de nossa lista de discussão

https://listas.dcc.ufmg.br/mailman/listinfo/cggt

Web-site

https://www.dcc.ufmg.br/projetos/cggt

Introdução a shaders

- **Shaders são programas que permitem a programação de GPUs**
  - **Efeitos em tempo-real**

Introdução a shaders História

- **Shade Trees – Cook, Robert L. (SIGRAPH84)**

http://graphics.pixar.com/ShadeTrees/paper.pdf

- **Interface Renderman (1989)**
  - **Possui vários tipos de shaders diferentes**
  - **Implementação PRMan – Pixar**

https://renderman.pixar.com/products/rispec/

- **Stanford RealTime Shading Language (2002)**

http://graphics.stanford.edu/projects/shading/

Introdução a shaders História

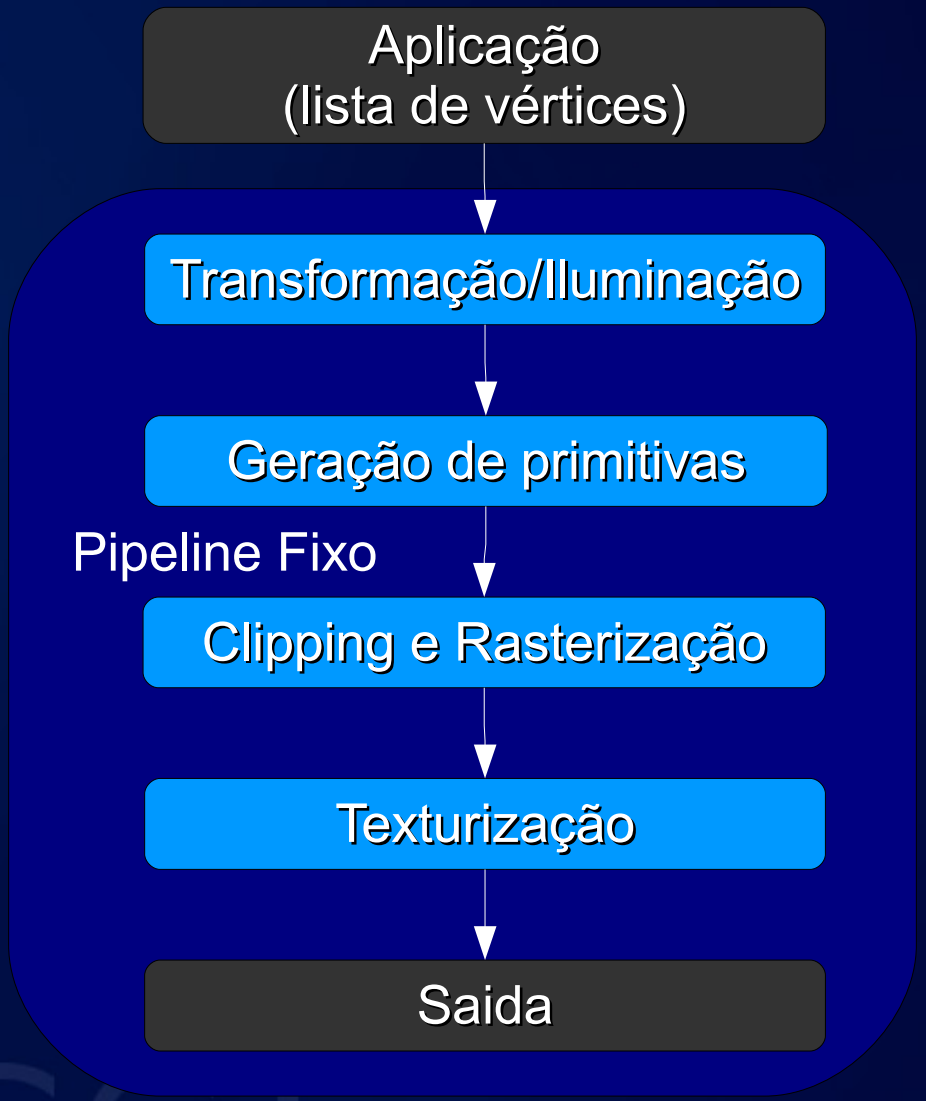
- **Hardware doméstico**
  - **2001**
    - **Programação por vértice (texture shader)**
  - **2002-2003**
    - **Desenvolvimento da programação por Pixel**
    - **Suporte a linguagens de alto nível**
    - **Multiple Render Target**
  - **2004**
    - **Suporte a branch(controle de fluxo dinâmico)**
    - **Suporte a leitura de textura no Proc. Vert.**

Introdução a shaders História

- **Hardware doméstico**
  - **2006-2007**
    - **Suporte a geração de primitivas**
    - **Estamos aqui!!!**

