Introdução as Unidades de Processamento Gráfico (GPUs)

Giovane Roslindo Kuhn

FURB UFRGS

Computação Gráfica x Realidade

Consulta ao BD

```
SELECT nome FROM Pessoa WHERE idade > 25
```

FURB UFRGS

Sumário

- Pipeline Gráfico Fixo (*Overview*)
- Pipeline Gráfico Programável
  - Porque os jogos avançaram tanto em realidade?
- GPU para Propósito Geral
  - Pra que? Como? Exemplos?
- Tecnologias Correntes
- Considerações

FURB UFRGS

Pipeline Gráfico Fixo (*Overview*)

Vértices
Coordenadas textura
Normais

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

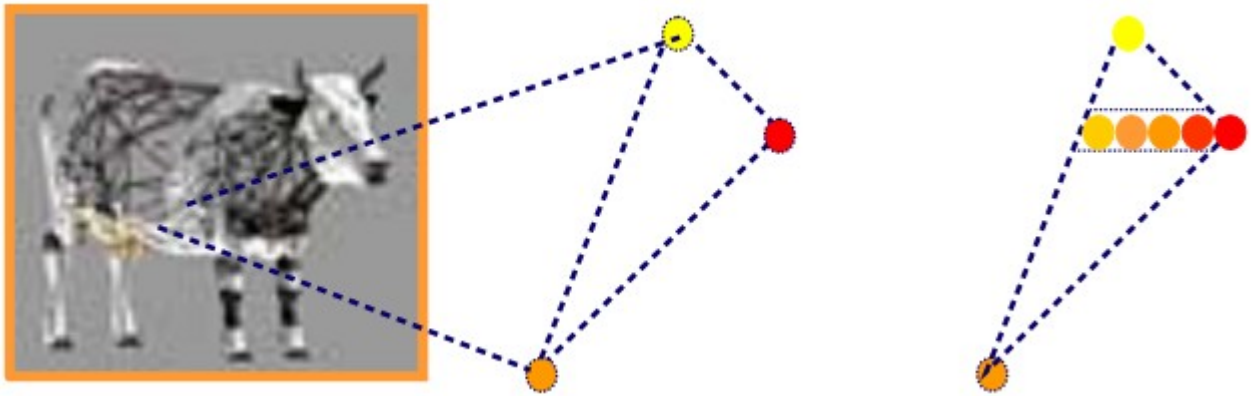
Pipeline Gráfico Fixo (*Overview*)

Transforma vértices para coordenadas do mundo
Transforma vértices para coordenadas da câmera
Aplica iluminação
Efetua clipping
Transforma para coordenadas de tela

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

Pipeline Gráfico Fixo (*Overview*)

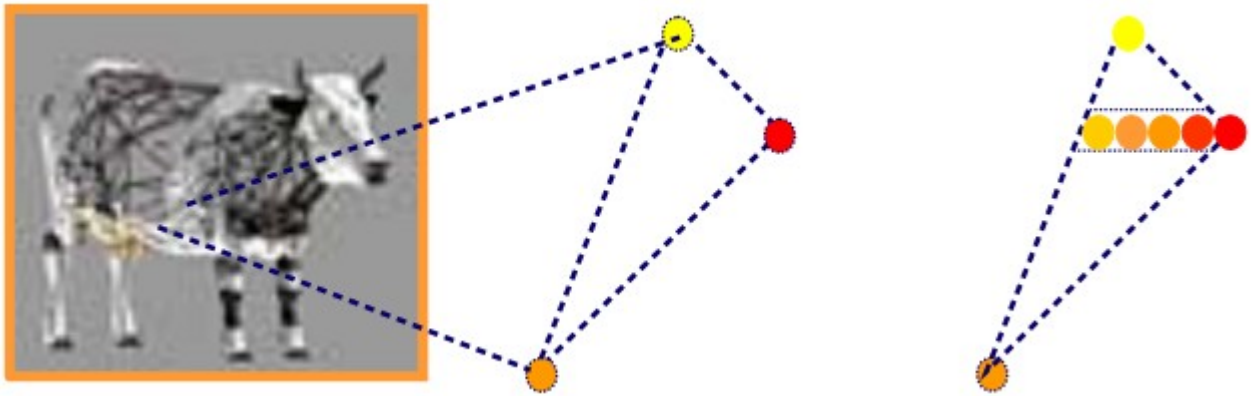


Primitivas são decompostas em fragmentos
Atributos dos fragmentos são a interpolação dos atributos vértices

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

Pipeline Gráfico Fixo (*Overview*)



Texturização é aplicada
Cor dos fragmentos são determinadas
Diversas operações de rasterização são efetuadas
Scissor-test, alpha-test, stencil-test, depth-test
Blending, dithering, operações lógicas

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

Pipeline Gráfico Fixo (*Overview*)

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

Pipeline Gráfico Fixo (*Overview*)

- 1ª geração (1998) – TNT2, Voodoo3
- 2ª geração (1999~2000) – GeForce 2, ATI 7500, Savage3D

