Projeção de Células baseada em GPU para Visualização Interativa de Volumes

Aluno: André de Almeida Maximo

Orientador: Ricardo Farias

LABORATÓRIO DE COMPUTAÇÃO GRÁFICA
PESC / COPPE / UFRJ

Sumário

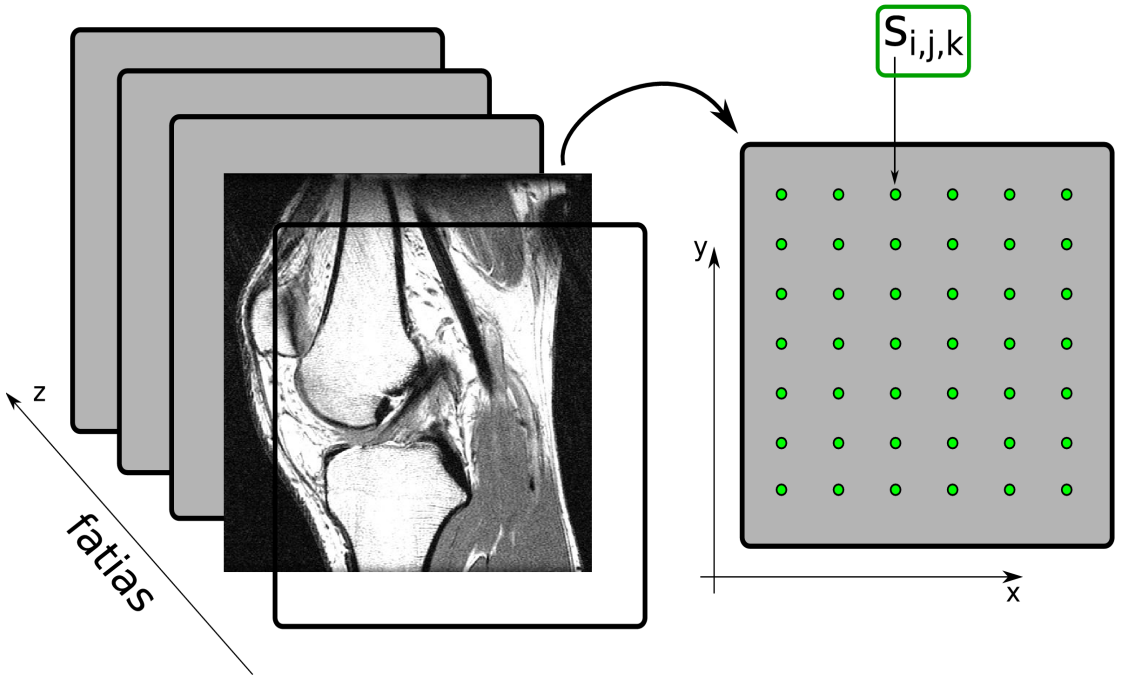
- Introdução
- Trabalhos Relacionados
- Algoritmo de Projeção de Tetraedros
- Algoritmo Proposto
- Resultados
- Conclusões

Introdução

- Aplicações de visualização volumétrica
  - Imagens Médicas
  - Geologia
  - Paleontologia
  - Análise Microscópica
  - Dinâmica de Fluidos
  - Indústria
  - Metereologia
  - Engenharia Civil

Introdução

- Tipos de dados volumétricos
  - Adquiridos
    - Amostrados
    - Simulados
  - Campos
    - Escalares
    - Vetoriais
  - Malhas Regulares ou Irregulares



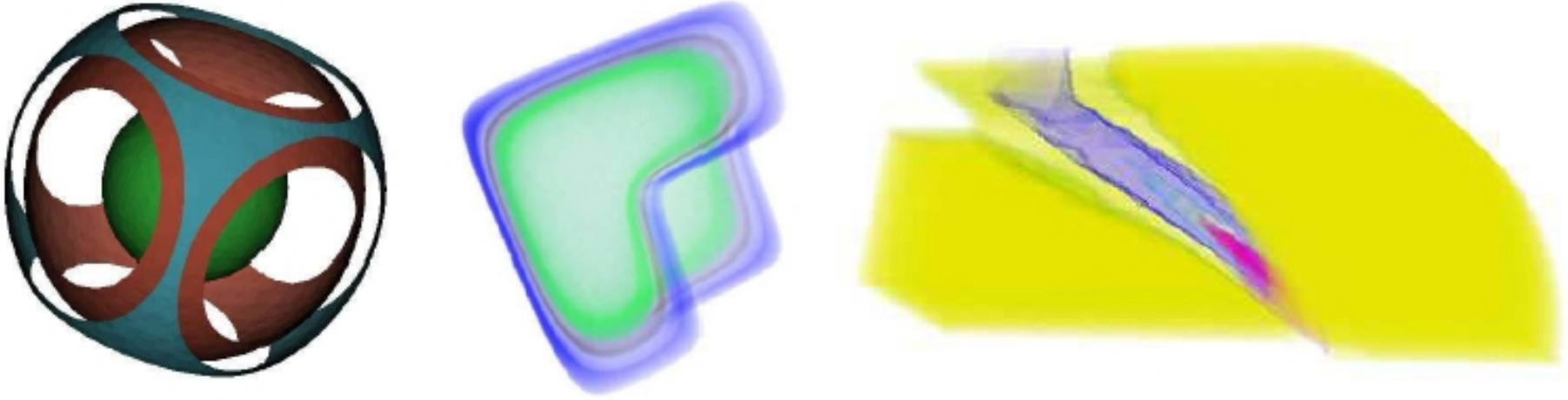
Proposta de Trabalho

- Visualização Interativa
  - Dados Escalares (regulares ou irregulares)
  - Algoritmo de Projeção de Células (tetraedros)
  - Programação em GPU

Trabalhos Relacionados

Hardware-Accelerated Ray Casting – HARC

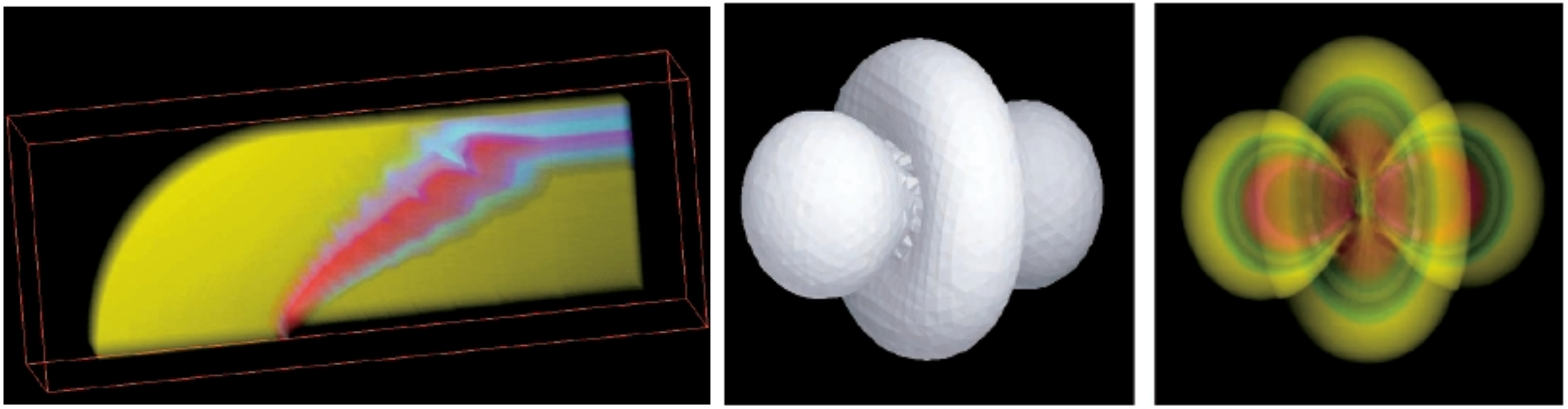
[Weiler et al. 2003]