Introdução a shaders

Pipeline abstrato

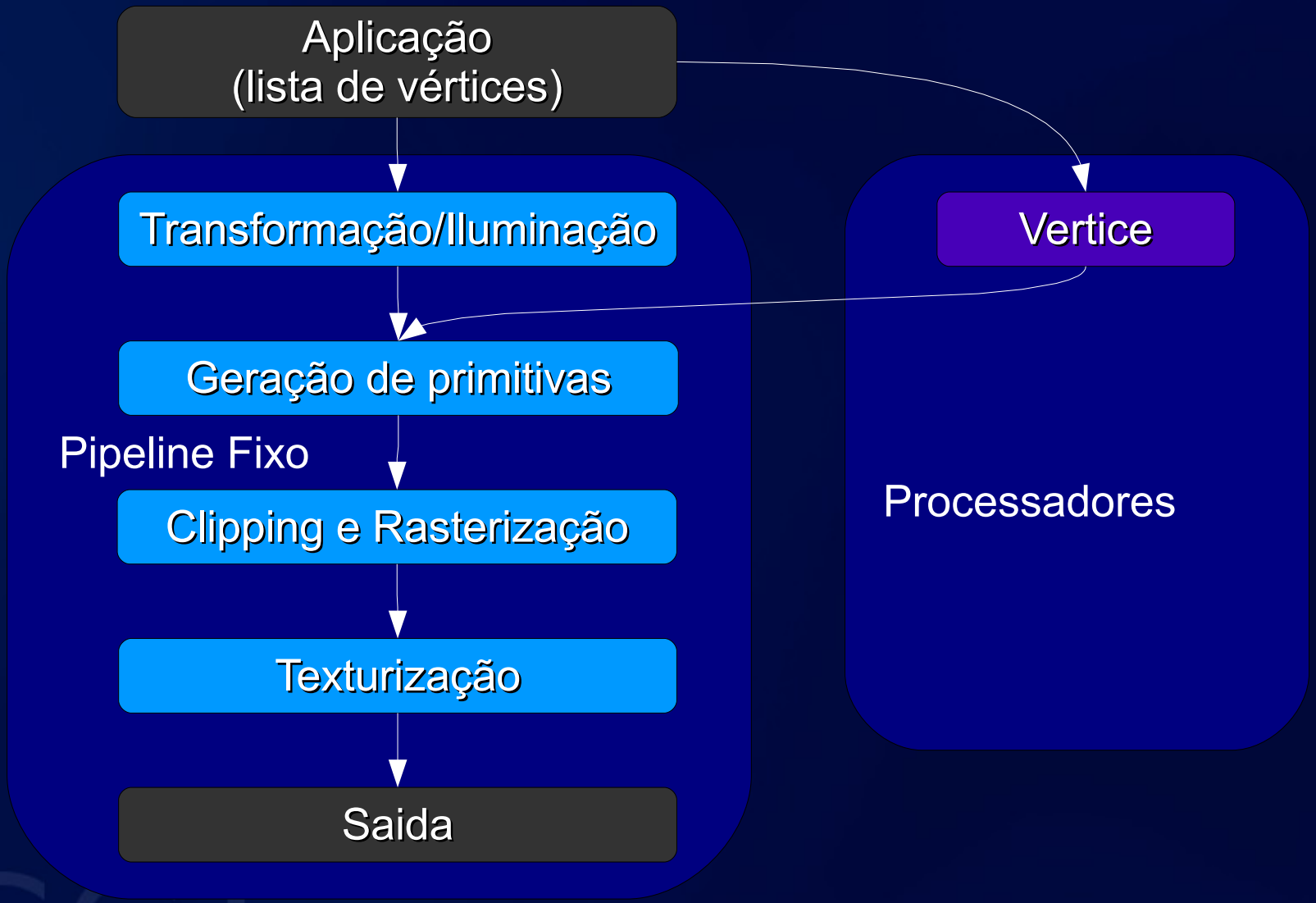


Introdução a shaders

Pipeline abstrato

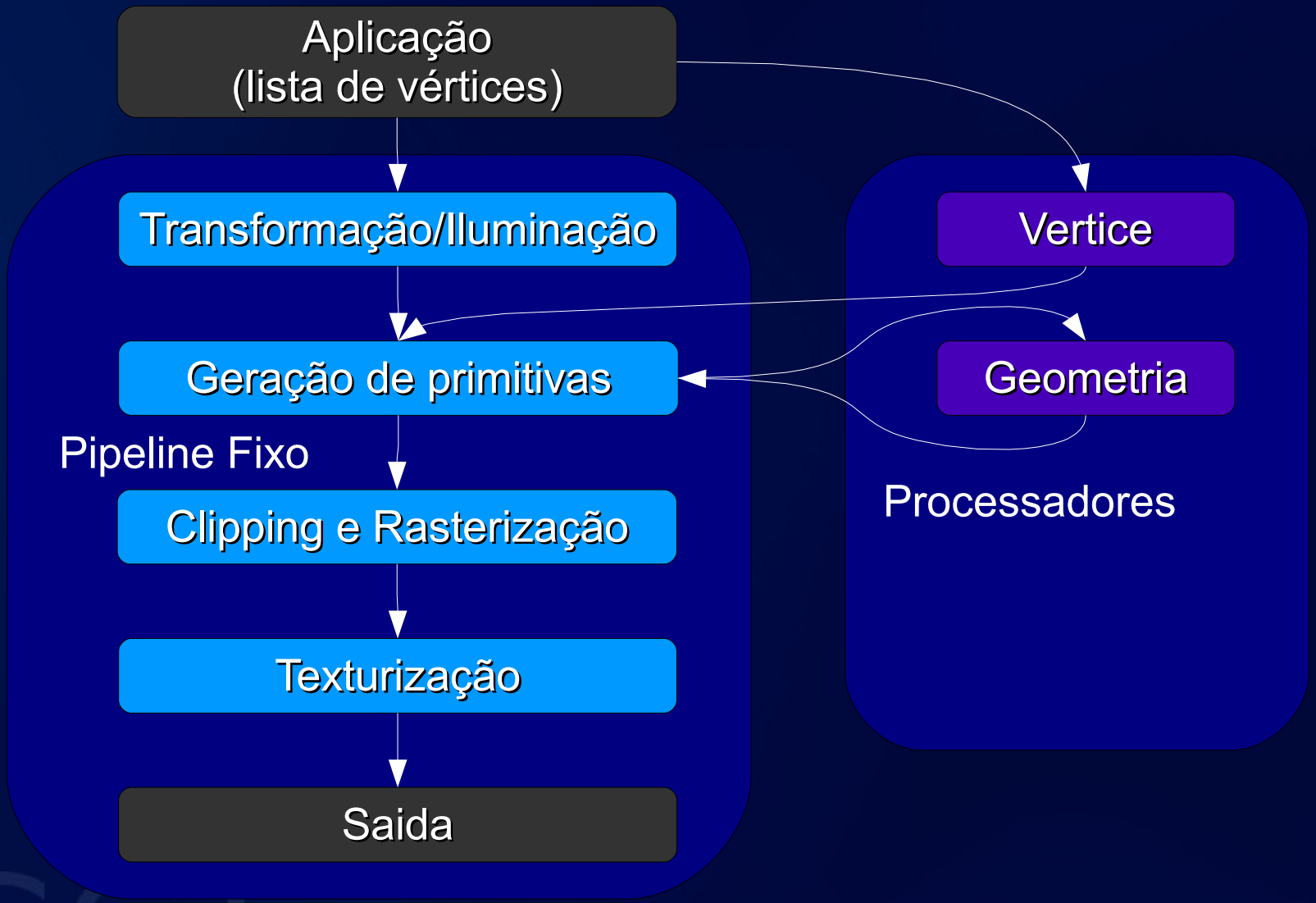
Introdução a shaders

Pipeline abstrato



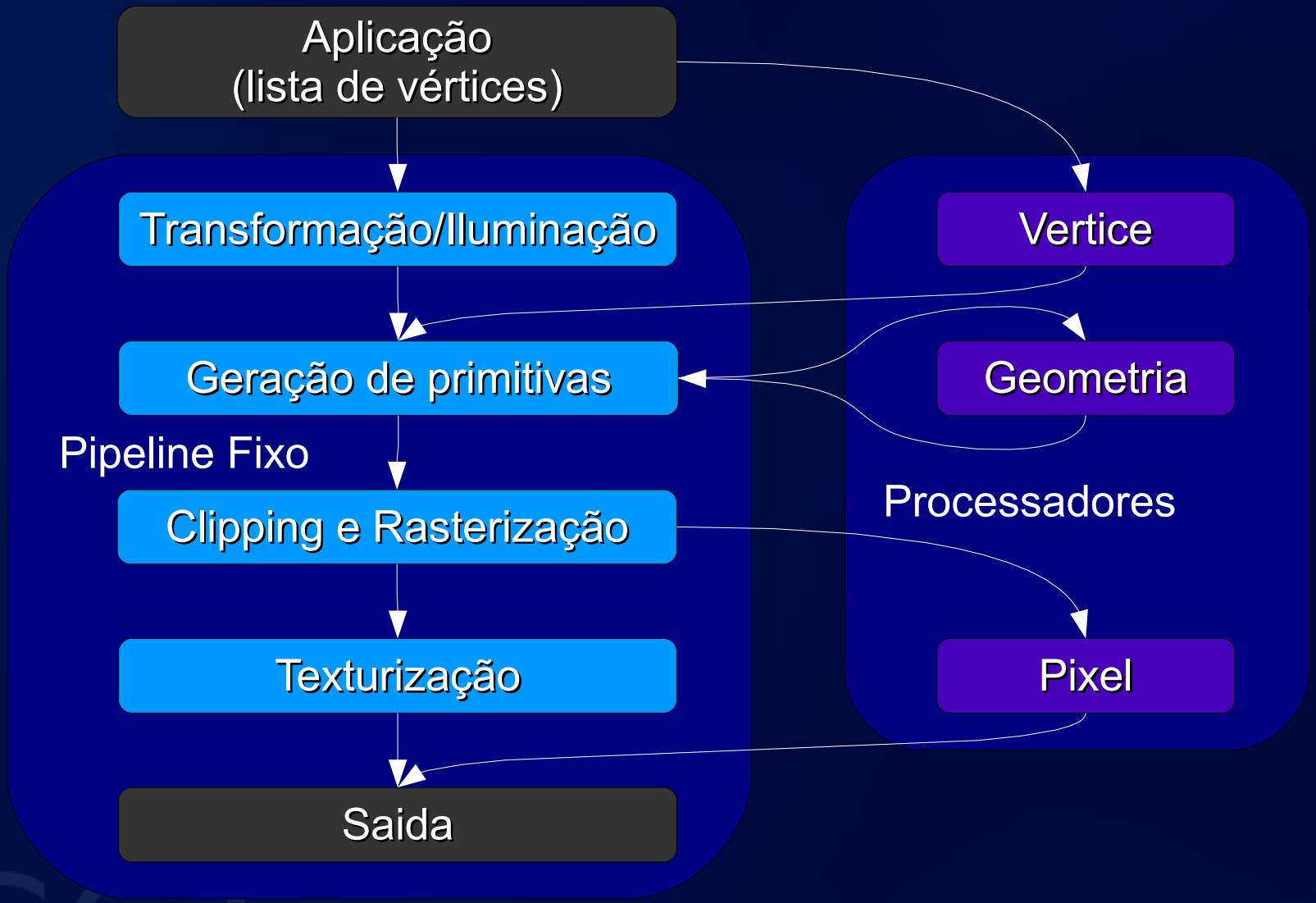
Introdução a shaders

Pipeline abstrato



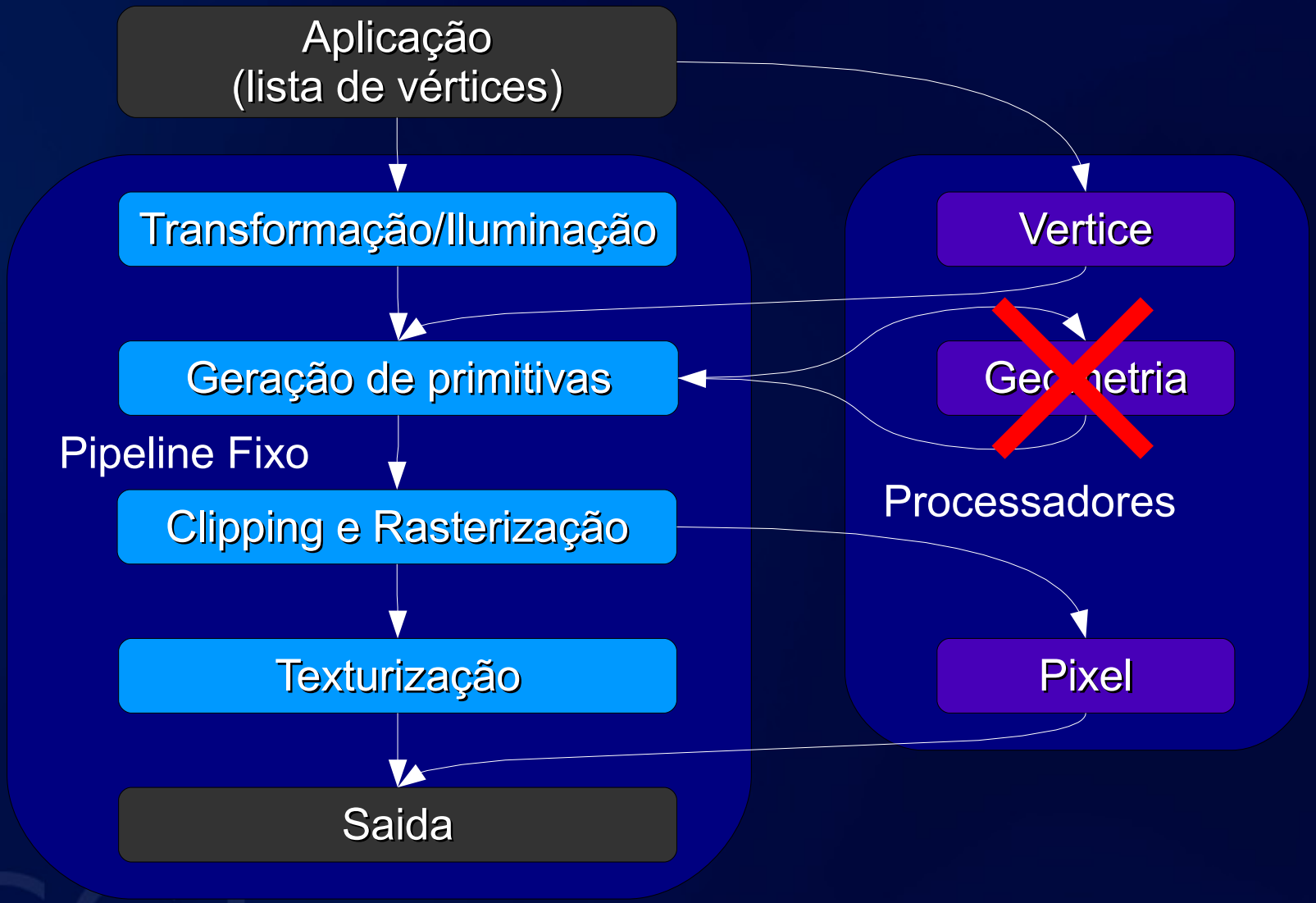
Introdução a shaders

Pipeline abstrato



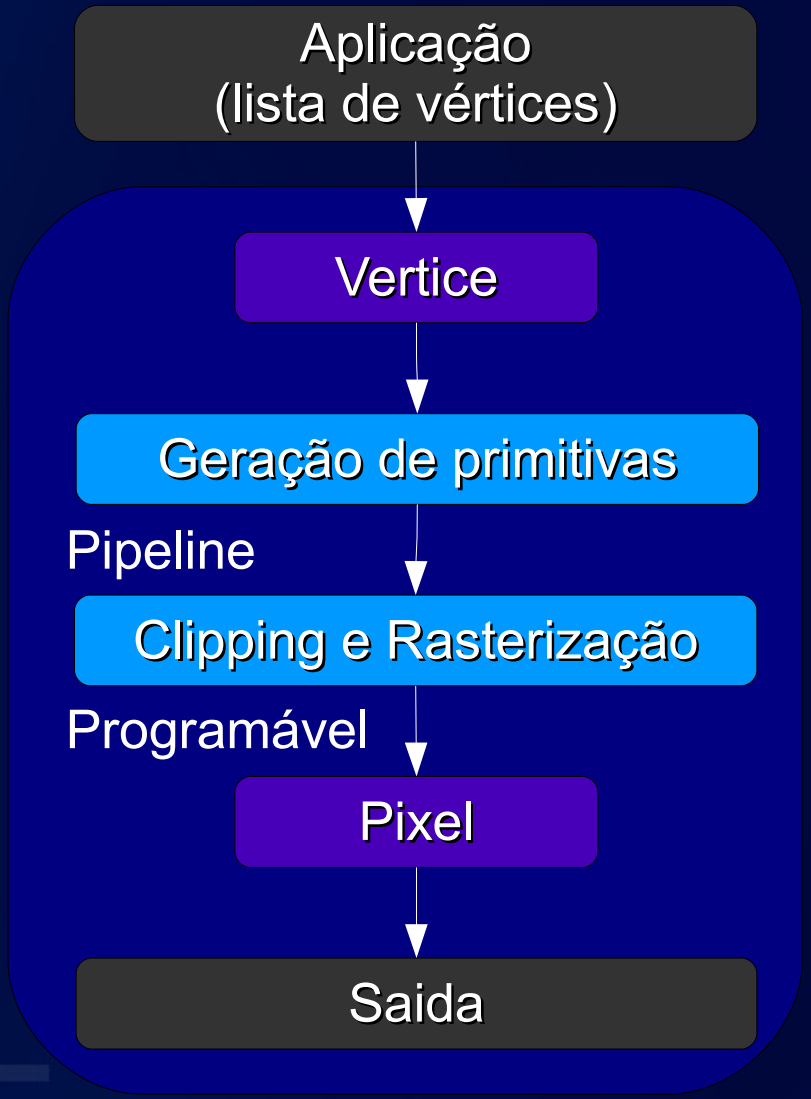
Introdução a shaders

Pipeline abstrato



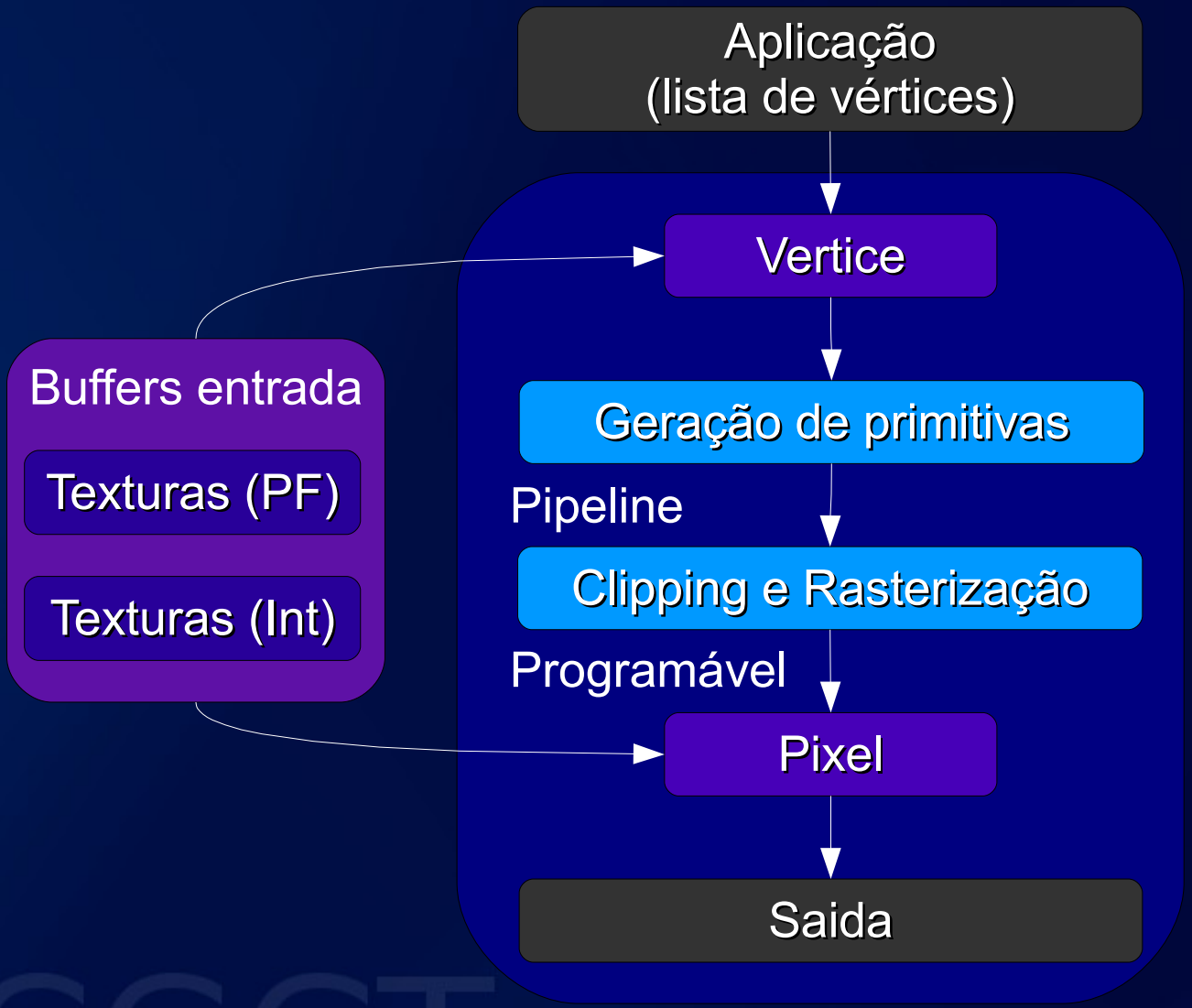
Introdução a shaders

Pipeline abstrato



Introdução a shaders

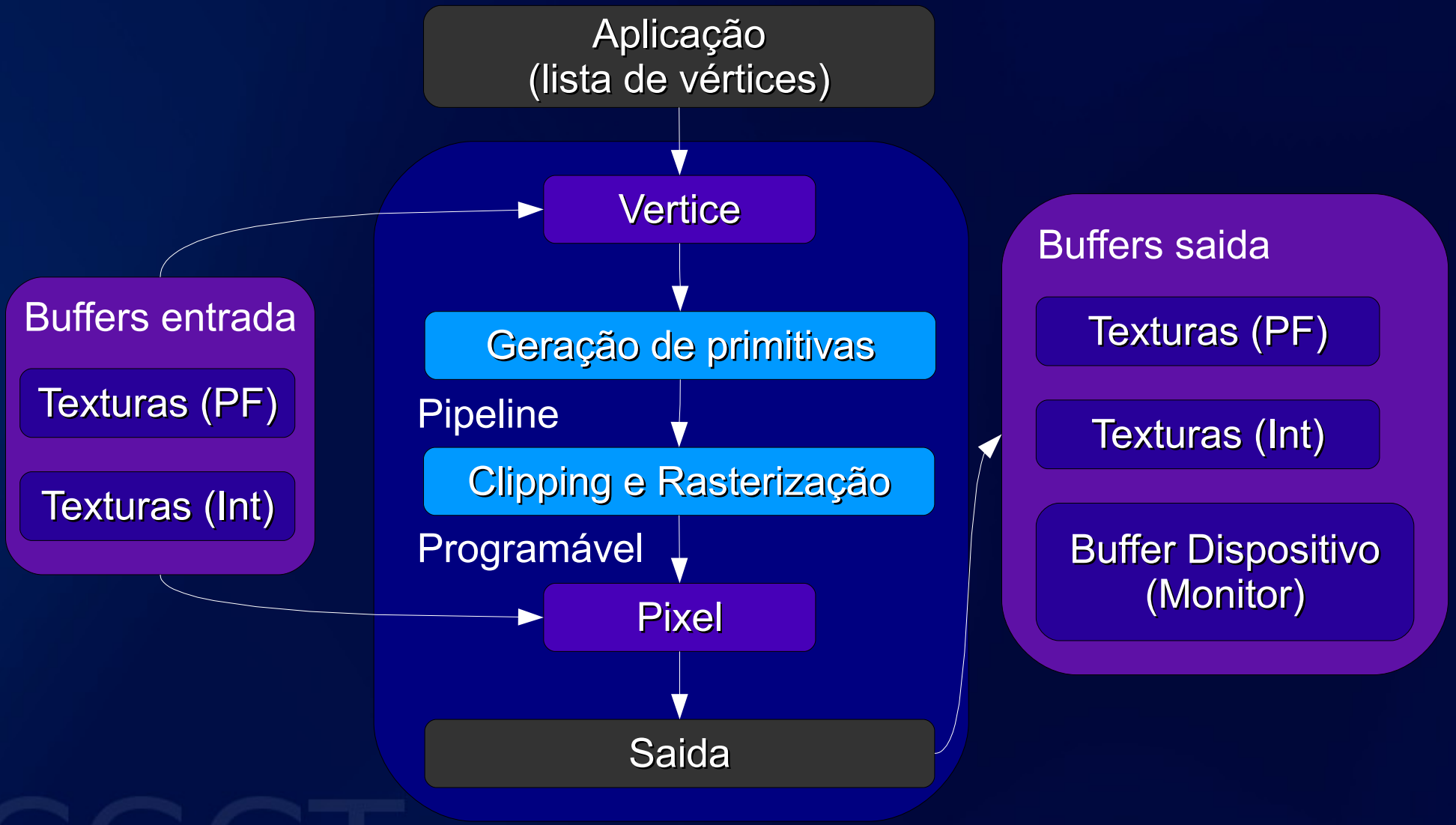
Pipeline abstrato



Alessandro Ribeiro da Silva

Introdução a shaders

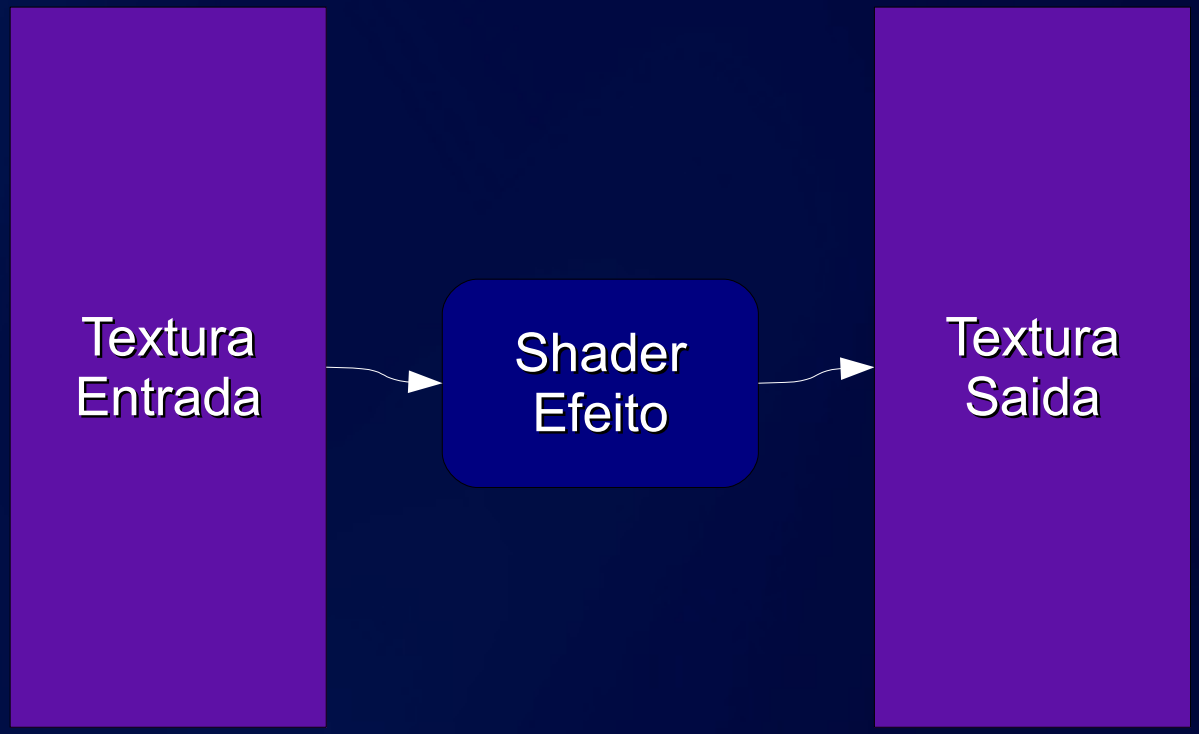
Pipeline abstrato



Alessandro Ribeiro da Silva

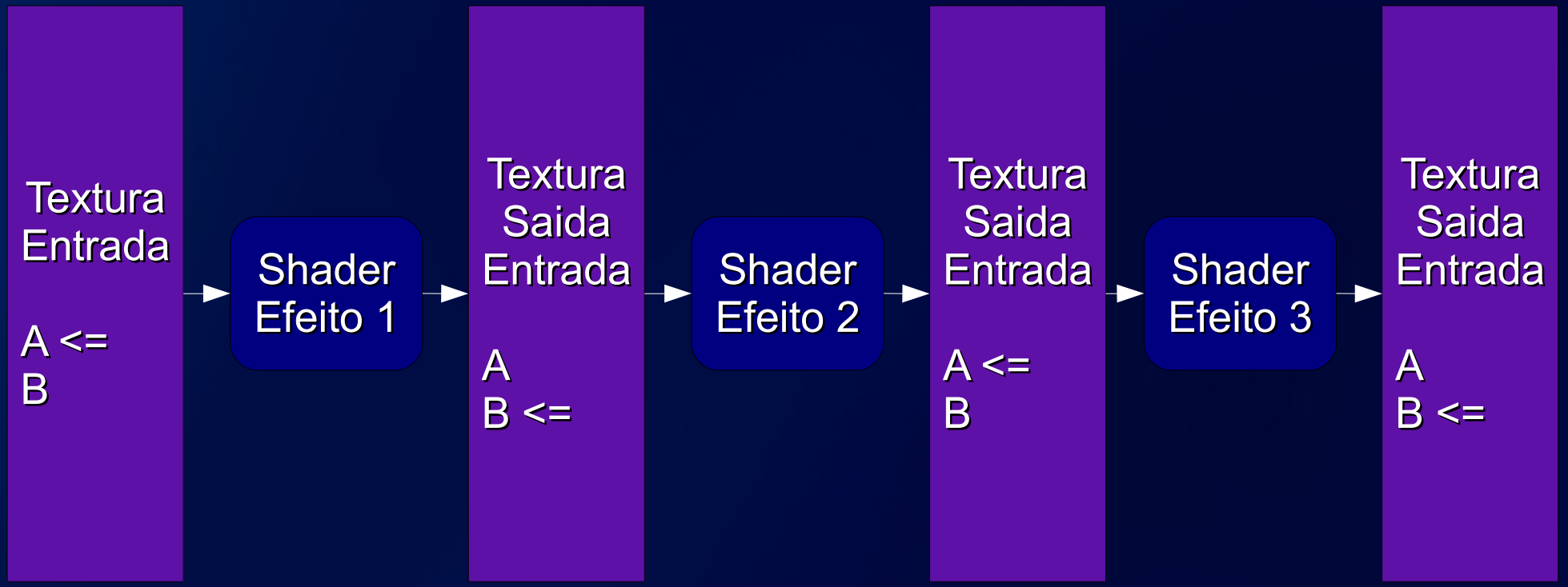
Introdução a shaders

Pipeline abstrato



Introdução a shaders

Pipeline abstrato



Alessandro Ribeiro da Silva

Linguagens

- **Baixo nível**
  - **Fragment Program – OpenGL**
  - **...**
- **Alto nível específicas de APIs**
  - **GLSL – OpenGL – Consórcio de empresas**
  - **HLSL – DirectX – Microsoft**
- **Virtual**
  - **Cg – (OGL e DX) – Nvidia**

Linguagens

- **Quando utilizar linguagens de baixo nivel tenha em mente**
  - **A eficiência do código depende do desenvolvedor**