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

FURB UFRGS

Paralelismo no Pipeline

Fonte: Comba, Dietrich, Oliveira, Pagot, Scheidegger

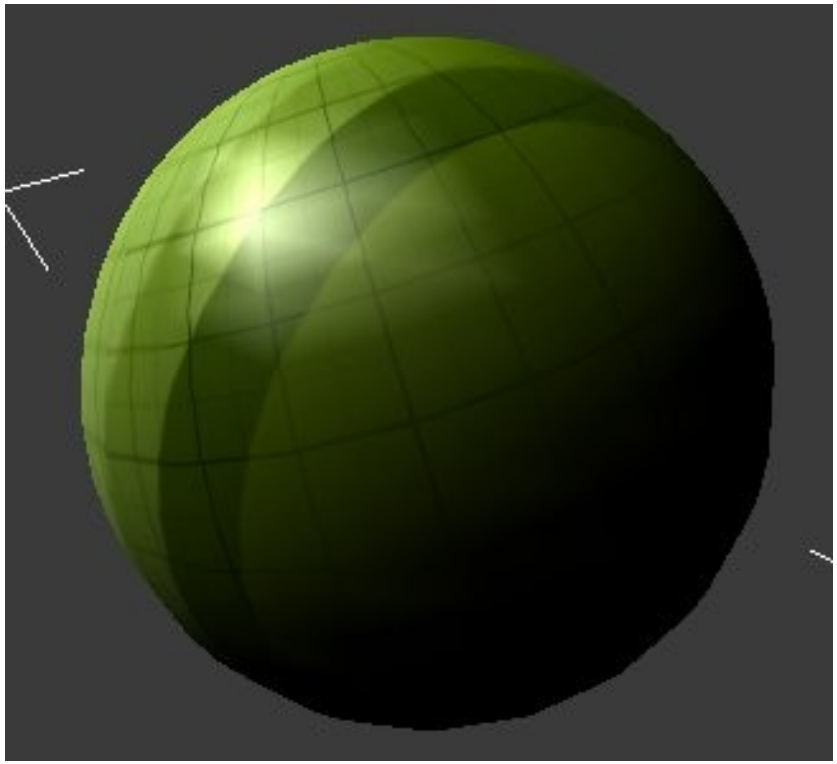
FURB UFRGS

Pipeline Gráfico Programável

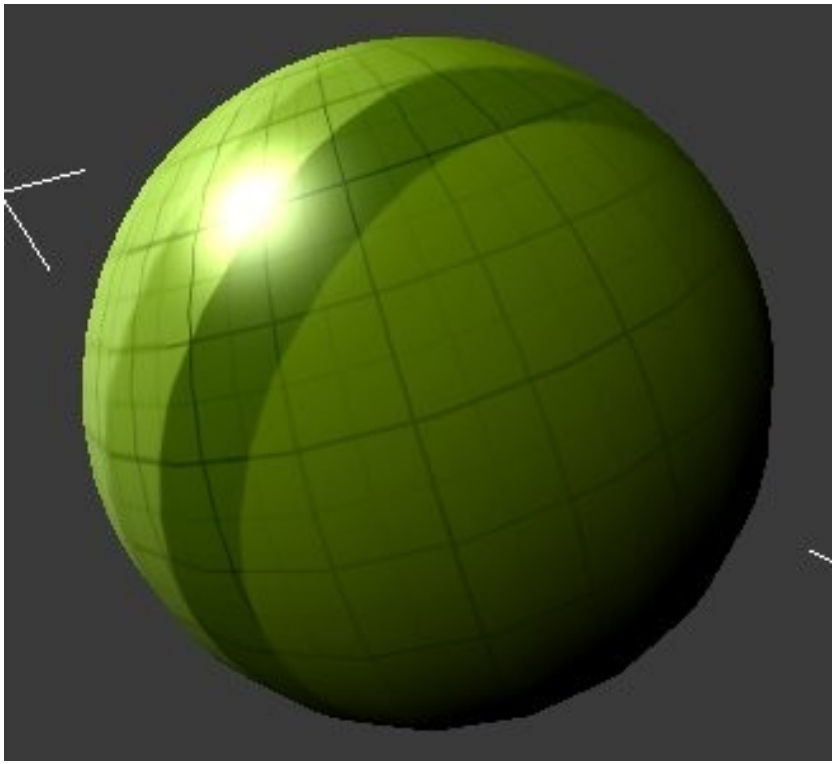
- Programar os processadores de fragmentos
  - 3ª geração (2001~2002) – GeForce 3 e 4, ATI 8500
- Programar os processadores de vértices e fragmentos
  - 4ª geração (2003~2006) – GeForce FX, 6 e 7, ATI 9700 e 9800
- Linguagens gráficas de alto-nível
  - Cg – NVidia
  - HLSL – Microsoft
  - GLSL – OpenGL
- Permitiram os avanços de realismo nos jogos

FURB UFRGS

Modelos de Iluminação



Gouraud shading (por vértice)



Phong shading (por pixel)

FURB UFRGS

Reflexão e Refração

Reflexão

Refração 1 desvio

FURB UFRGS

Efeito Fresnel

Refração 1 desvio

Fresnel

FURB UFRGS

Normal Map

Dilatação

Erosão

FURB UFRGS

Normal Map + Fresnel

Logo

Estudante

FURB UFRGS

Relief Map

Vídeos

Fonte: Manuel M. Oliveira (http://www.inf.ufrgs.br/~oliveira/RTM.html)

FURB UFRGS

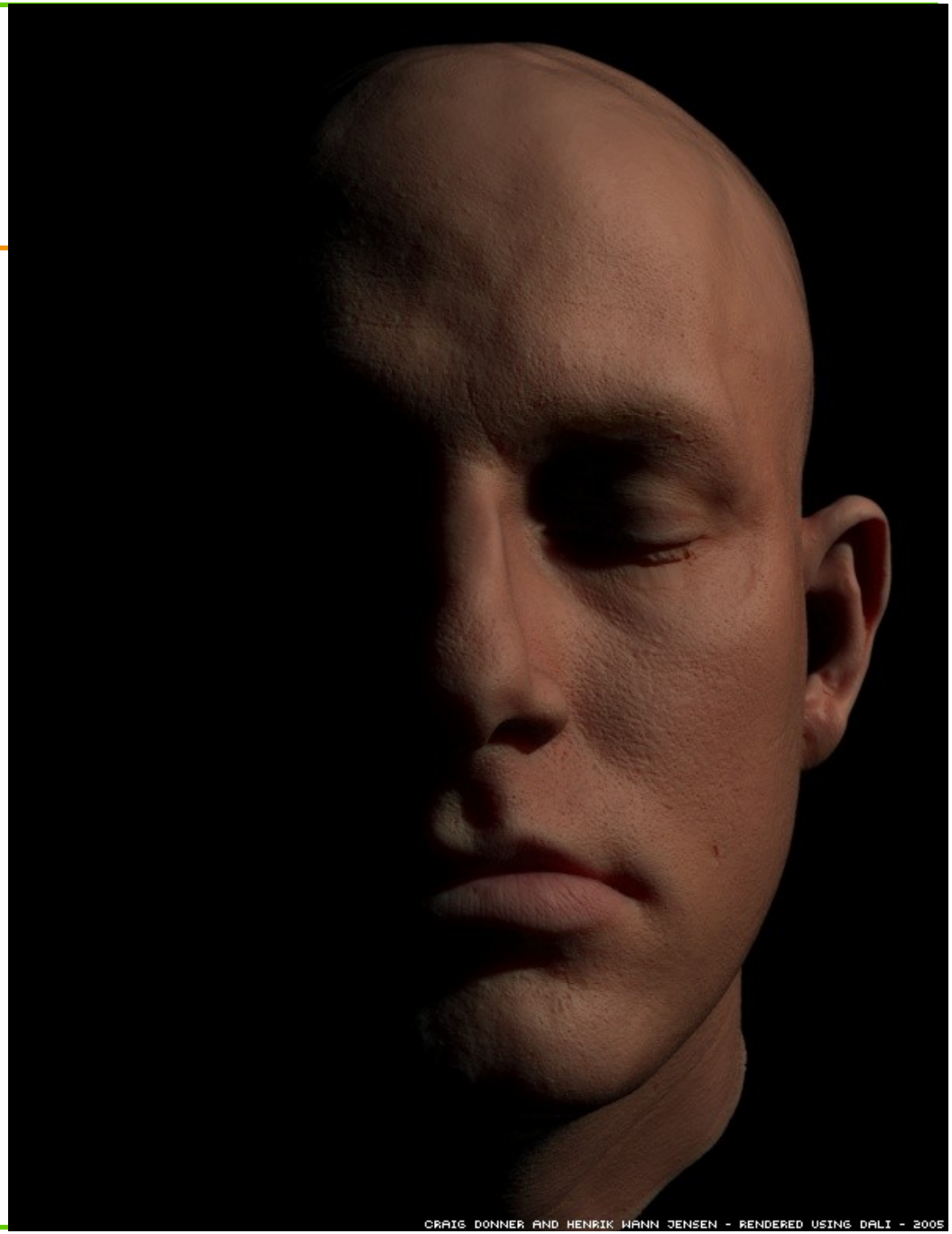
Shadow Map + Caustic Map

Mapeamento de sombras e cáusticas

FURB UFRGS

O que mais?

Subsurface scattering



Fonte: http://graphics.ucsd.edu/~henrik

O que mais?

Translucência

Fonte: http://graphics.ucsd.edu/~henrik

FURB UFRGS

O que mais?

Translucência

Fonte: http://graphics.ucsd.edu/~henrik

FURB UFRGS

Como Começo?

- Recomendo o programa FX Composer
  - Suporte OpenGL e DirectX 9
  - Suporte HLSL, CgFX
  - Dezenas de modelos prontos
- Perfeito para prototipar os shaders
  - Centenas de exemplos prontos
  - Programe apenas o seu shader
  - Altere os parâmetros sem stress

FURB UFRGS

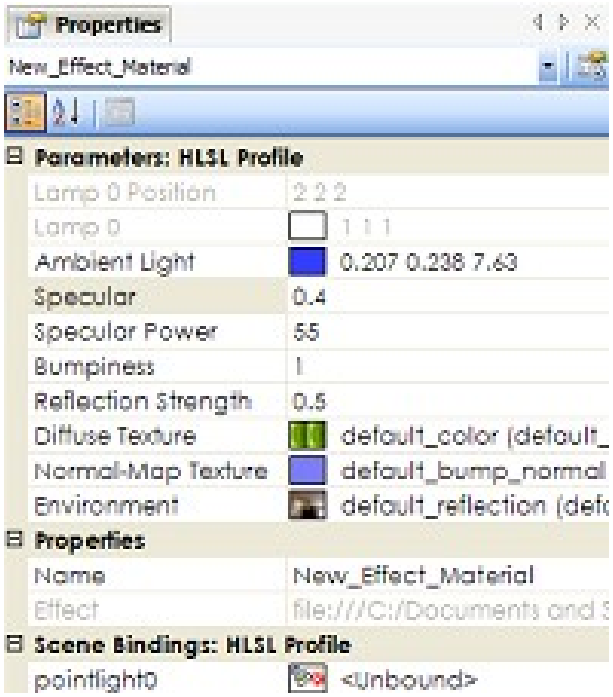
Como Começo?