Trabalhos Relacionados

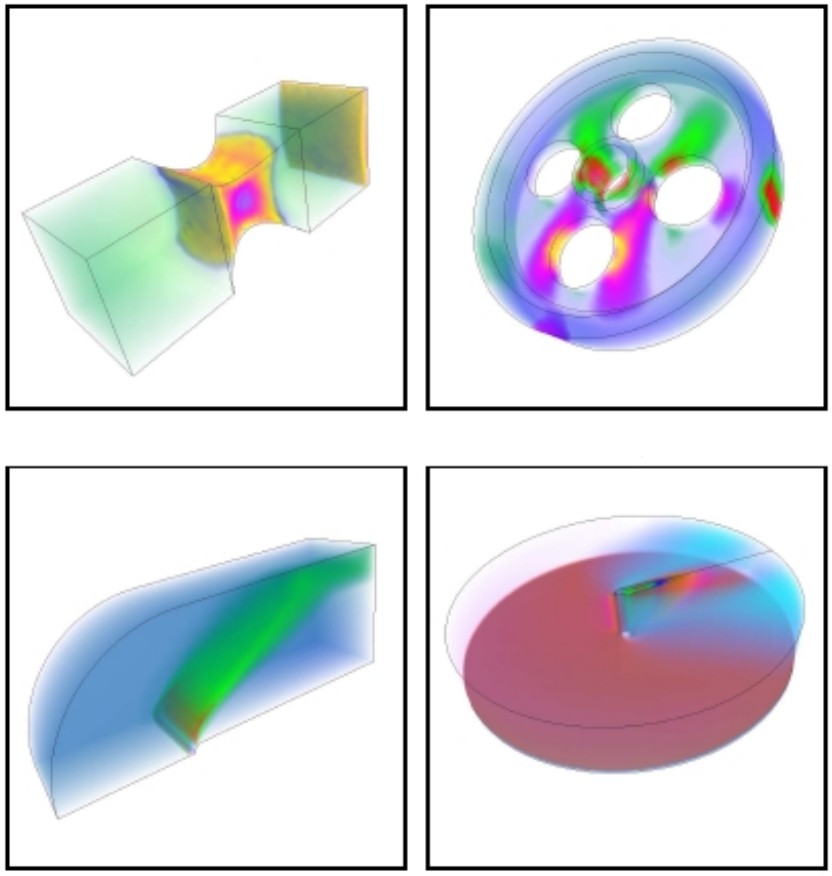
View-independent Cell Projection – VICP

[Weiler et al. 2003]



Trabalhos Relacionados

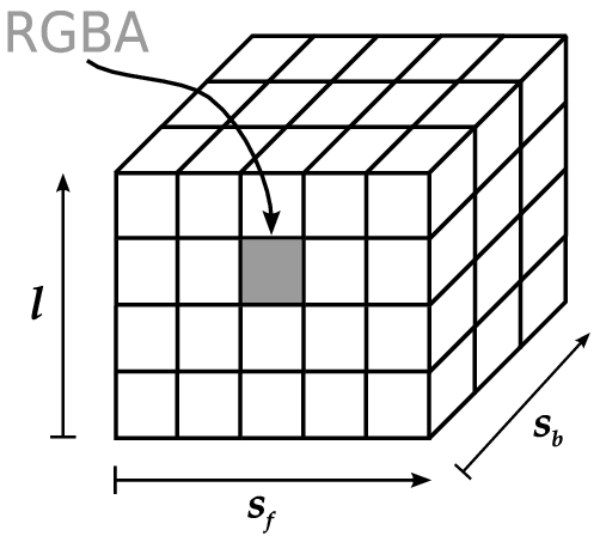
HARC com pré-integração parcial [Espinha e Celes 2005]



Trabalhos Relacionados

Técnica de Pré-integração parcial

[Moreland e Angel 2004]



Pré-integração

Transfer Function

S	{R, G, B, A}
0	{0, 0, 0.9, 0.4}
1	{0, 0.1, 0.8, 0.3}
2	{0, 0.2, 0.7, 0.2}
3	{0, 0.3, 0.6, 0.1}
⋮	⋮
253	{0. , .2, 0, 0.5}
254	{0.8, 0.1, 0, 0.6}
255	{0.9, 0, 0, 0.7}

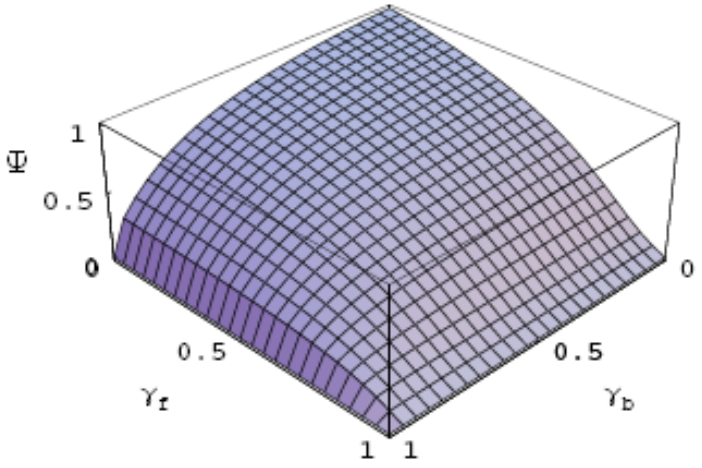
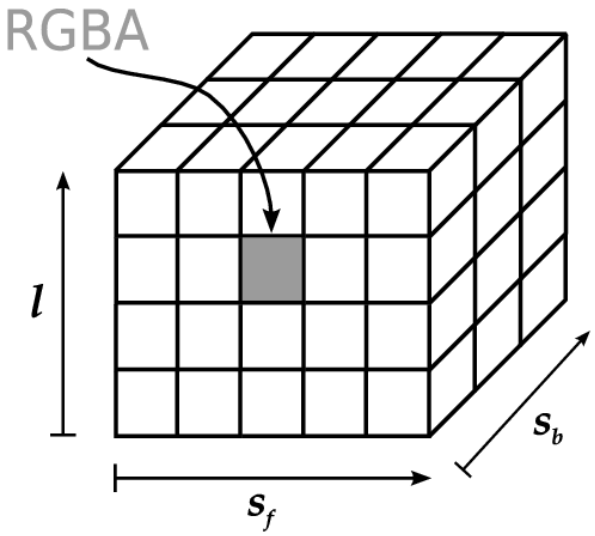


Tabela Ψ

Trabalhos Relacionados

Técnica de Pré-integração parcial

[Moreland e Angel 2004]



Pré-integração

Transfer Function

S	{R, G, B, A}
0	{0, 0, 0.9, 0.4}
1	{0, 0.1, 0.8, 0.3}
2	{0, 0.2, 0.7, 0.2}
3	{0, 0.3, 0.6, 0.1}
⋮	⋮
253	{0. , .2, 0, 0.5}
254	{0.8, 0.1, 0, 0.6}
255	{0.9, 0, 0, 0.7}