  - **A criação e manutenção dos algoritmos podem consumir MUITO tempo**

Linguagens

```
!!ARBfp1.0
TEMP t1; TEMP t2; TEMP t3; TEMP t4; TEMP t5;
TEMP a;  TEMP b;  TEMP c;  TEMP d;  TEMP e;
ADD a,{0,-2},fragment.texcoord[0];
ADD b,{0,-1},fragment.texcoord[0];
MOV c,fragment.texcoord[0];
ADD d,{0, 1},fragment.texcoord[0];
ADD e,{0, 2},fragment.texcoord[0];
TEX t1, a, texture[0], RECT;
TEX t2, b, texture[0], RECT;
TEX t3, c, texture[0], RECT;
TEX t4, d, texture[0], RECT;
TEX t5, e, texture[0], RECT;
ADD t1,t1,t2;
ADD t3,t3,t4;
ADD t1,t1,t3;
ADD t1,t1,t5;
MUL result.color.rgb,t1,{0.2,0.2,0.2};
END
```

Linguagens

```
!!ARBfp1.0
TEMP t1; TEMP t2; TEMP t3; TEMP t4; TEMP t5;
TEMP a;  TEMP b;  TEMP c;  TEMP d;  TEMP e;
ADD a,{0,-2},fragment.texcoord[0];
ADD b,{0,-1},fragment.texcoord[0];
MOV c,fragment.texcoord[0];
ADD d,{0, 1},fragment.texcoord[0];
ADD e,{0, 2},fragment.texcoord[0];
TEX t1, a, texture[0], RECT;
TEX t2, b, texture[0], RECT;
TEX t3, c, texture[0], RECT;
TEX t4, d, texture[0], RECT;
TEX t5, e, texture[0], RECT;
ADD t1,t1,t2;
ADD t3,t3,t4;
ADD t1,t1,t3;
ADD t1,t1,t5;
MUL result.color.rgb,t1,{0.2,0.2,0.2};
END
```

ARG!!!

Linguagens

- **O mesmo algoritmo implementado em uma linguagem de alto nível**

```
void main(out float4 outputColor: COLOR0,
          in float2 texcoord: TEXCOORD0,