- Recomendo o programa FX Composer
  - Suporte OpenGL e DirectX 9
  - Suporte HLSL, CgFX
  - Dezenas de modelos prontos
- Perfeito para prototipar os shaders
  - Centenas de exemplos prontos
  - Programe apenas o seu shader
  - Altere os parâmetros sem stress

FURB UFRGS

Como Começo?

- Recomendo o programa FX Composer
  - Suporte OpenGL e DirectX 9
  - Suporte HLSL, CgFX
  - Dezenas de modelos prontos
- Perfeito para prototipar os shaders
  - Centenas de exemplos prontos
  - Programe apenas o seu shader
  - Altere os parâmetros sem stress



FURB UFRGS

Perguntas (1º parte)

Giovane Roslindo Kuhn

grkuhn@gmail.com

FURB UFRGS

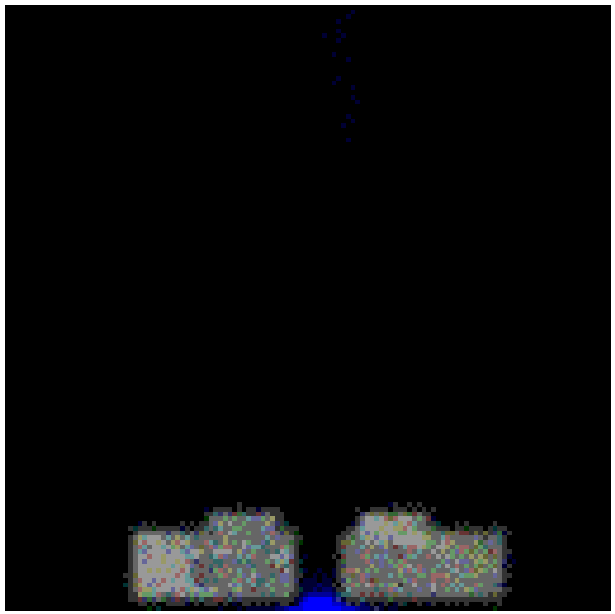
GPU para Propósito Geral

- Porque GPU em computação de propósito geral?

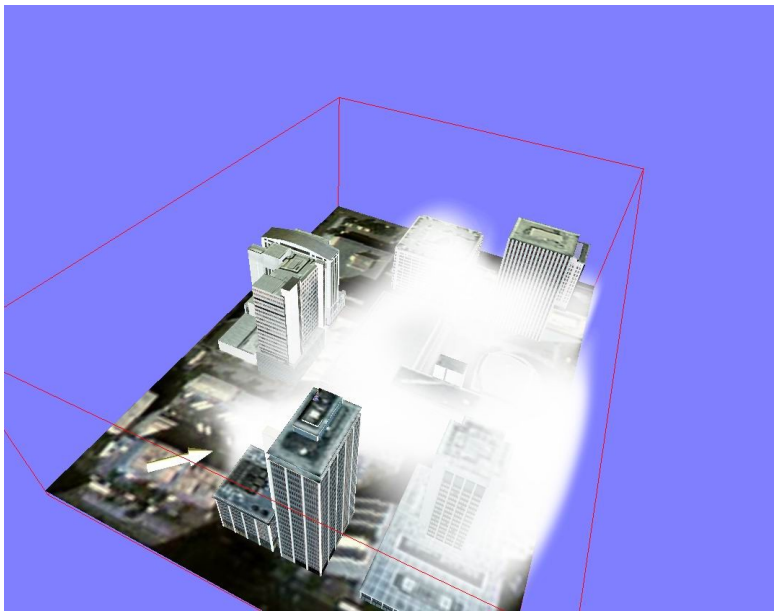
GPU para Propósito Geral

- Porque GPU em computação de propósito geral?

Simular dinâmica de fluídos



Demo da Nvidia

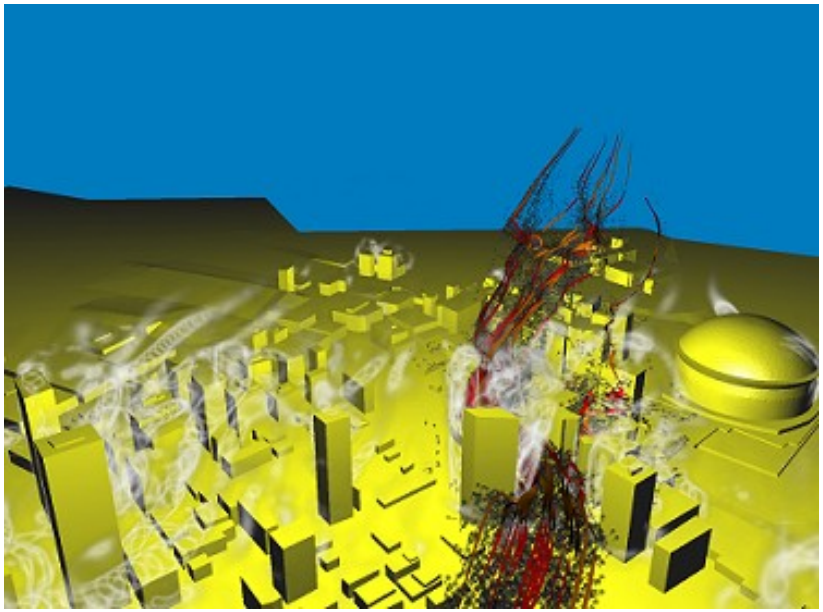


[Liu et al. 2004]

FURB UFRGS

GPU para Propósito Geral (cont.)

Visualizar e interagir com dados científicos

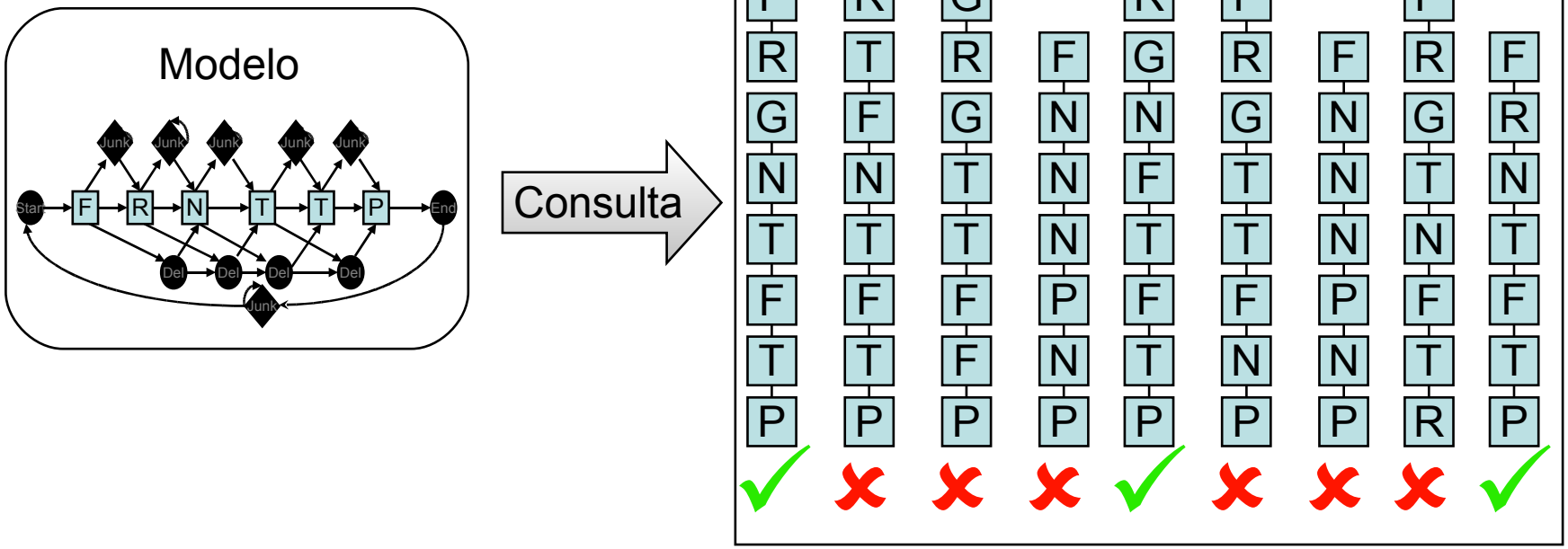


[Krüger et al. 2005]

FURB UFRGS

GPU para Propósito Geral (cont.)