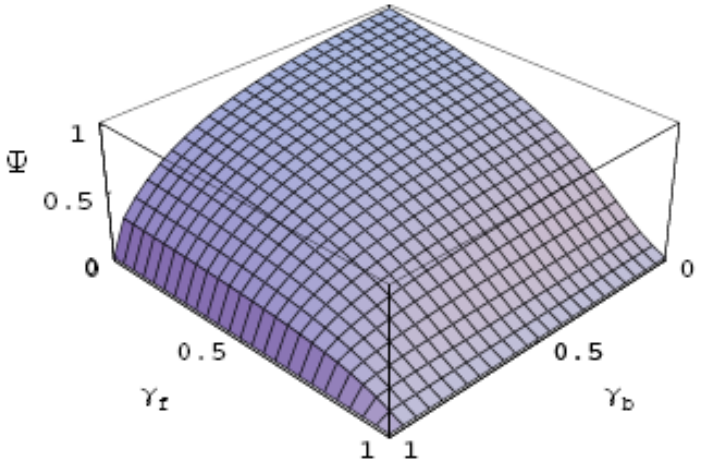


Tabela Ψ

↑ edição interativa da TF

↑ cor da pré-integração

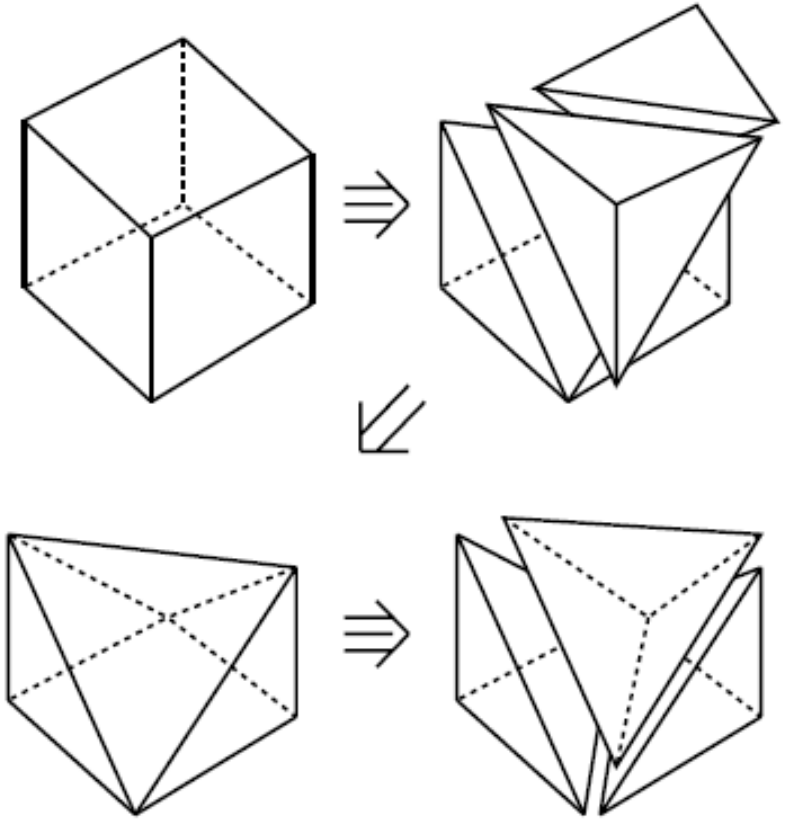
Algoritmo PT

- Projeção de Tetraedros [Shirley & Tuchman 1990]

  1- Dividir hexaedros em tetraedros
  2- Projetar cada tetraedro no plano da imagem
  3- Decompor o tetraedro projetado em triângulos
  4- Computar cor e opacidade nos vértices
  5- Renderizar os triângulos