          in float4 inputColor:COLOR,
          uniform samplerRECT texture){
  float3 aux=0;
  for(int i=-2;i<=2;i++)
    aux += texRECT(texture, texcoord + float2(0,i) ).rgb;
  outputColor = float4(aux*0.2,1);
}
```

Linguagens

- **O mesmo algoritmo implementado em uma linguagem de alto nível**

```
void main(out float4 outputColor: COLOR0,
          in float2 texcoord: TEXCOORD0,
          in float4 inputColor:COLOR,
          uniform samplerRECT texture){
  float3 aux=0;
  for(int i=-2;i<=2;i++)
    aux += texRECT(texture, texcoord + float2(0,i) ).rgb;
  outputColor = float4(aux*0.2,1);
}
```

xD

Linguagens Sintaxe do formato CGFX/HLSL

```
float4x4 ModelViewProjMatrix : ModelViewProjectionMatrix;
...
struct VertIn{
  float4 pos : POSITION;
  float4 Normal : NORMAL;};
struct Vert2Frag{
  float4 pos : POSITION;
  float4 Color: COLOR;};

void vert(uniform float4x4 mvp,...,
          in VertIn IN, out Vert2Frag OUT){ ... }

float4 frag(in Vert2Frag IN):COLOR{ return IN.Color; }

technique teste <int it = 10; string name = "hoho";> {
 pass P0{
  VertexProgram = compile arbvp1 vert(ModelViewProjMatrix,...);
  FragmentProgram = compile arbfp1 frag();
 }
}
```

Linguagens Observações

- **O controle de buffers não é feito pela “linguagem” (API)**
  - **Deve ser realizado na aplicação**