Combinar sequências genéticas



[Horn et al. 2005]

FURB UFRGS

Queremos tudo isso...

- ...no nosso PC desktop
- ...de modo que possamos interagir
- ...de modo a obter resultados instantâneos

FURB UFRGS

Queremos tudo isso...

- ...no nosso PC desktop
- ...de modo que possamos interagir
- ...de modo a obter resultados instantâneos
- Estamos na era do paralelismo em desktops

FURB UFRGS

Paralelismo em Desktops

- Softwares mais rápidos precisam paralelismo
- 2x mais rápido ao dobrar número de cores

FURB UFRGS

Paralelismo em Desktops

- Softwares mais rápidos precisam paralelismo
- 2x mais rápido ao dobrar número de cores

Modelo	P4 3GHz
Ano	2004
Qtde. núcleos.	1
Perf. aritmética	6 GFLOPS
Largura banda	6 GB/sec
Valor	

Fonte: Kurt Akeley, Stanford

FURB UFRGS

Paralelismo em Desktops

- Softwares mais rápidos precisam paralelismo
- 2x mais rápido ao dobrar número de cores

Modelo	P4 3GHz	GF 5900
Ano	2004	2004
Qtde. núcleos.	1	8
Perf. aritmética	6 GFLOPS	20 GFLOPS
Largura banda	6 GB/sec	25 GB/sec
Valor		

Fonte: Kurt Akeley, Stanford

FURB UFRGS

Paralelismo em Desktops

- Softwares mais rápidos precisam paralelismo
- 2x mais rápido ao dobrar número de cores

Modelo	P4 3GHz	GF 5900	Core2 Quad 3GHz
Ano	2004	2004	2007
Qtde. núcleos.	1	8	4
Perf. aritmética	6 GFLOPS	20 GFLOPS	96 GFLOPS
Largura banda	6 GB/sec	25 GB/sec	21 GB/sec
Valor			$ 1100

Fonte: Kurt Akeley, Stanford

FURB UFRGS

Paralelismo em Desktops

- Softwares mais rápidos precisam paralelismo
- 2x mais rápido ao dobrar número de cores

Modelo	P4 3GHz	GF 5900	Core2 Quad 3GHz	GF 8800
Ano	2004	2004	2007	2006
Qtde. núcleos.	1	8	4	128
Perf. aritmética	6 GFLOPS	20 GFLOPS	96 GFLOPS	330 GFLOPS
Largura banda	6 GB/sec	25 GB/sec	21 GB/sec	104 GB/sec
Valor			$ 1100	$ 550

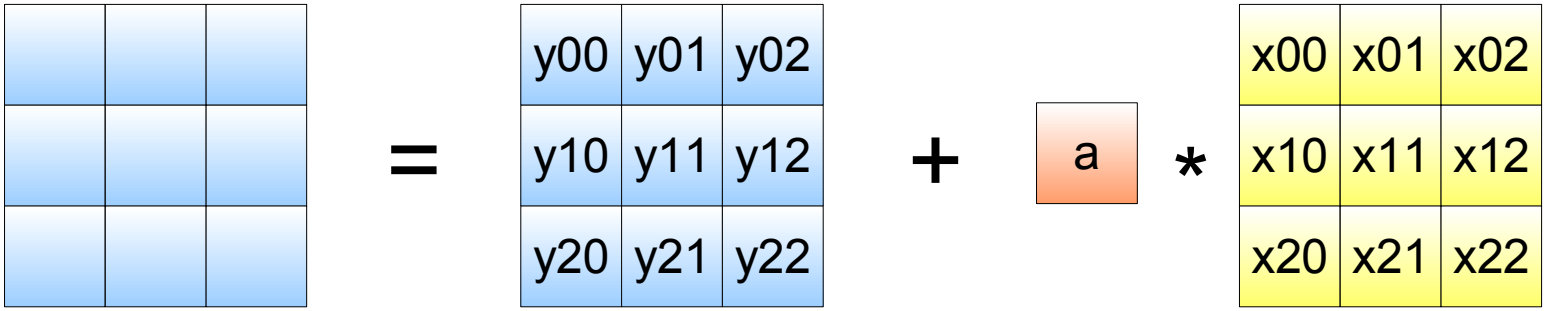
Fonte: Kurt Akeley, Stanford

FURB UFRGS

Características GPU

- Diversos processadores paralelos (SIMD)
- Alta eficiência para processamento de streams
- Grande largura de banda para memória
- Suporte a ponto flutuante 32 bits

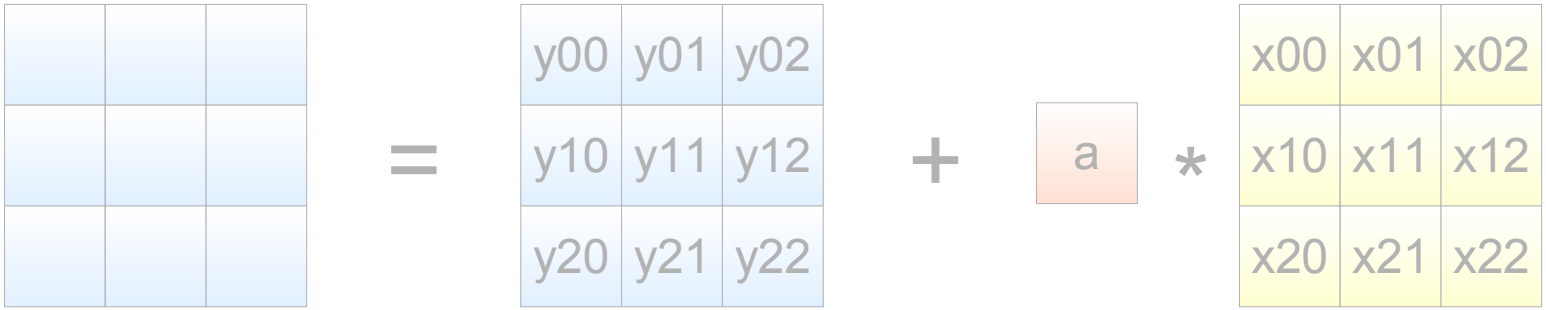
Estudo de Caso #1 (Matrizes)



$$y = y + a.x$$

FURB UFRGS

Estudo de Caso #1 (Matrizes)



$$y = y + a.x$$

Implementação CPU

```
for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
        y[i][j] = y[i][j] + a * x[i][j];
```

FURB UFRGS

Stream de Dados

CPU

y00	y01	y02
y10	y11	y12
y20	y21	y22

Matrizes

(i, j)

Índices

FURB UFRGS

Stream de Dados

CPU

y00	y01	y02
y10	y11	y12
y20	y21	y22

Matrizes

GPU

y00	y01	y02
y10	y11	y12
y20	y21	y22

Texturas

(i, j)

Índices

(s, t)

Coordenadas

FURB UFRGS

Kernel de Processamento

CPU

```
for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
        y[i][j] = y[i][j] + a * x[i][j];
```

Loops internos

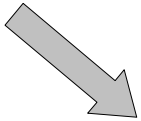
FURB UFRGS

Kernel de Processamento

CPU

```
for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
        y[i][j] = y[i][j] + a * x[i][j];
```

Loops internos



GPU

```
y = texRect(texY, coord);
x = texRect(texX, coord);
return y + a * x;
```