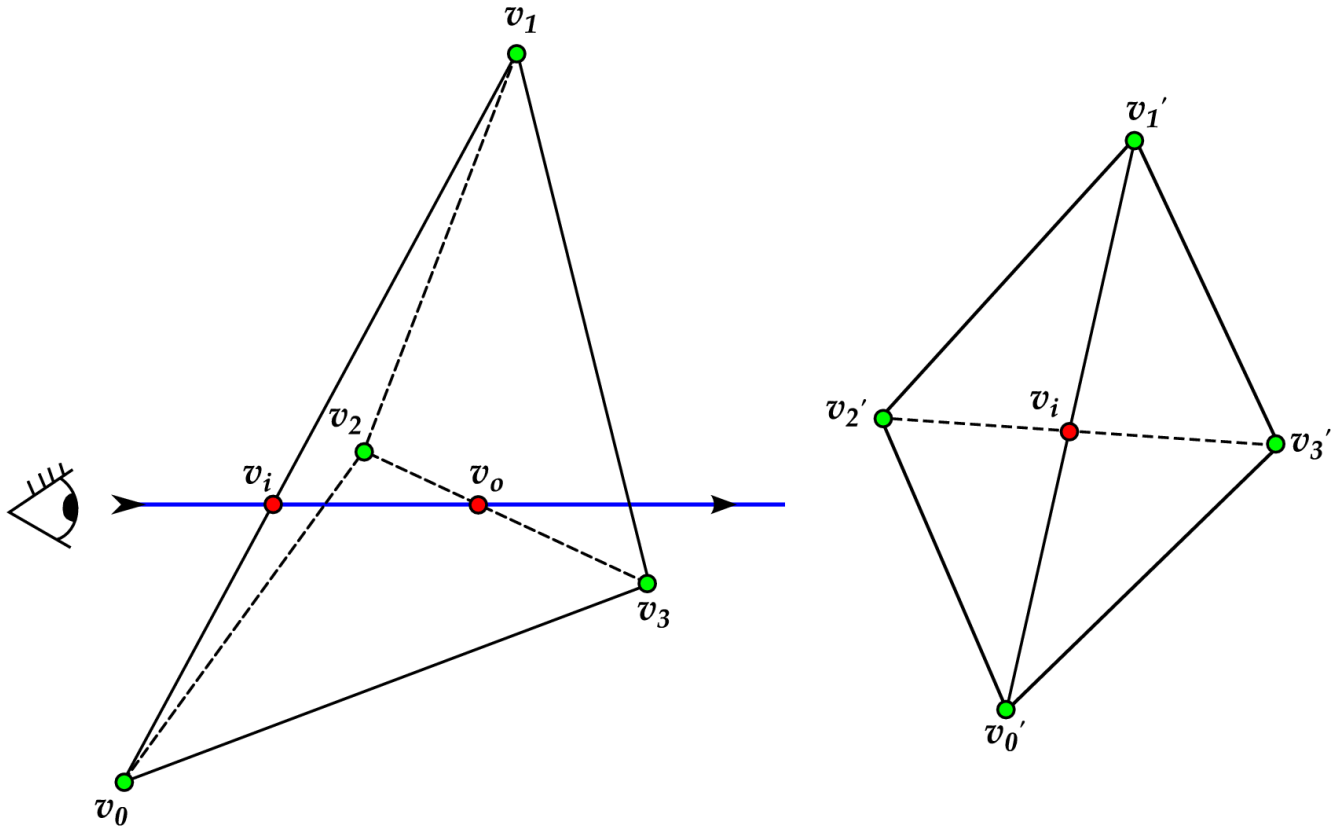
Algoritmo PT

1- Dividir hexaedros em tetraedros



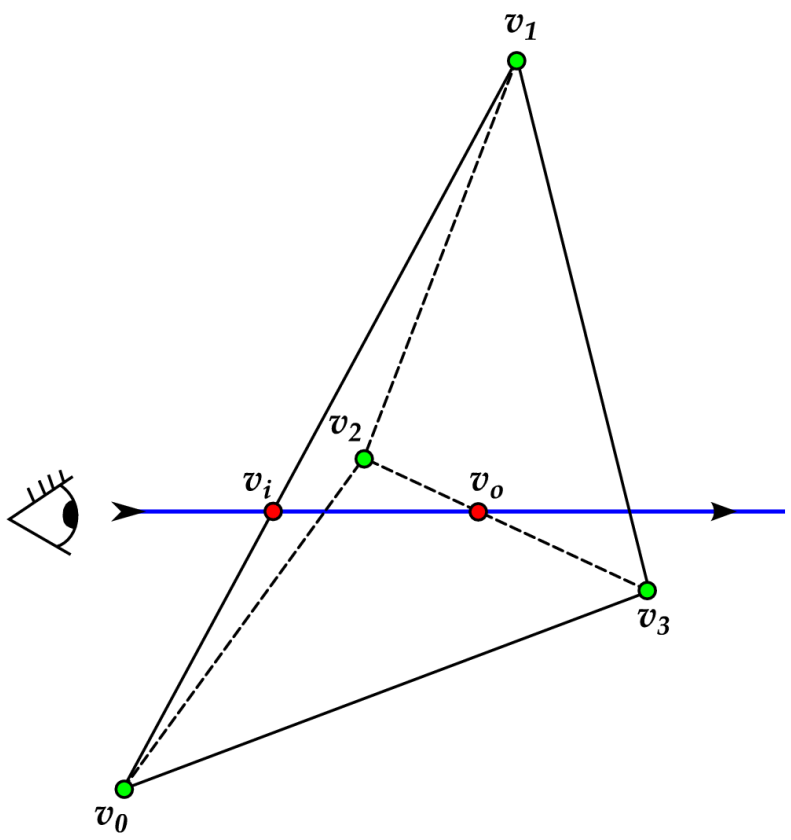
Algoritmo PT

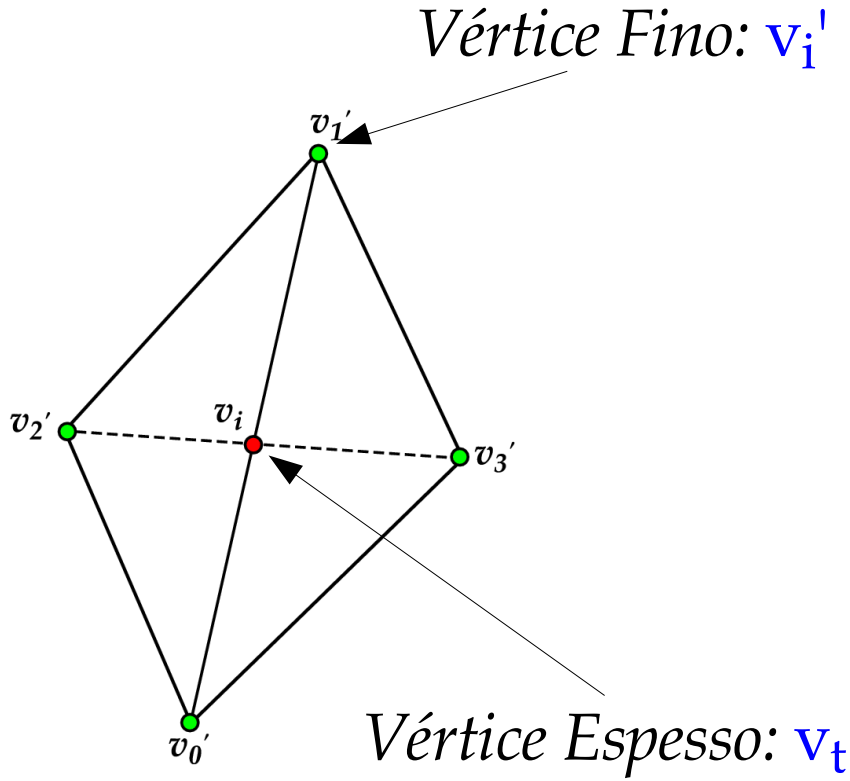
2- Projetar cada tetraedro no plano da imagem



Algoritmo PT

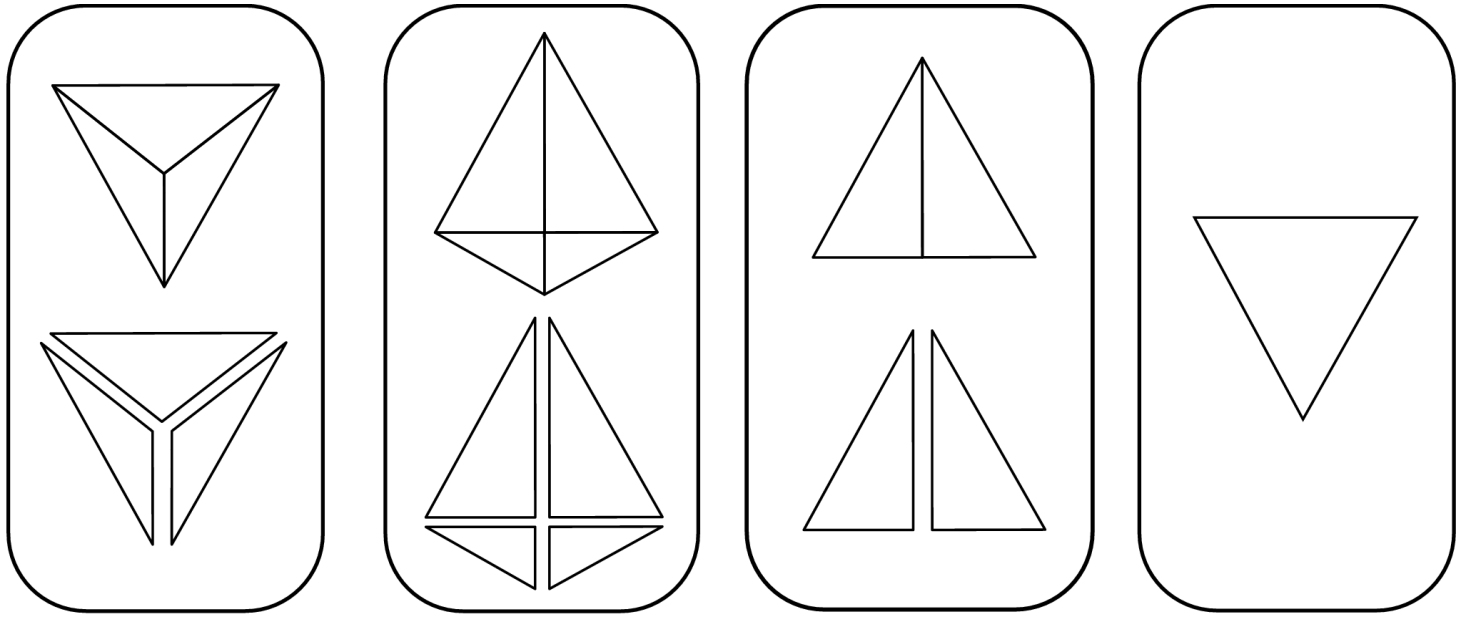
2- Projetar cada tetraedro no plano da imagem