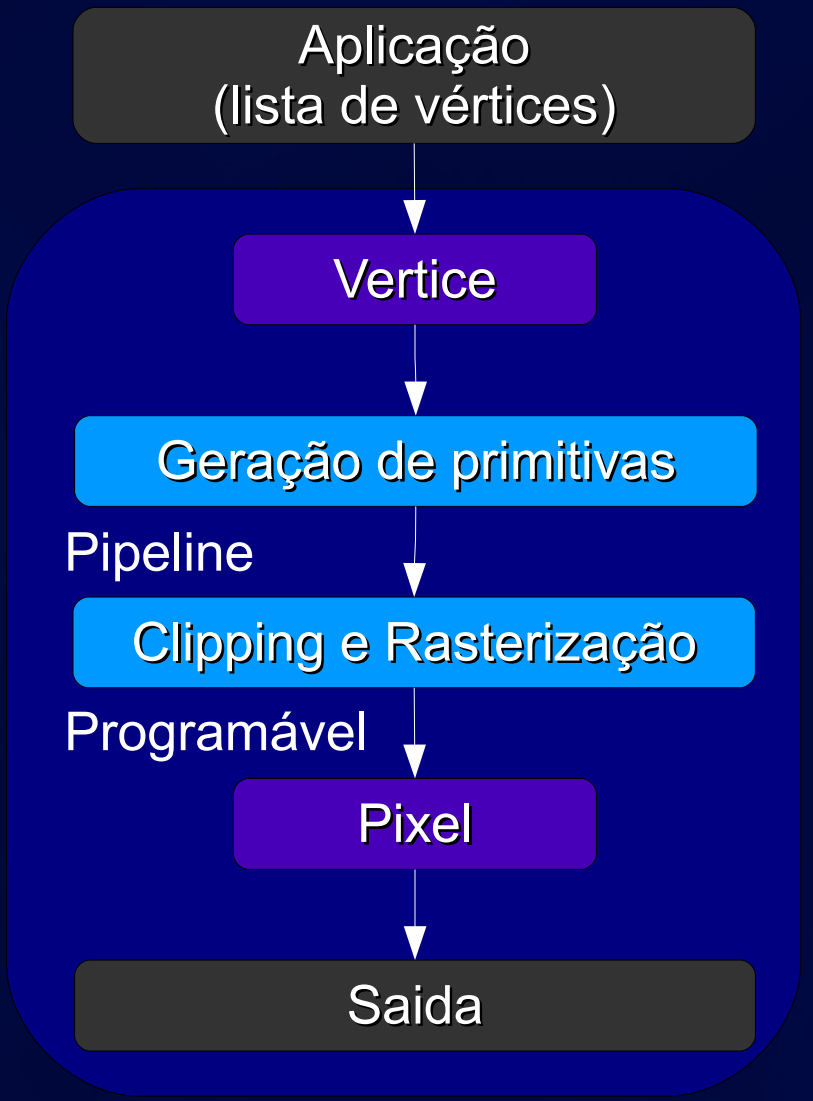
- **Existem duas formas de entrar com os dados nos shaders**
  - **Uniformes (vindos da máquina de estado ou da aplicação)**
  - **Texturas**

Linguagens Observações

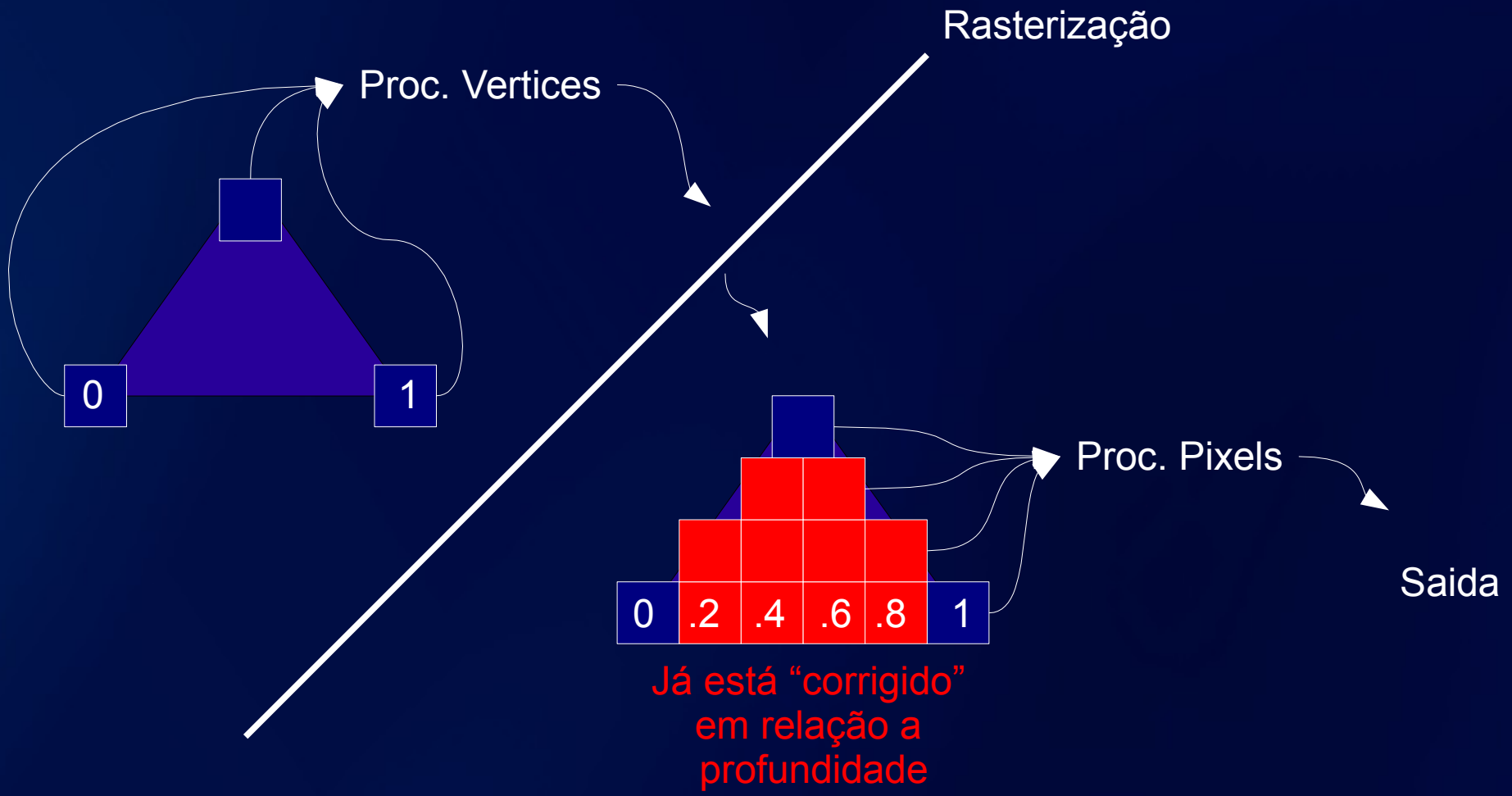
- **TODOS os dados passados do proc. de vértices para o de proc. de fragmentos são interpolados linearmente**

```
...
struct Vert2Frag{
  float4 pos : POSITION;
  float4 Color: COLOR;};

void vert(..., out Vert2Frag OUT)
{...}
float4 frag(in Vert2Frag IN):COLOR
{...}
...
```



Linguagens Observações



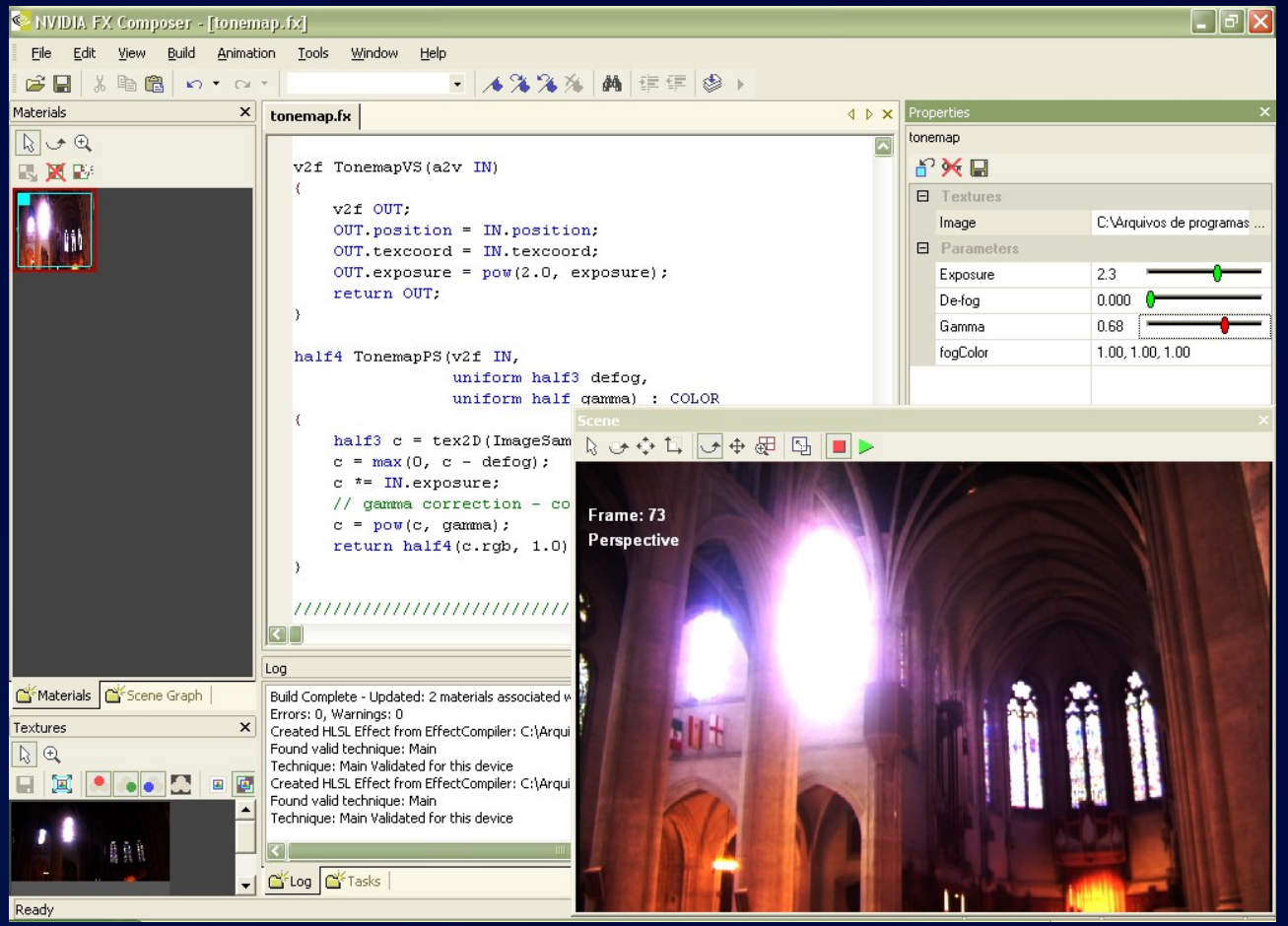
Ferramentas

- **Interessante para desenvolver programas de shaders antes de serem utilizados em um jogo ou aplicação**

- **Ferramentas:**
  - **FXComposer 2.0 (Todas linguagens) – Nvidia**
  - **RenderMonkey (GLSL e HLSL) – ATI**
  - **ShaderDesign (GLSL) – ThypoonLabs3D**