Programas de fragmentos

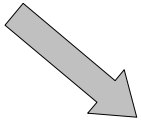
FURB UFRGS

Kernel de Processamento

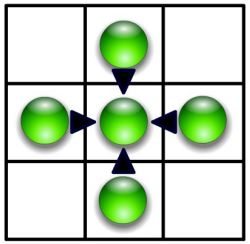
CPU

```
for (int i=0; i<3; i++)
    for (int j=0; j<3; j++)
        y[i][j] = y[i][j] + a * x[i][j];
```

Loops internos



GPU



Gather

```
y = texRect(texY, coord);
x = texRect(texX, coord);
return y + a * x;
```

Programas de fragmentos

FURB UFRGS

Executando *Kernel*

- Configurar OpenGL para desenhar 1:1
- Desenhar quadrilátero (e.g. `3x3`)

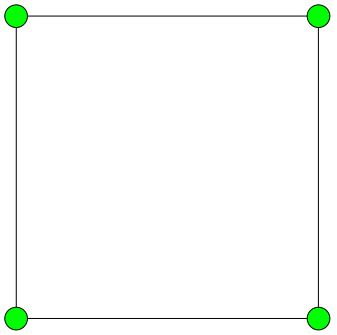
```
glBegin(GL_QUADS);
    glVertex2f(0, 0);
    glVertex2f(0, 3);
    glVertex2f(3, 3);
    glVertex2f(3, 0);
glEnd();
```

FURB UFRGS

Executando *Kernel*

- Configurar OpenGL para desenhar 1:1
- Desenhar quadrilátero (e.g. `3x3`)

```
glBegin(GL_QUADS);
    glVertex2f(0, 0);
    glVertex2f(0, 3);
    glVertex2f(3, 3);
    glVertex2f(3, 0);
glEnd();
```

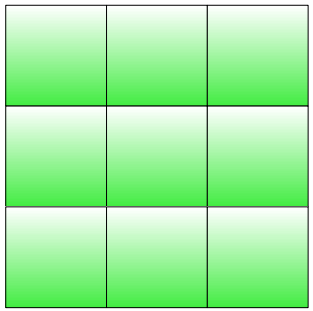


FURB UFRGS

Executando *Kernel*

- Configurar OpenGL para desenhar 1:1
- Desenhar quadrilátero (e.g. `3x3`)

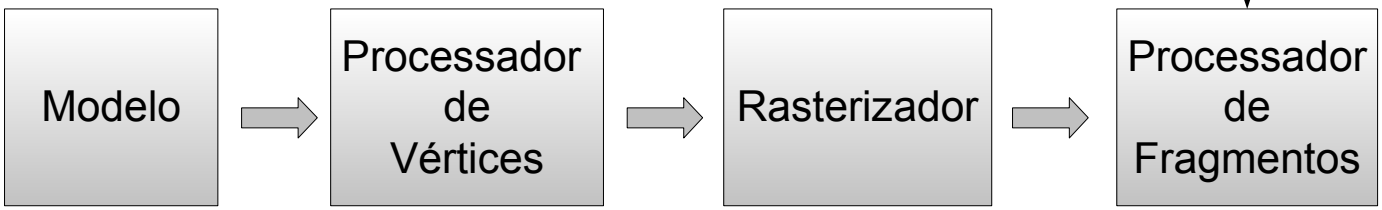
```
glBegin(GL_QUADS);
    glVertex2f(0, 0);
    glVertex2f(0, 3);
    glVertex2f(3, 3);
    glVertex2f(3, 0);
glEnd();
```



FURB UFRGS

Overview do Mapeamento

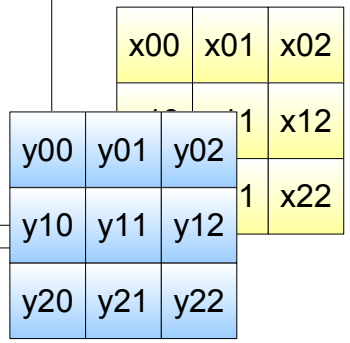
```
float4 somaKernel(uniform samplerRect texY,
                  uniform samplerRect texX,
                  float2 coord: WPOS,
                  uniform float a)
{
    y = texRect(texY, coord);
    x = texRect(texX, coord);
    return y + a * x;
}
```

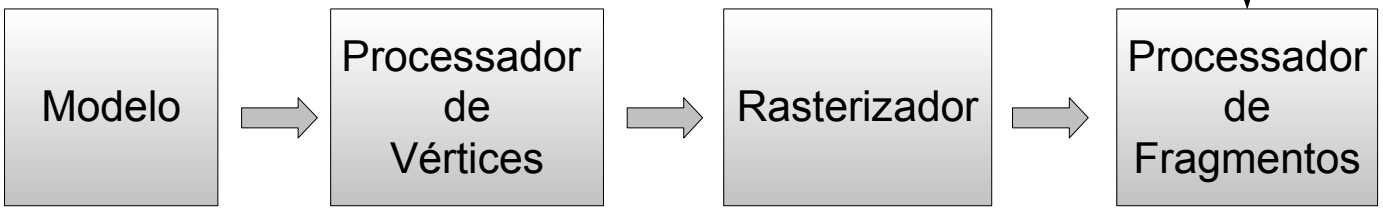


FURB UFRGS

Overview do Mapeamento

```
float4 somaKernel(uniform samplerRect texY,
                  uniform samplerRect texX,
                  float2 coord: WPOS,
                  uniform float a)
{
    y = texRect(texY, coord);
    x = texRect(texX, coord);
    return y + a * x;
}
```

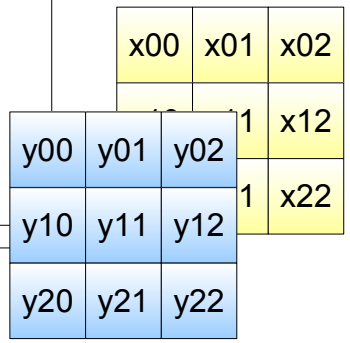




FURB UFRGS

Overview do Mapeamento

```
float4 somaKernel(uniform samplerRect texY,
                  uniform samplerRect texX,
                  float2 coord: WPOS,
                  uniform float a)
{
    y = texRect(texY, coord);
    x = texRect(texX, coord);
    return y + a * x;
}
```



FURB UFRGS

Limitações Computacionais

- Sem *heap*
- Sem *stack*
- Sem *scatter* (`a[i] = b`)
- Sem operações de redução (`max, min, sum`)
- Número limitado de *outputs*

FURB UFRGS

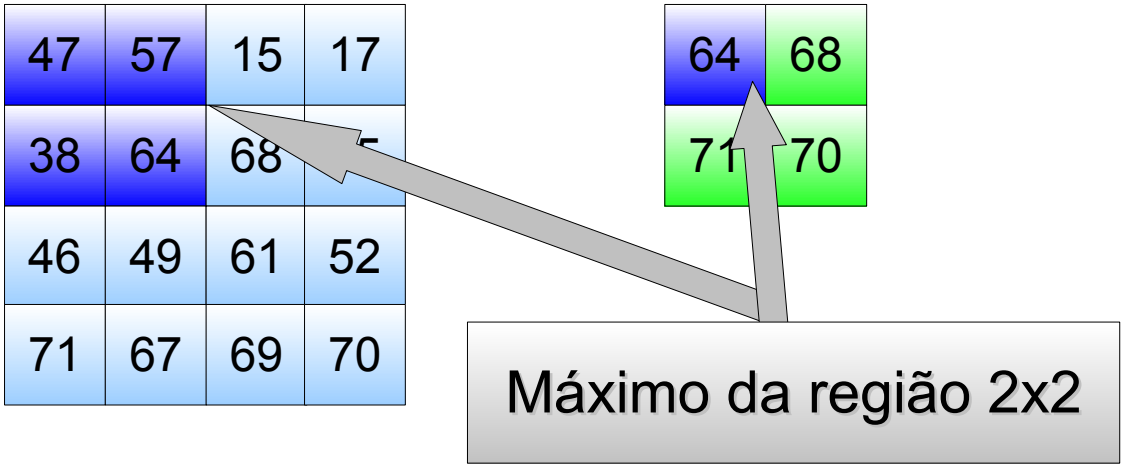
Emulando Operações de Redução

- Utilizam todos os elementos (`max, min, sum`)
- Utilizam `log(n)` passadas no pipeline