Algoritmo PT

3- Decompor o tetraedro projetado em triângulos



Classe 1
3 Triângulos

Classe 2
4 Triângulos

Classe 3
2 Triângulos

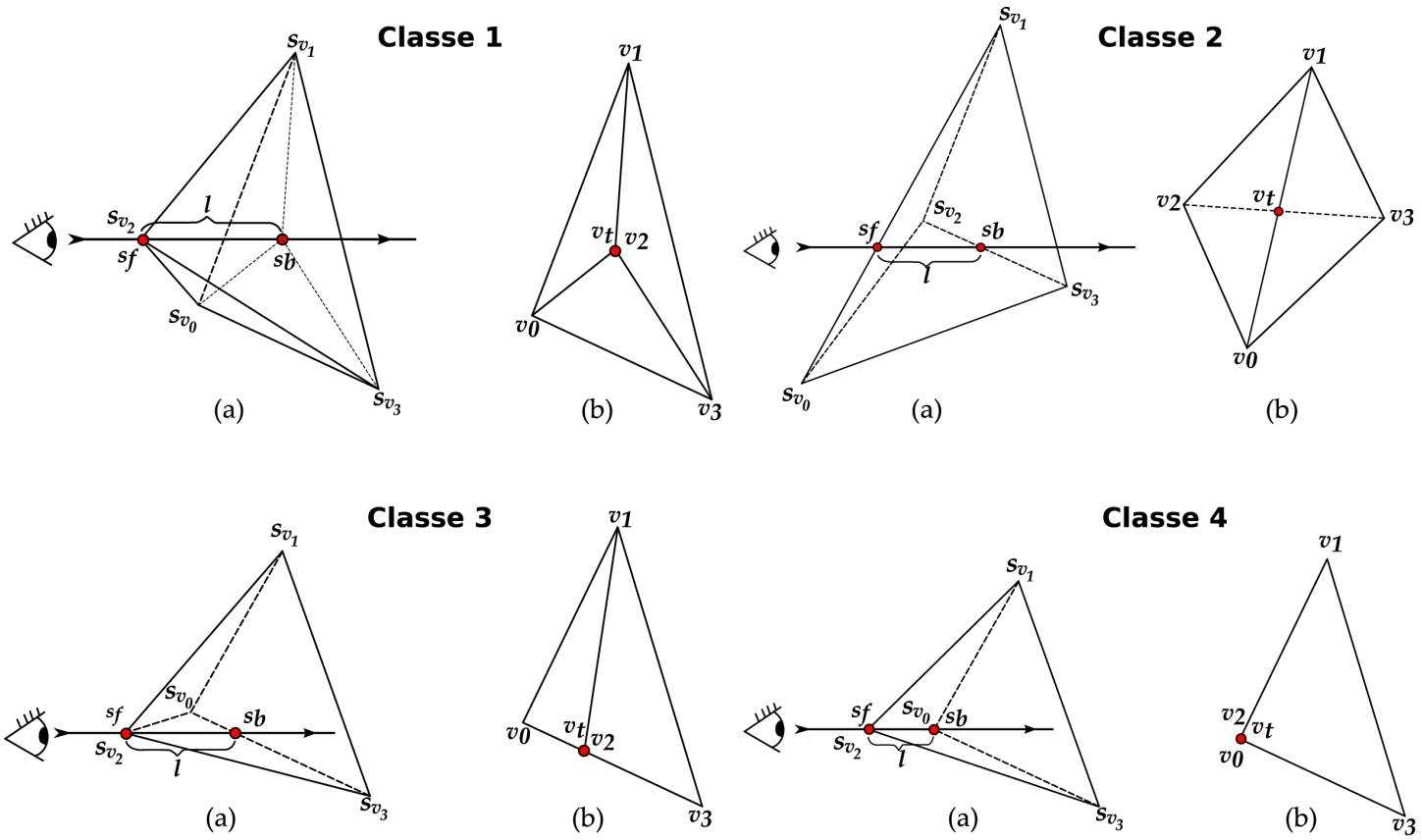
Classe 4
1 Triângulo

Algoritmo PT

4- Computar cor e opacidade nos vértices

$$C = \frac{C(s_f) + C(s_b)}{2}$$

$$\alpha = 1 - e^{-\frac{\tau(s_f)+\tau(s_b)}{2} l}$$

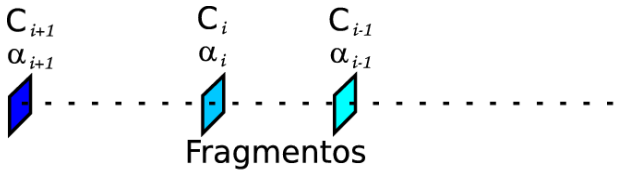


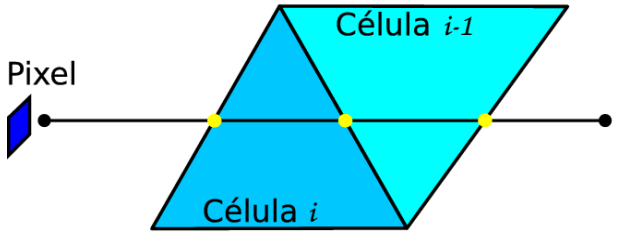
Algoritmo PT

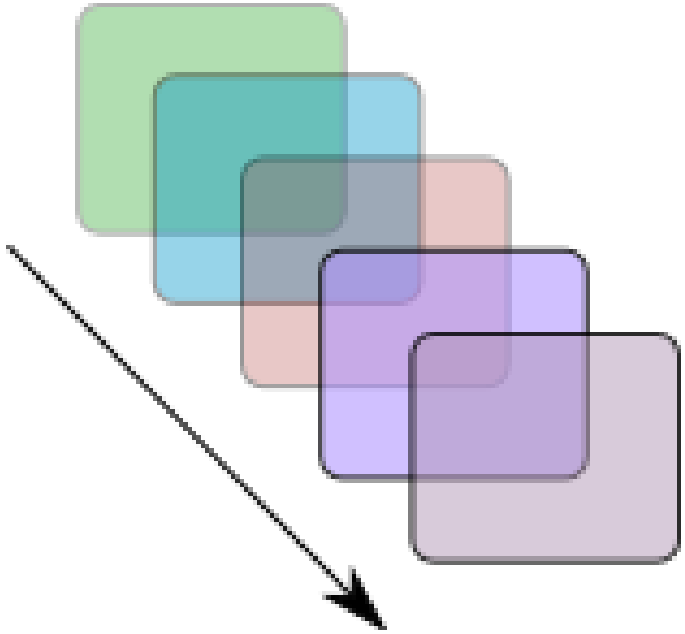
5- Renderizar os triângulos

$$C_{i+1} = \alpha_i \, C_i + (1 - \alpha_i) \, C_{i-1}$$

$$\alpha_{i+1} = \alpha_i + \alpha_{i-1}$$







Algoritmo PT em GPU

- GPU Accelerated Projected Tetrahedra – GATOR [Wylie et al. 2002]

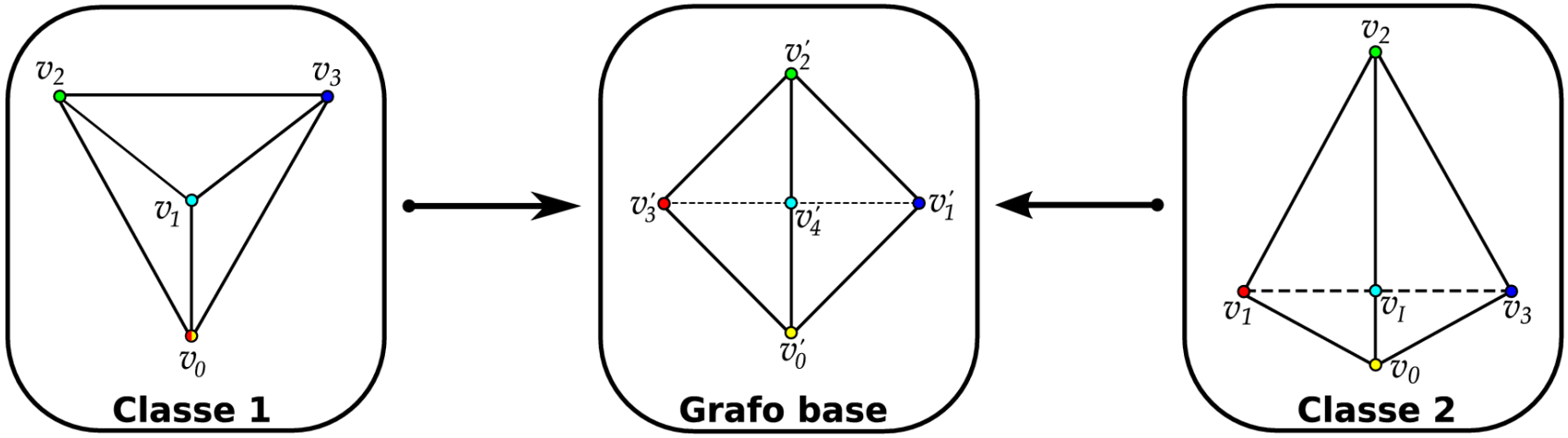
  1- Mapeia os triângulos em GPU (Grafo Base)

  2- Classificação dos tetraedros em casos

  3- Computação da cor e opacidade (intersecção)

Algoritmo PT em GPU

1- Mapeia os triângulos em GPU (Grafo Base)



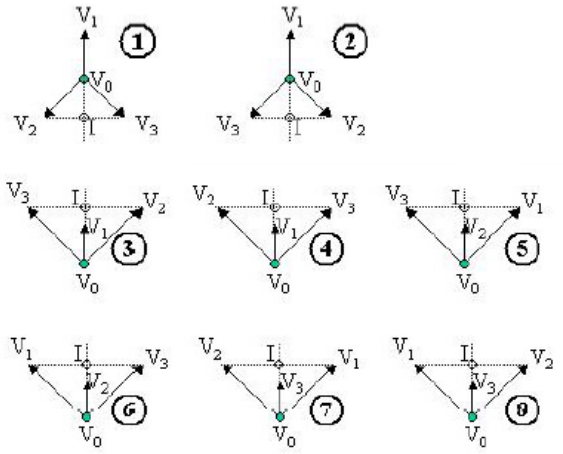
Algoritmo PT em GPU

2- Classificação dos tetraedros em casos

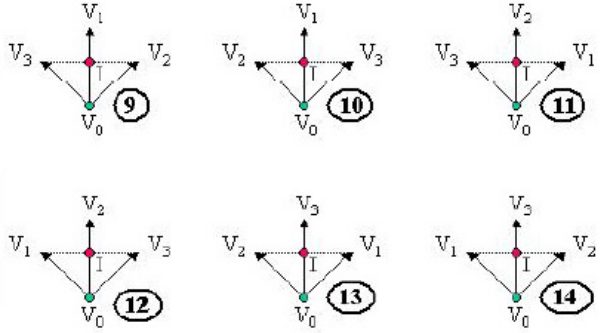
$$vec1 = v_1 - v_0$$
$$vec2 = v_2 - v_0$$
$$vec3 = v_3 - v_0$$

$$teste1 = (vec1 \times vec2).z > 0$$
$$teste2 = (vec1 \times vec3).z < 0$$
$$teste3 = (\text{distância de } v_0 \text{ para } v_M - \text{ distância de } v_0 \text{ para } v_I) > 0$$
$$teste4 = vec1.z > 0$$