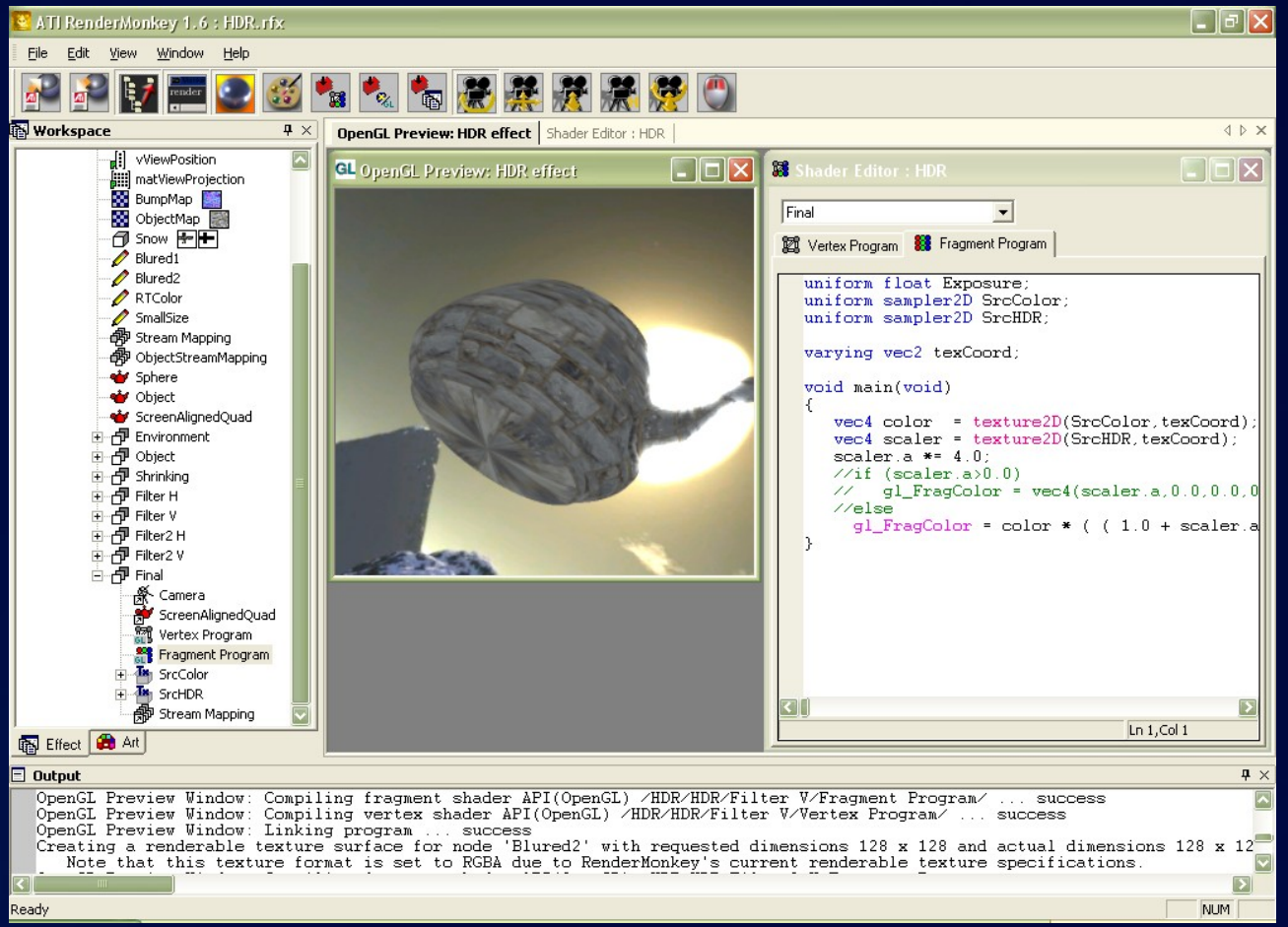
Ferramentas

- **FXComposer 2.0**



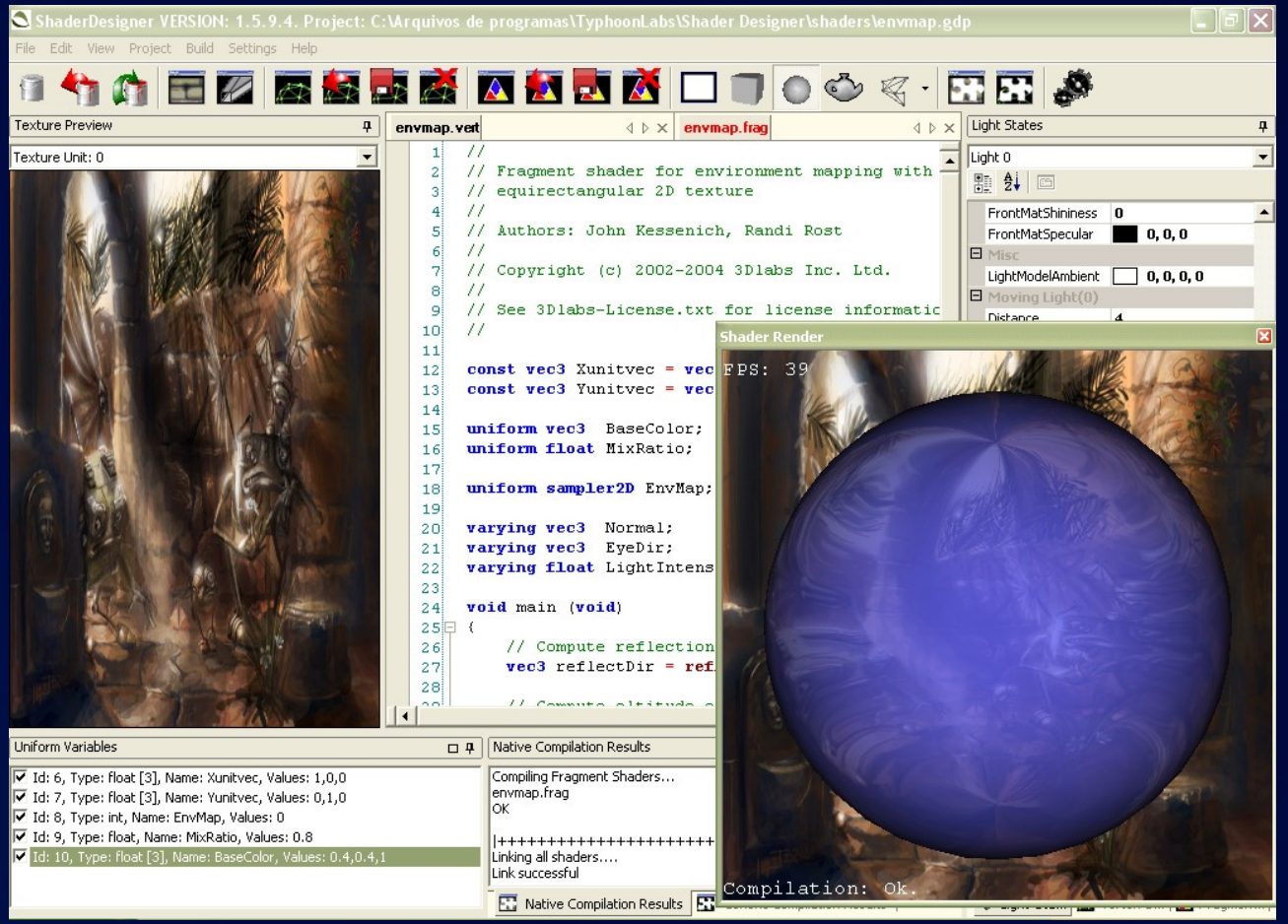
Ferramentas

- **RenderMonkey (GLSL e HLSL) – ATI**



Ferramentas

- **ShaderDesign (GLSL) – ThypoonLabs3D**

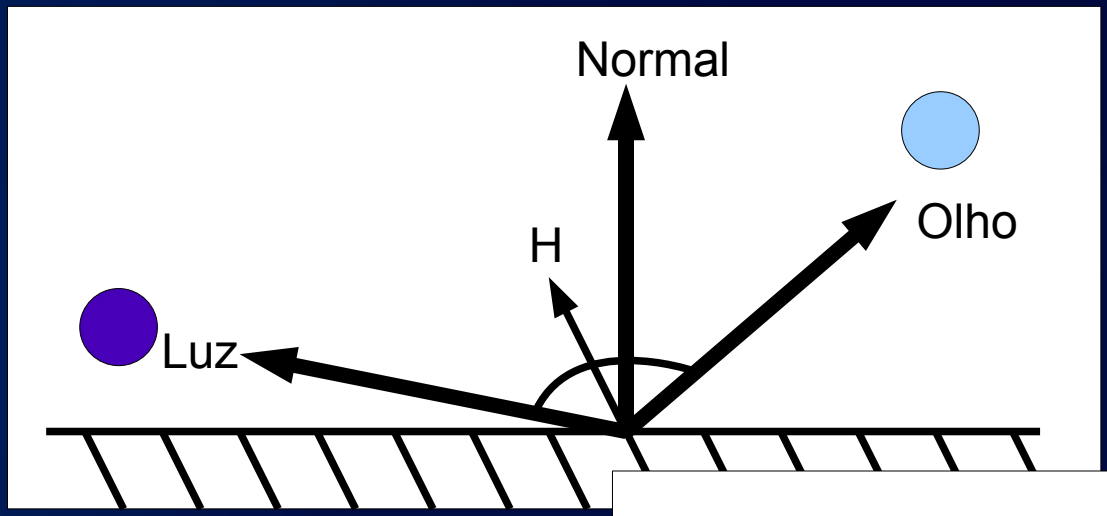


Exemplos

- **Modelo de iluminação**
- **Texturas**
- **Efeitos elaborados**
- **Pós-processamento**