47	57	15	17
38	64	68	35
46	49	61	52
71	67	69	70

Emulando Operações de Redução

- Utilizam todos os elementos (`max, min, sum`)
- Utilizam `log(n)` passadas no pipeline

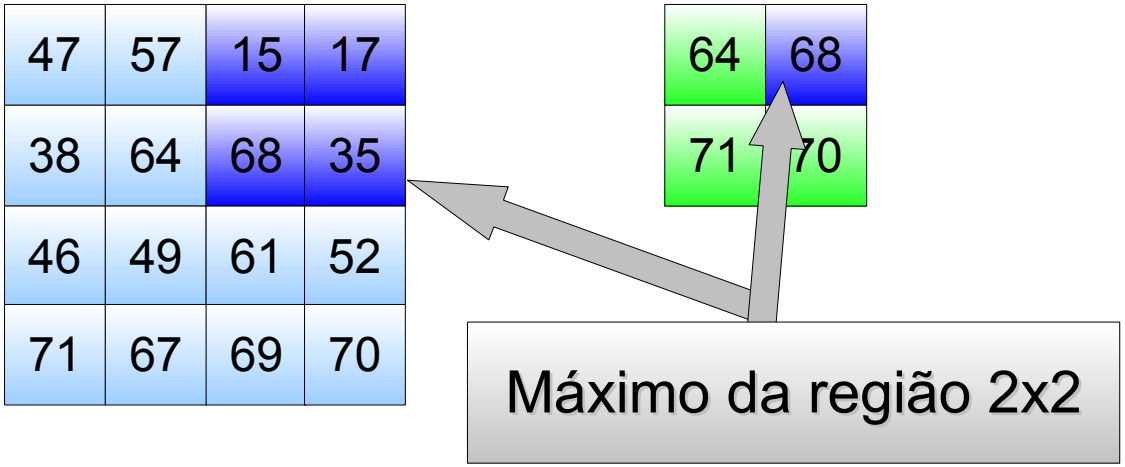


1º passo

FURB UFRGS

Emulando Operações de Redução

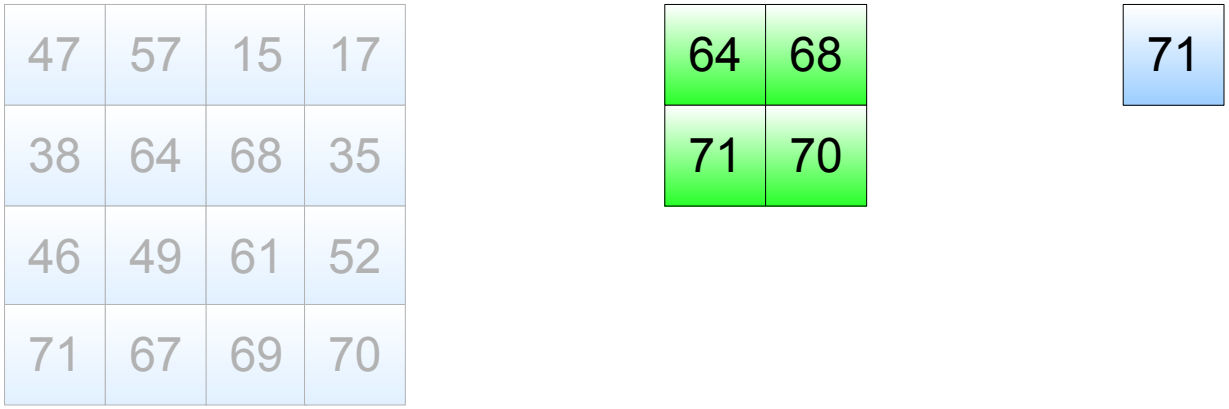
```
float4 v1 = texRect(tex, coord);
float4 v2 = texRect(tex, coord+float2(1,0));
float4 v3 = texRect(tex, coord+float2(1,1));
float4 v4 = texRect(tex, coord+float2(0,1));
return max(v1, max(v2, max(v3, v4)));
```



1º passo

Emulando Operações de Redução

```
float4 v1 = texRect(tex, coord);
float4 v2 = texRect(tex, coord+float2(1,0));
float4 v3 = texRect(tex, coord+float2(1,1));
float4 v4 = texRect(tex, coord+float2(0,1));
return max(v1, max(v2, max(v3, v4)));
```

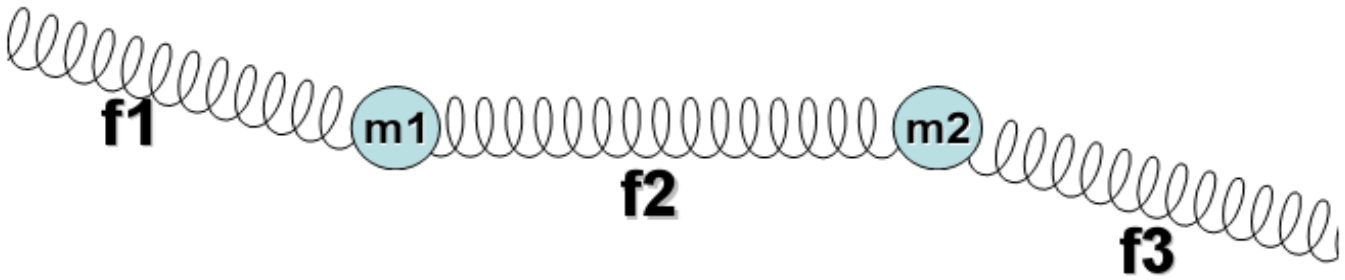


FURB UFRGS

Estudo de Caso #2 (Massa-Mola)

- Sistemas massa-mola
  - particulas conectadas por molas
  - simulação de corpos rígidos

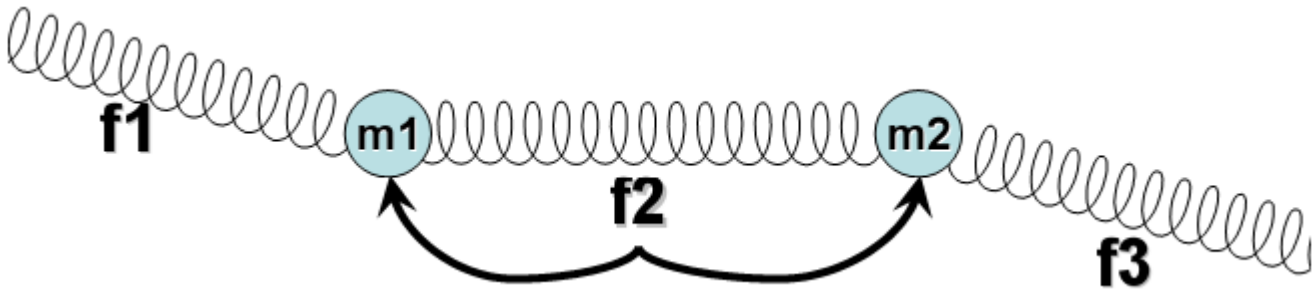
- Posição de cada partícula é dada:
  - velocidade e aceleração (soma das forças aplicadas)
  - força externa = gravidade, colisão, atrito, inércia
  - força interna = força restauradora das molas



FURB UFRGS

Calculando Forças Restauradoras [scatter]

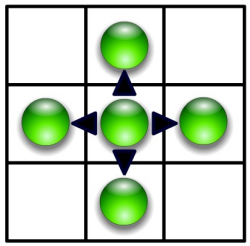
```
para cada mola
    calcule a força restauradora
    diminua a força na partícula esquerda
    soma a força na partícula direita
```