Classe 1 com permutações

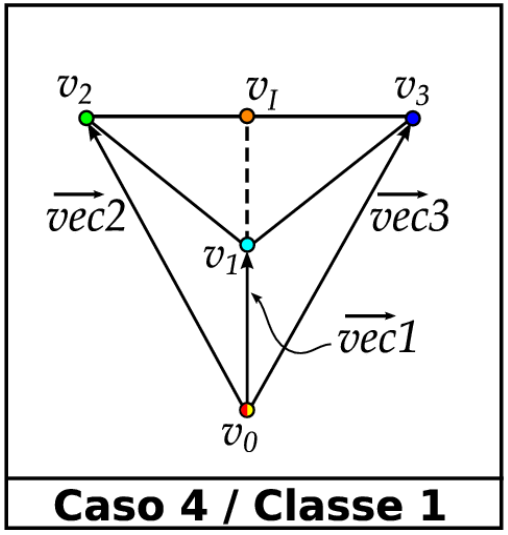


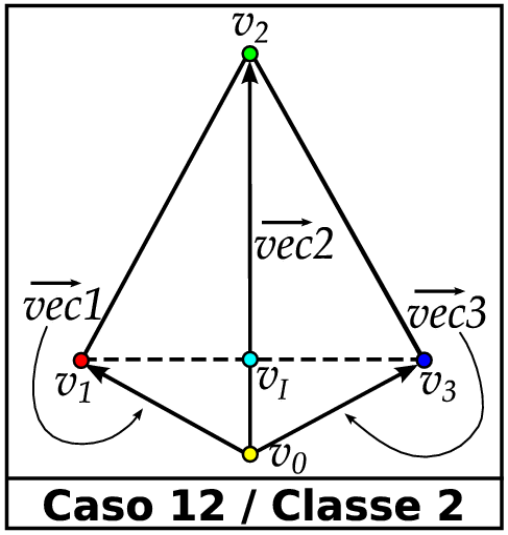
Classe 2 com permutações



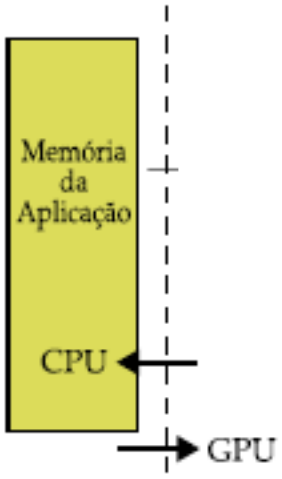
Algoritmo PT em GPU

3- Computação da cor e opacidade (intersecção)