Exemplos Modelo de iluminação

- **Blinn – Phong**



$$H = \frac{(Olho + L)}{|Olho + L|}$$

$$Kd = N.L\,(Lambert)$$

$$Ke = H.N\,(Simplificação\ Phong)$$

$$Cor = Ka * ambiente + Kd * corDifusa + Ke^{pow} * corRefletida$$

Exemplos Modelo de iluminação

- **Blinn – Phong**

```
void BlinPhong(...){
  ...
  float3 h = normalize(eye + light );
  float kd = saturate(dot(normal, light));
  float ke = saturate(dot(normal, h));
  outputColor = 0.02*color + kd*color*0.5 + pow(ke,256)*float3(1);
}
```

$$H = \frac{(Olho + L)}{|Olho + L|}$$

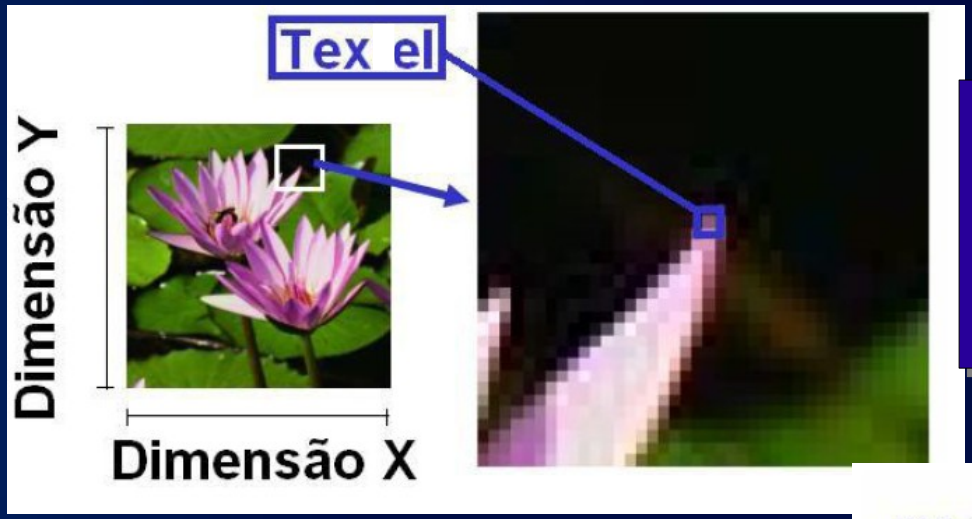
$$Kd = N.L\,(Lambert)$$

$$Ke = H.N\,(Simplificação\ Phong)$$

$$Cor = Ka * ambiente + Kd * corDifusa + Ke^{pow} * corRefletida$$

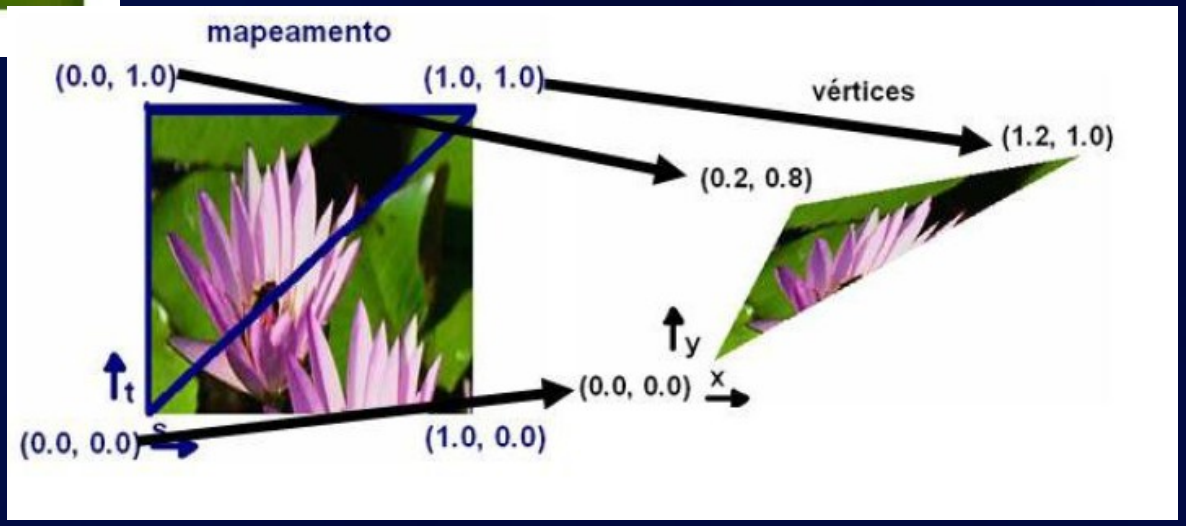
Exemplos Texturas

- **Forma de mapear uma imagem a um triângulo**




```
float4 textureSimple():COLOR0{
...
cor = tex2D(tex_color,IN.texCoord);
...
}
```

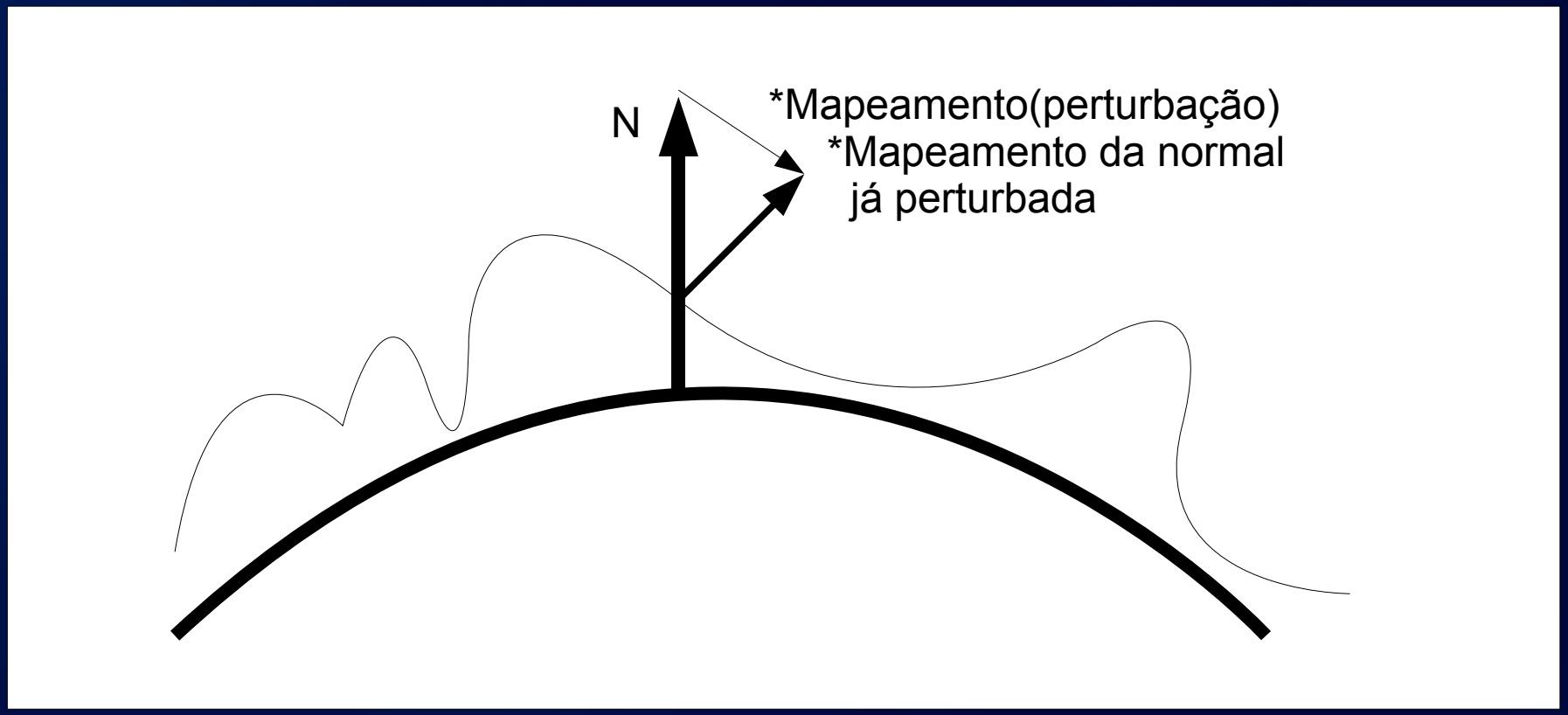
```
... sampler_state {
 WrapS = Clamp;
 WrapT = Clamp;
 MinFilter =
         LinearMipMapLinear;
 MagFilter = Linear;
};
```



Exemplos Efeitos elaborados

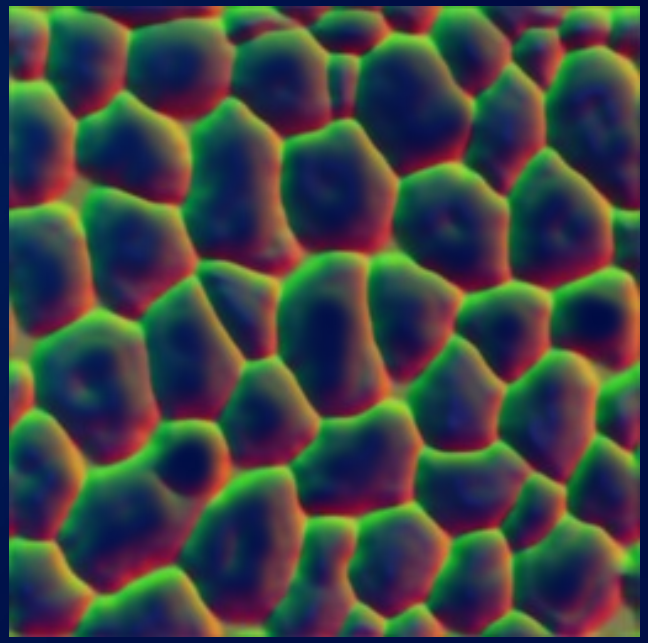
- **Acrescentar detalhes**
  - **Bump-mapping**
  - **Cone-step-mapping**