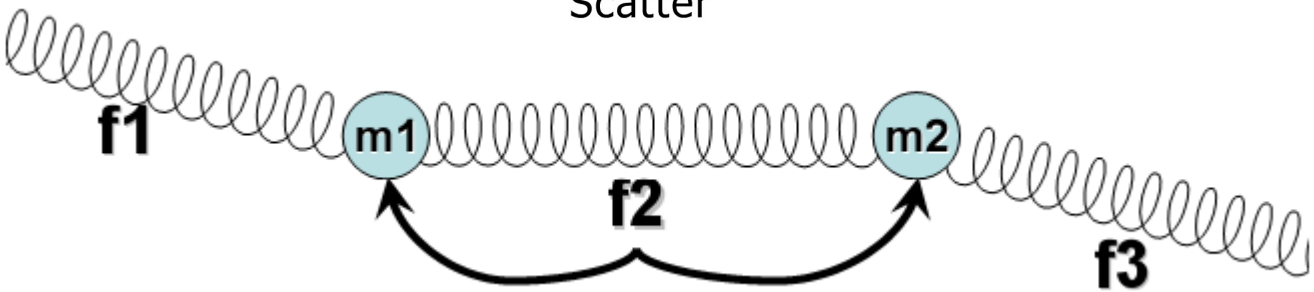
FURB UFRGS

Calculando Forças Restauradoras [scatter]

```
para cada mola
    calcule a força restauradora
    diminua a força na partícula esquerda
    soma a força na partícula direita
```



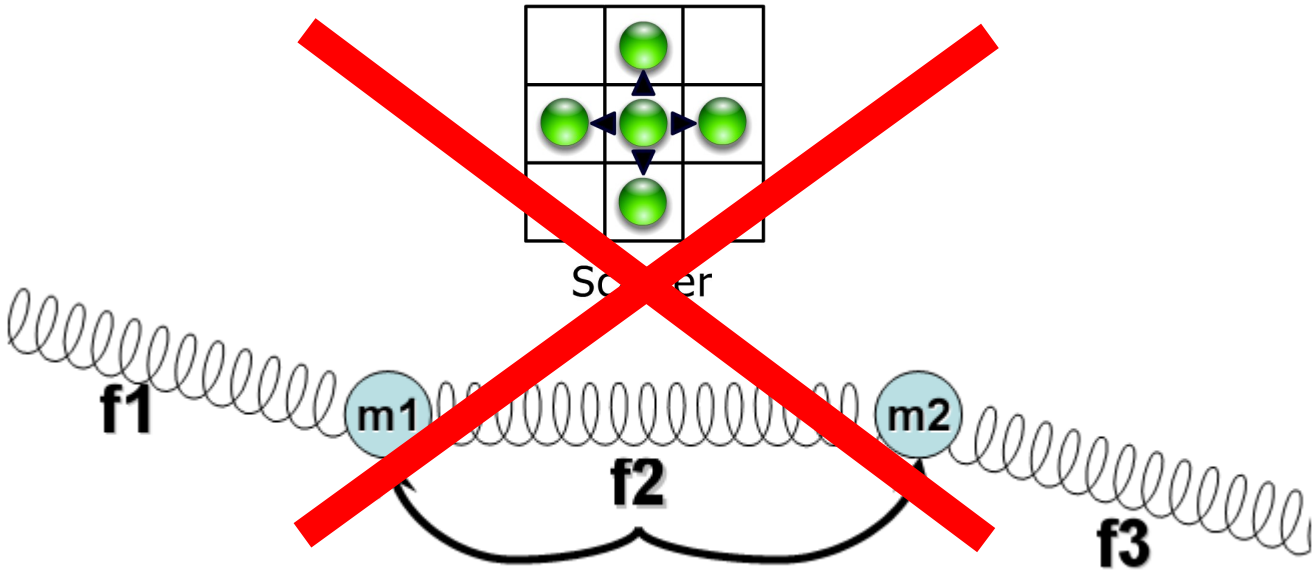
Scatter



FURB UFRGS

Calculando Forças Restauradoras [scatter]

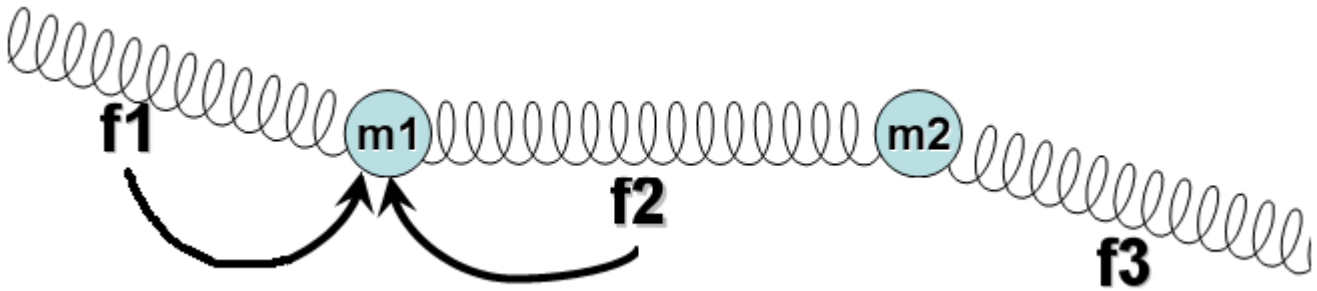
```
para cada mola
    calcule a força restauradora
    diminua a força na partícula esquerda
    soma a força na partícula direita
```



FURB UFRGS

Calculando Forças Restauradoras [gatter]

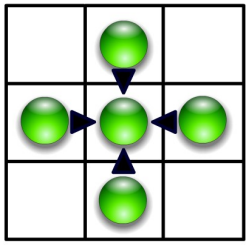
```
para cada partícula
    para cada mola conectada
        calcule a força restauradora
        diminua a força nesta partícula
```



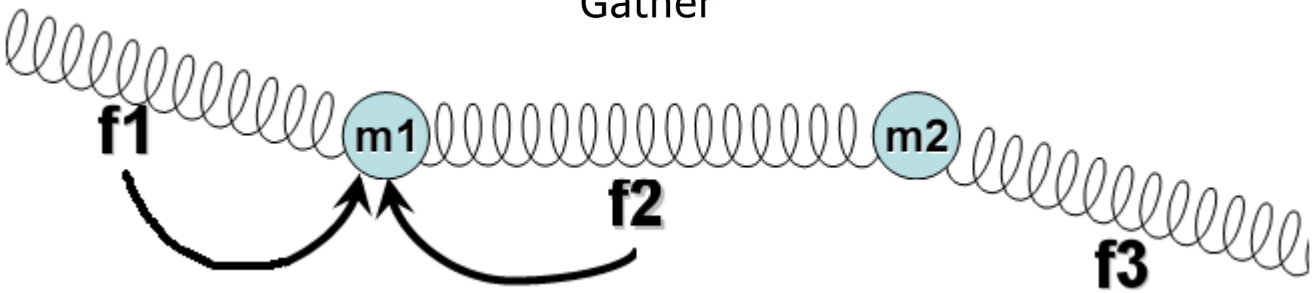
FURB UFRGS

Calculando Forças Restauradoras [gatter]

```
para cada partícula
    para cada mola conectada
        calcule a força restauradora
        diminua a força nesta partícula
```



Gather



FURB UFRGS

Simulando Sistema Massa-Mola

1º passo

```
para cada partícula
    para cada mola conectada
        calcule a força restauradora
        diminua a força nesta partícula
```

2º passo

```
para cada partícula
    some forças restauradoras + forças externas
    calcule a aceleração com estas forças
    calcule a nova velocidade com aceleração
    calcule a nova posição com velocidade
```

FURB UFRGS

Operações de BD [Govindaraju 2004]