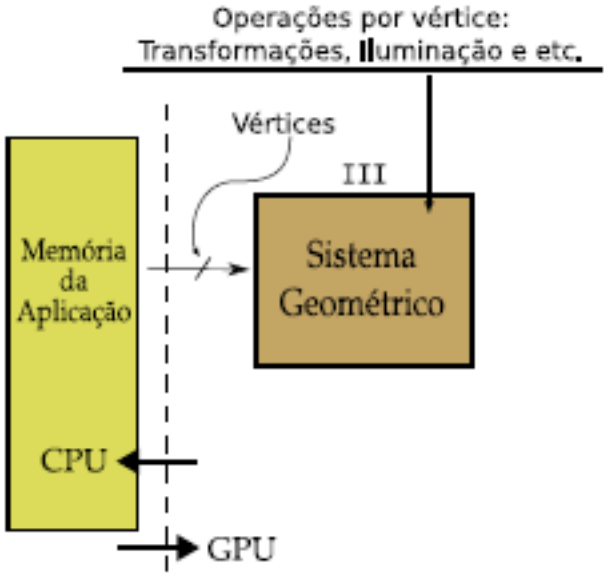




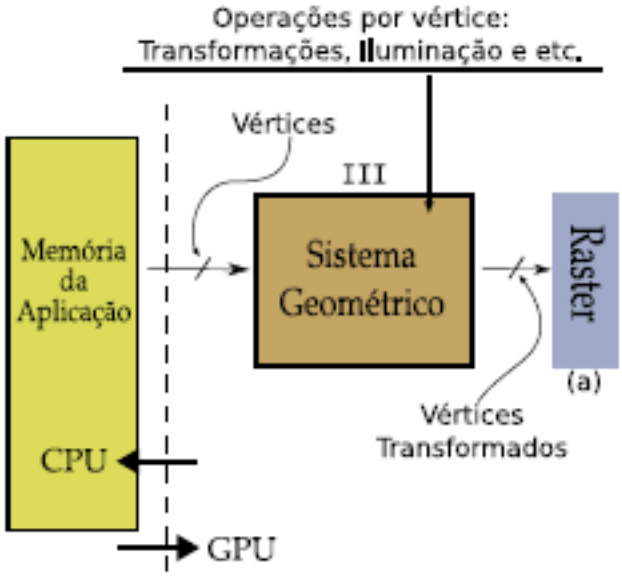
Algoritmos em GPU



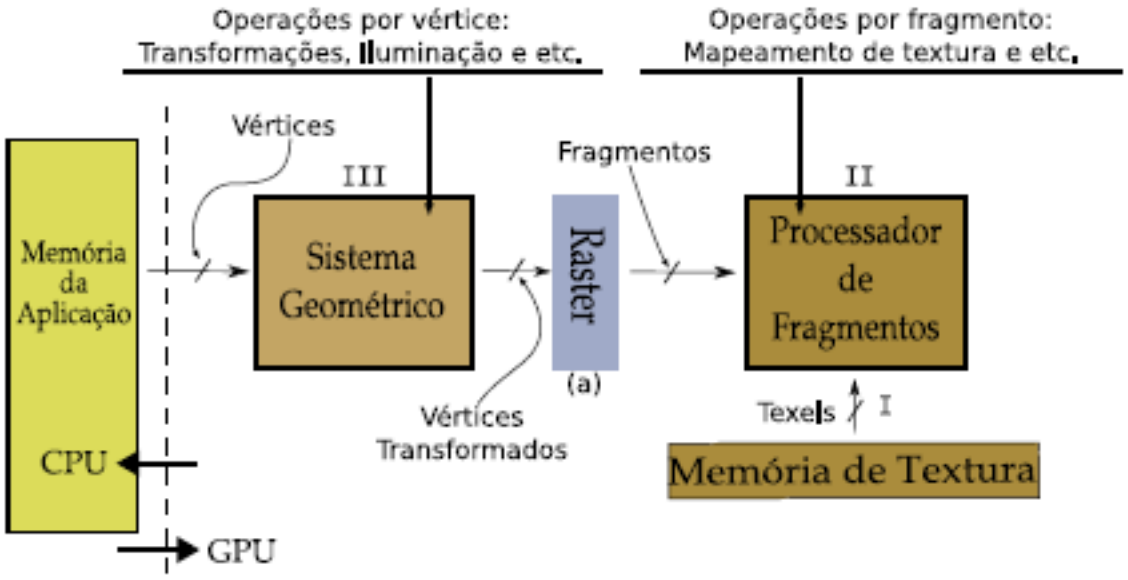
Algoritmos em GPU



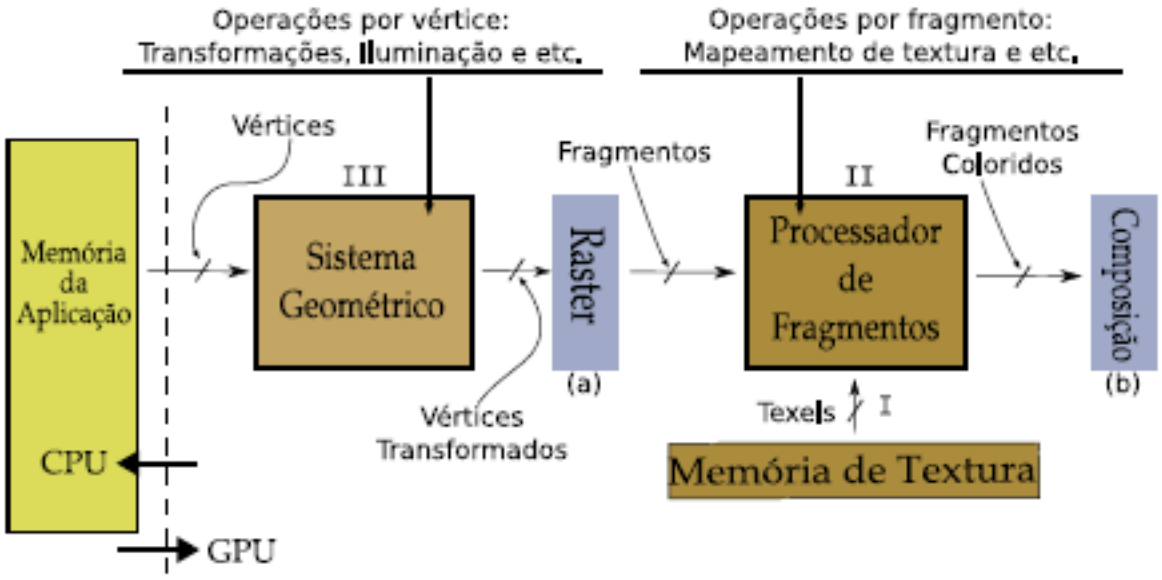
Algoritmos em GPU



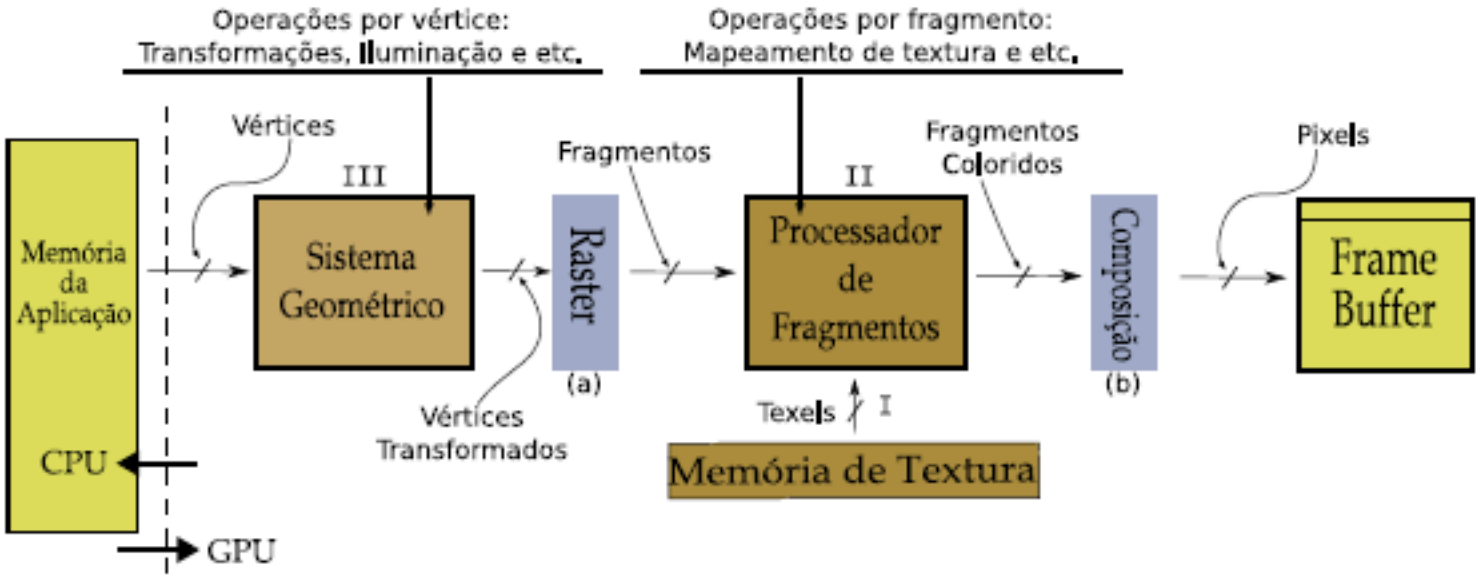
Algoritmos em GPU