Exemplos Efeitos elaborados



Exemplos Efeitos elaborados

- **Texturas usadas como dados**

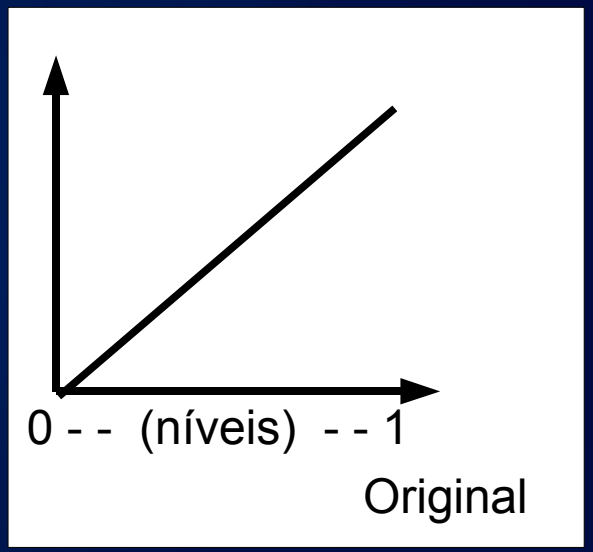


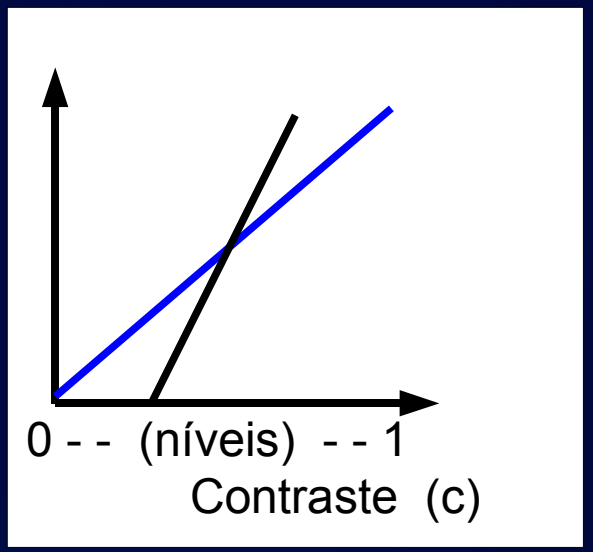
Exemplos Pós-processamento

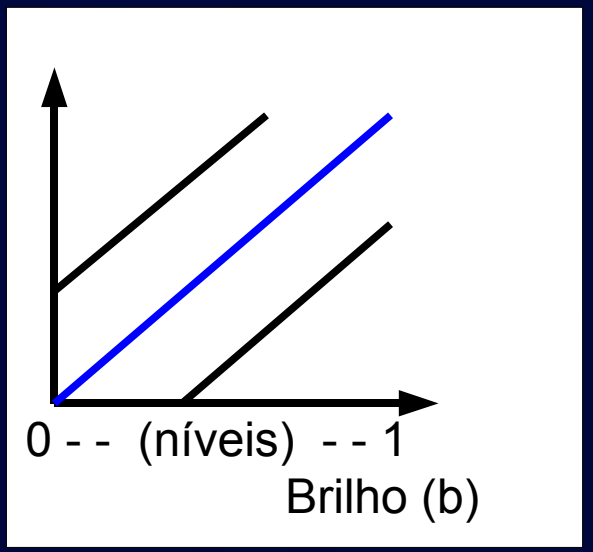
- **A partir da imagem renderizada efetuar um ou mais processamentos sobre a mesma para gerar o(s) efeito(s)**
  - **Possibilita implementação de algoritmos de PDI**
    - **Transformações radiométricas: Brilho/Contraste**
    - **Filtros: média/detecção de bordas**
    - **Composição**

Exemplos Pós-processamento

- **Contraste e brilho**







Exemplos Pós-processamento

- **Extração de tons de cinza**

cinza = (vermelho+verde+azul)/3 (Value - HSV)

cinza = (vermelho * 0.299) +

(verde * 0.587) +

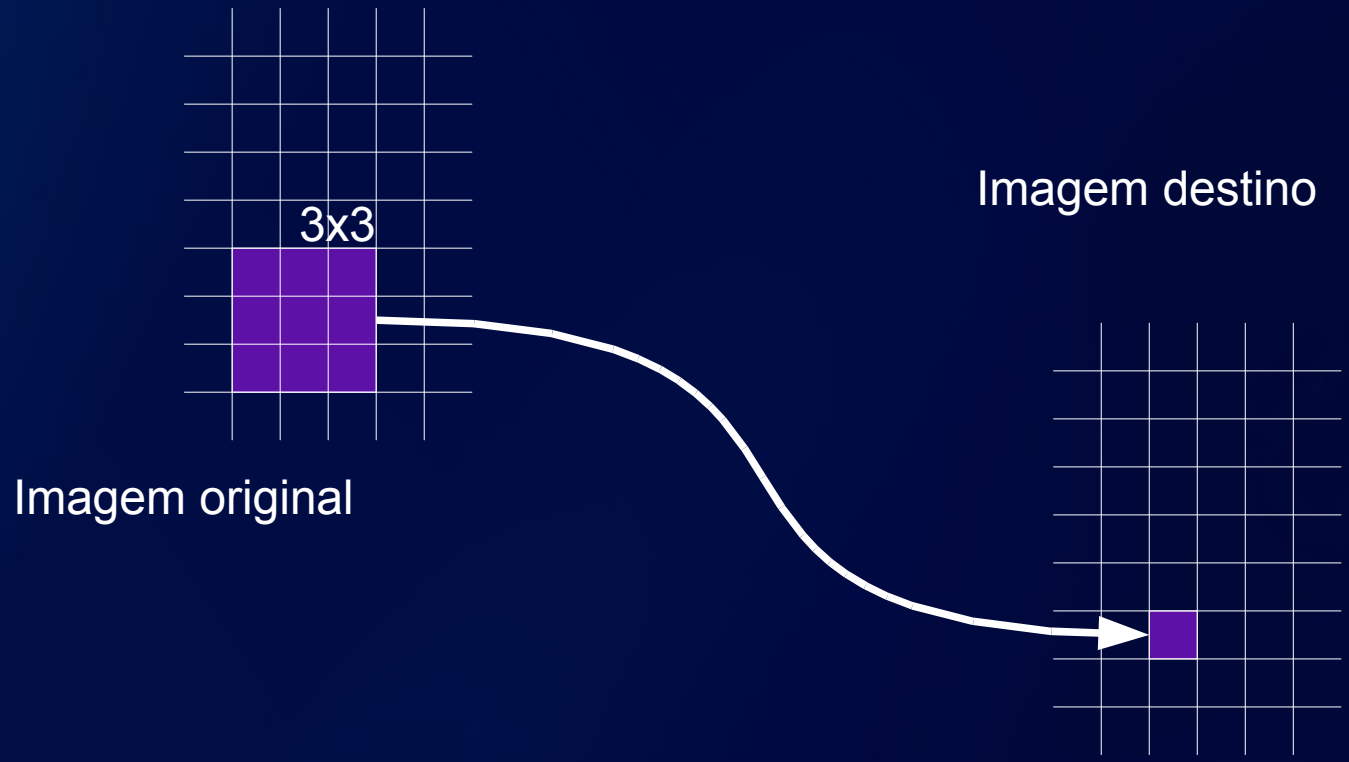
(azul * 0.114) (SVH)

cinza = máximo(vermelho, verde, azul)

(encontrado na internet)

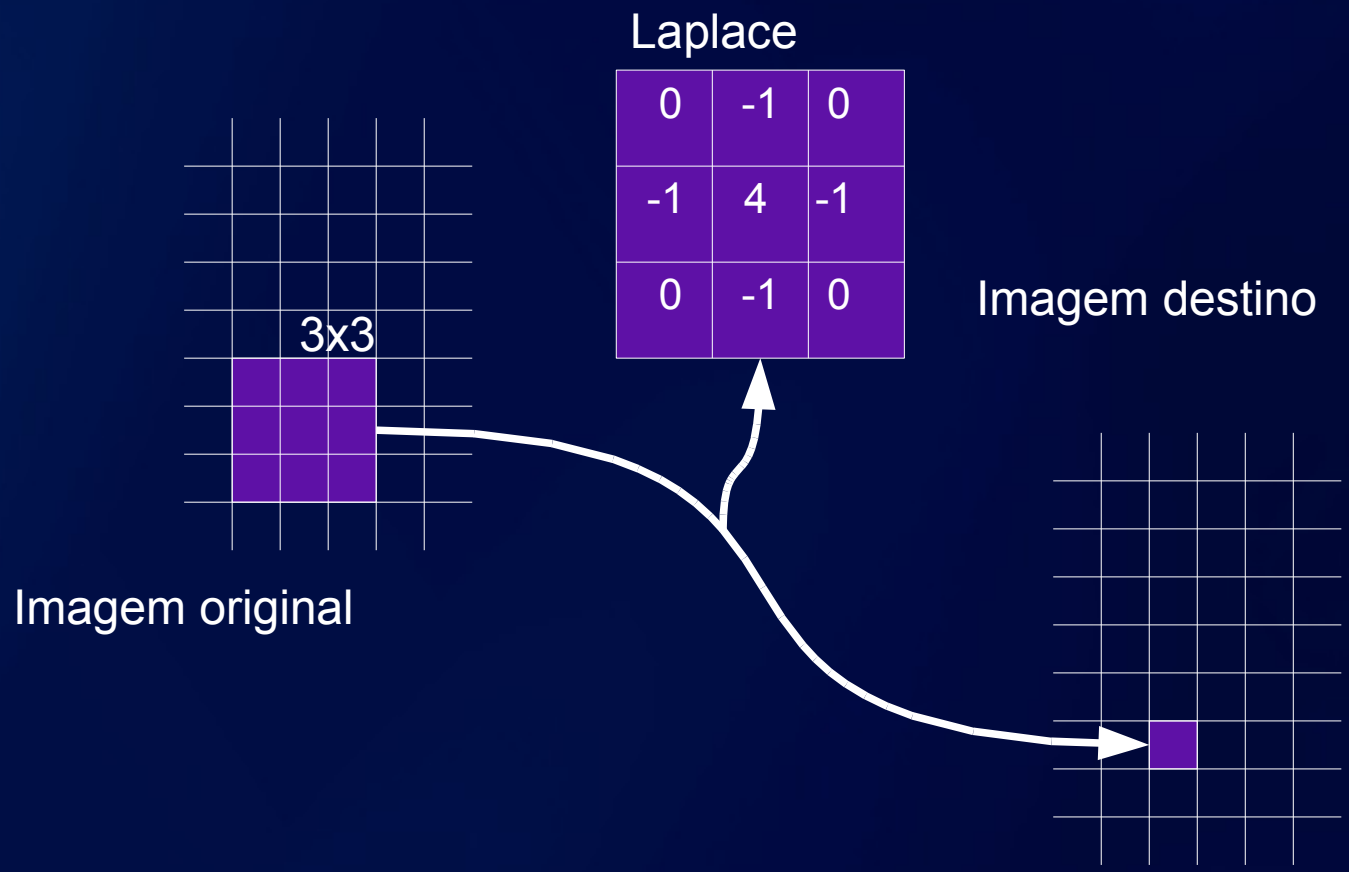
Exemplos Pós-processamento

- **Aplicação de filtros**



Exemplos Pós-processamento

- **Aplicação de filtros**



Considerações finais

- **Tópicos interessantes**
  - **GPGPU**
  - **Animações (Vertex Blend/Skinning)**
  - **Composição de efeitos**

Considerações finais

- **APIs que podem ser utilizadas para programação de GPUs com a linguagem C:**
  - **Close to Metal(CTM) - ATI**
  - **Compute Unified Device Architecture(CUDA) – NVIDIA**

- **Para placas de video antigas ainda é necessário utilizar o pipeline gráfico programável**
  - **< Geforce série 8**
  - **< Radeon série R5x**

Considerações finais

- **Crie**
- **Invente**
- **Tente**

- **Faça um shader diferente!!!**

Considerações finais

- **Tentei abordar de forma abrangente o que é preciso para se desenvolver shaders**
  - **Noção das partes programáveis**
  - **Sintaxe (não é bicho de 7k cabeças)**
  - **Ferramentas**
  - **Efeitos (alguns dos principais tipos de efeitos)**
- **Espero que tenham gostado**

Perguntas?

Alessandro Ribeiro da Silva

Site: www.alessandrosilva.com

Email: alessandro.ribeiro.silva@gmail.com