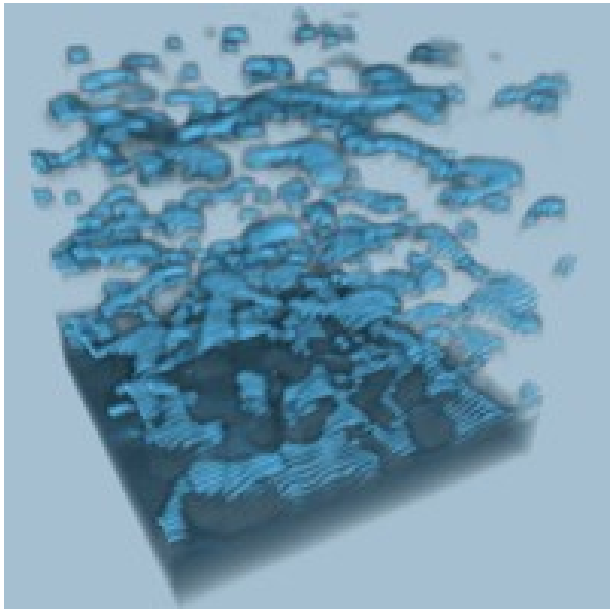
- Avaliação de predicados (`<, >, <=, >=, !=, =`)
  - utiliza *depth-test* para efetuar comparações
- Combinações booleanas (`AND, OR, NOT`)
  - utiliza *stencil-test* para efetuar combinações
- Agregações (`count, max, min, sum, avg`)
  - utiliza *occlusion query* para efetuar *count*
- Speedups de 2~4x

FURB UFRGS

Simulações Baseadas em Física

- Fenômenos físicos pode ser aproximados com PDEs
  - PDEs = equações diferenciais parciais
  - integrar o estado de um objeto ao longo do tempo

- Evaporação d'água
  - estado é temperatura



FURB UFRGS

Simulações Baseadas em Física

- Fenômenos físicos pode ser aproximados com PDEs
  - PDEs = equações diferenciais parciais
  - integrar o estado de um objeto ao longo do tempo

- Evaporação d'água
  - estado é temperatura

- Dinâmica de fluídos [Harris 2004]
  - estado é velocidade e pressão

FURB UFRGS

Simulações Baseadas em Física

- Fenômenos físicos pode ser aproximados com PDEs
  - PDEs = equações diferenciais parciais
  - integrar o estado de um objeto ao longo do tempo

- Evaporação d'água
  - estado é temperatura
- Dinâmica de fluídos [Harris 2004]
  - estado é velocidade e pressão
- Dinâmica de nuvens
  - estado é velocidade, pressão e temperatura

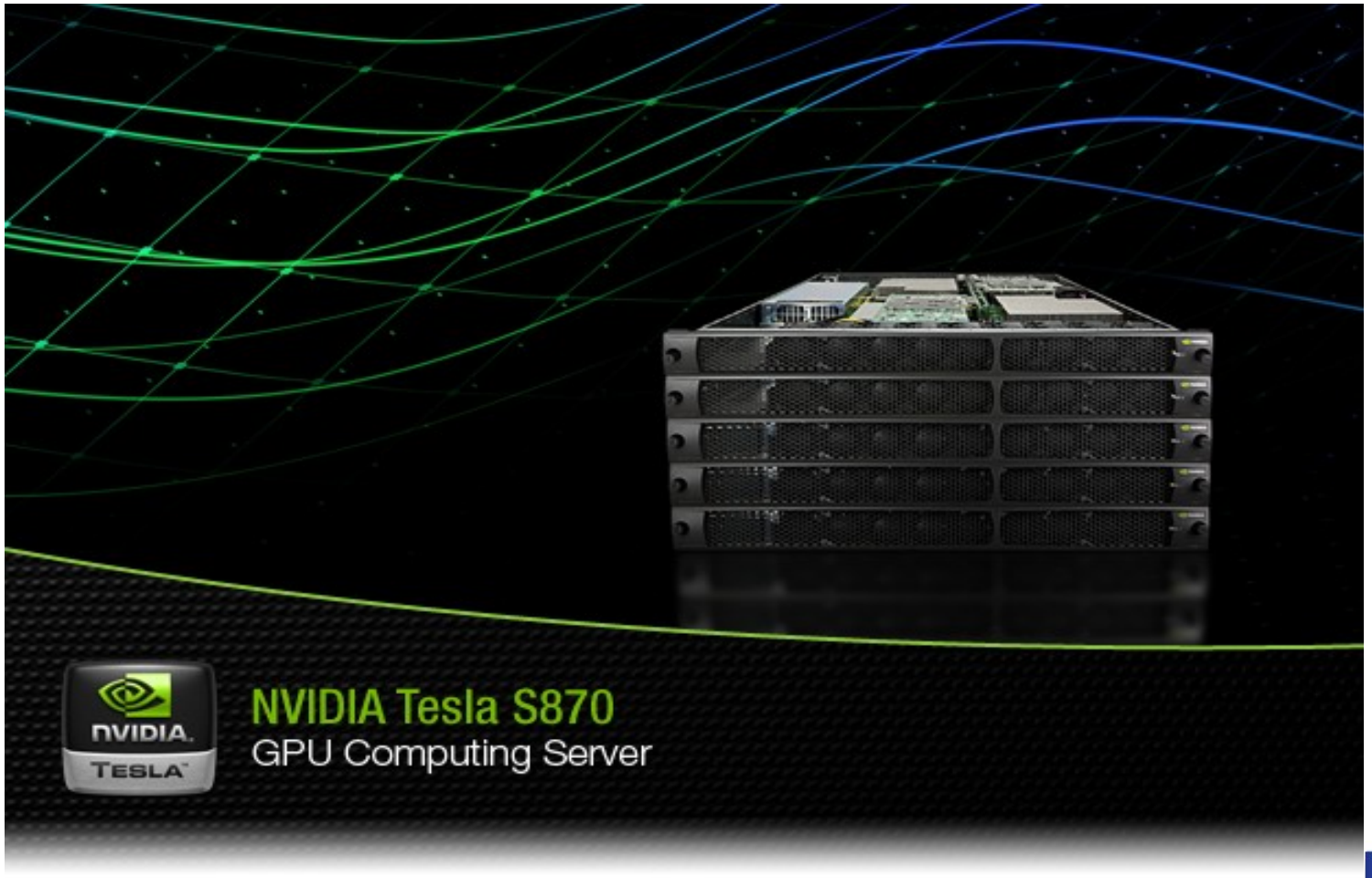
FURB UFRGS

Tecnologias Correntes

- Glift
  - Estruturas de dados genéricas para GPU estilo STL
  - N-trees, estruturas adaptativas, matrizes esparsas, iteradores
- Arquitetura unificada
  - Um tipo de processador para diversas funções
  - Vertíces, geometria, fragmentos
  - CUDA e CTM, biblioteca para abstrair pipeline
- Hardwares atuais
  - NVidia com GeForce 8800 Ultra
  - ATI com Radeon HD 2900 XT
  - NVidia Tesla GPU Computing Server

FURB UFRGS

Tecnologias Correntes



- NVidia Tesla GPU Computing Server

FURB UFRGS

Considerações

- Paralelismo em desktops é uma tendência
- Grandes fornecedores de CPU estão no mercado GPU
  - Intel pesquisando GPUs
  - AMD comprou ATI (Jul / 2006)
- Esforços para simplificar o desenvolvimento
  - CUDA, CTM
- Universidades precisam ensinar paralelismo

FURB UFRGS

Perguntas

Giovane Roslindo Kuhn

grkuhn@gmail.com

FURB UFRGS

Links

NVidia Developer Zone - http://developer.nvidia.com/page/home.html
ATI Developer Zone - http://ati.amd.com/developer/index.html
GPGPU - http://www.gpgpu.org/
OpenGL - http://www.opengl.org/
CMP249 Rendering Avançado - http://grkuhn.googlepages.com/cmp249
NVIDIA CUDA Homepage - http://developer.nvidia.com/object/cuda.html
AMD Stream Processor - http://ati.amd.com/products/streamprocessor/index.html
Chips NVidia - http://www.clubedohardware.com.br/artigos/519
Chips ATI - http://www.clubedohardware.com.br/artigos/520

FURB UFRGS