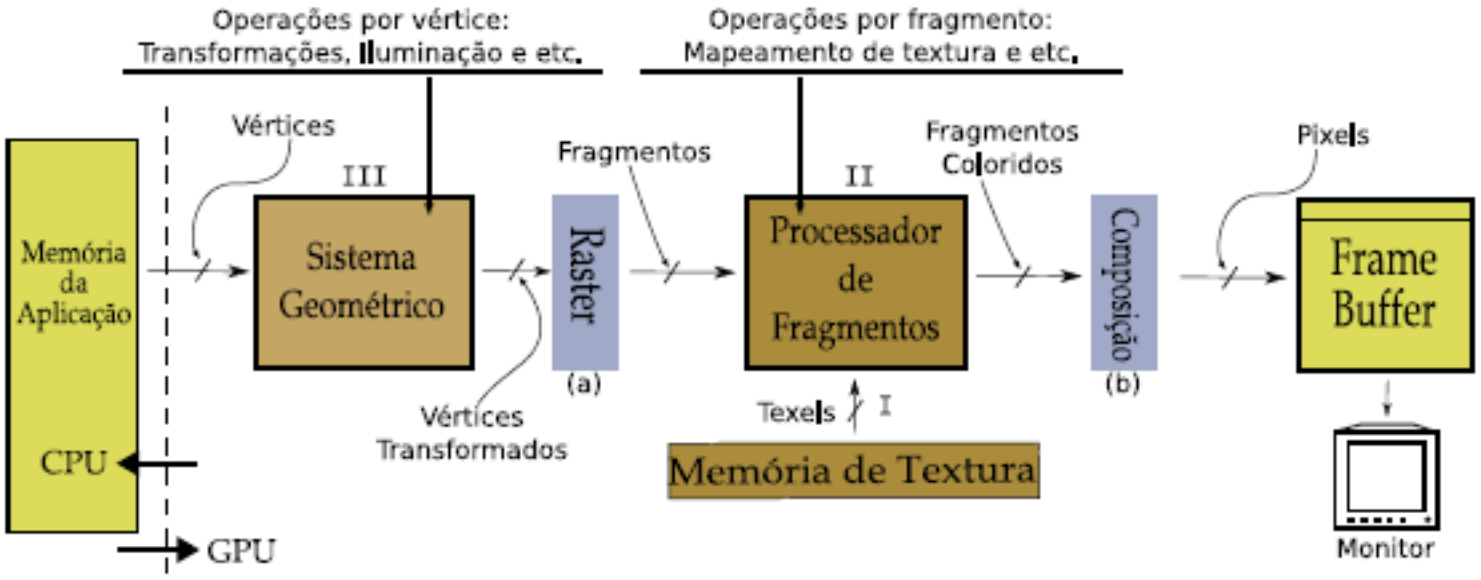
Algoritmos em GPU



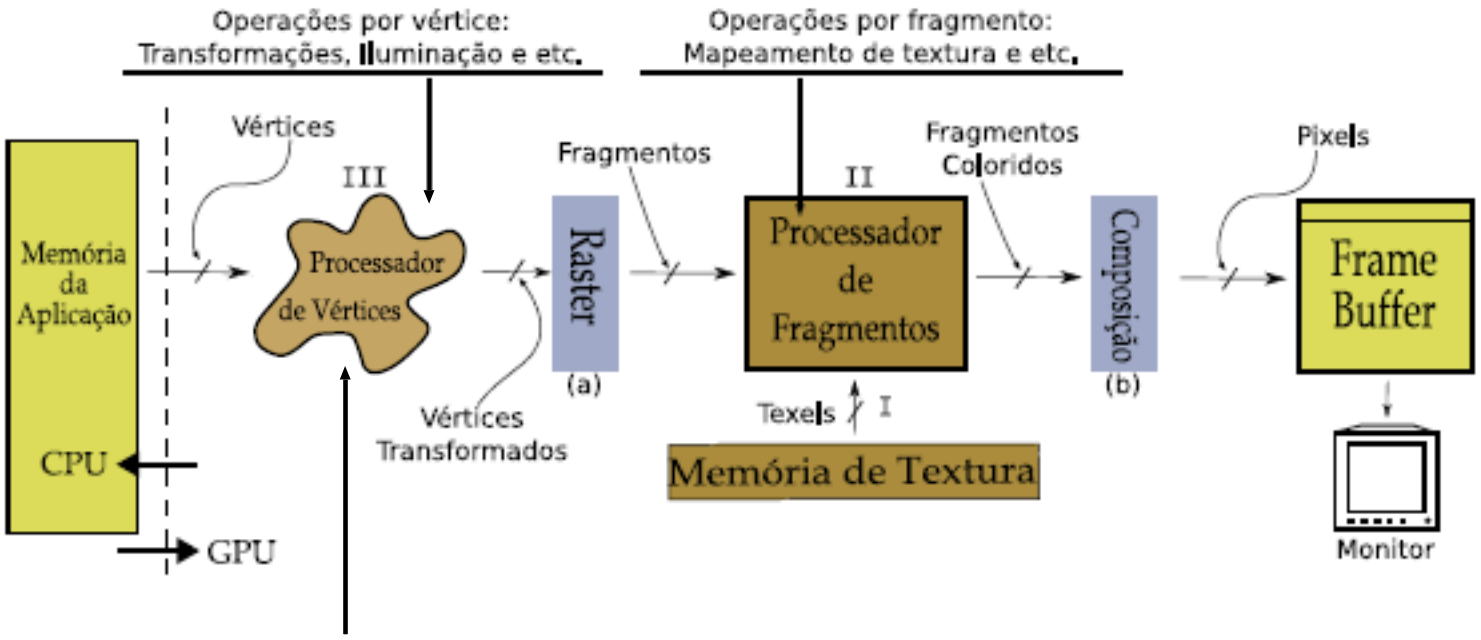
Algoritmos em GPU



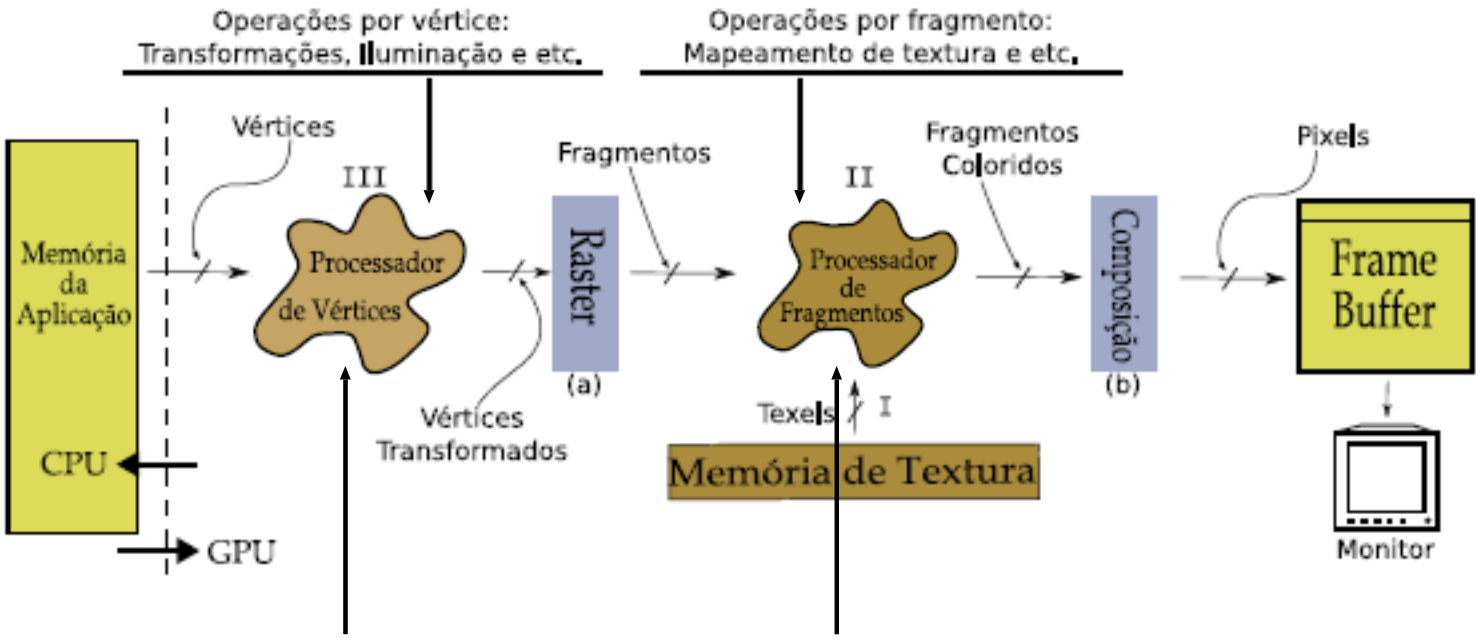
Algoritmos em GPU



Algoritmos em GPU

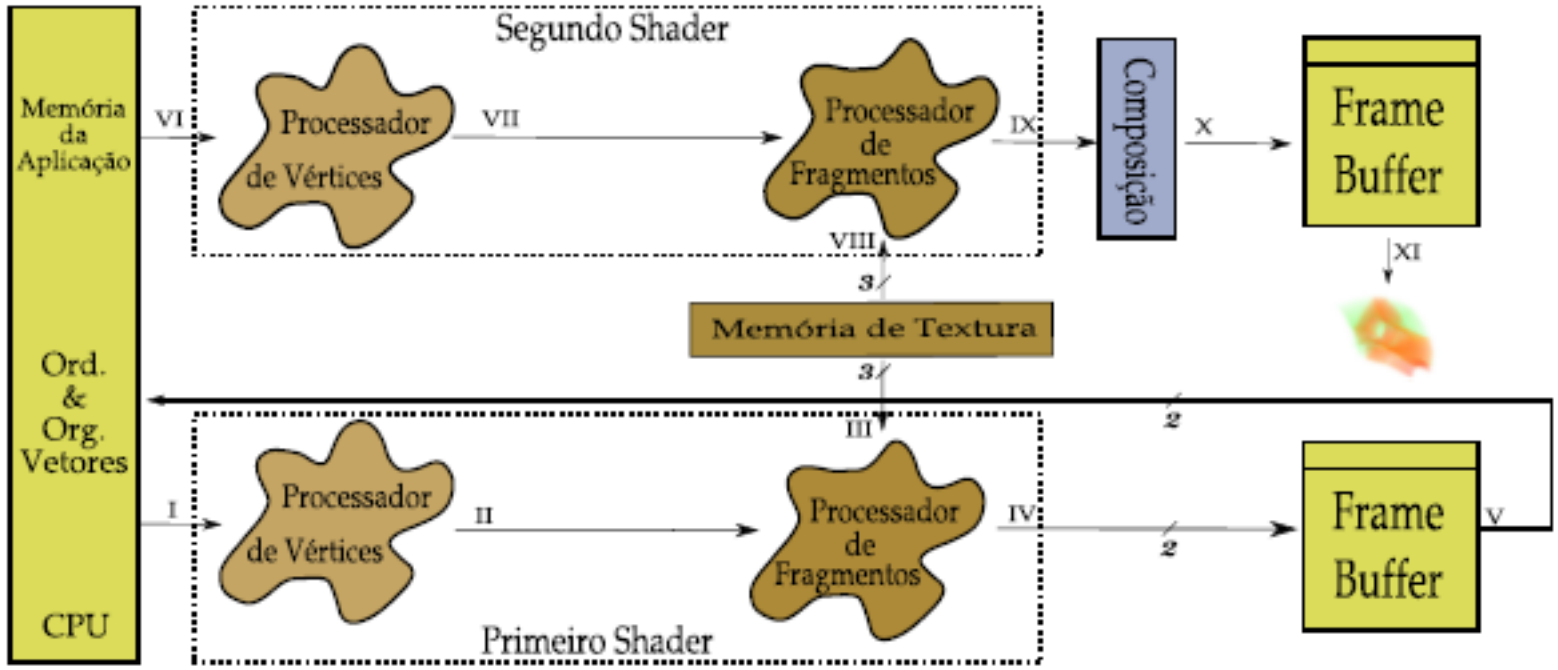


Algoritmos em GPU



Algoritmo proposto – Visão Geral

2 passos principais em GPU

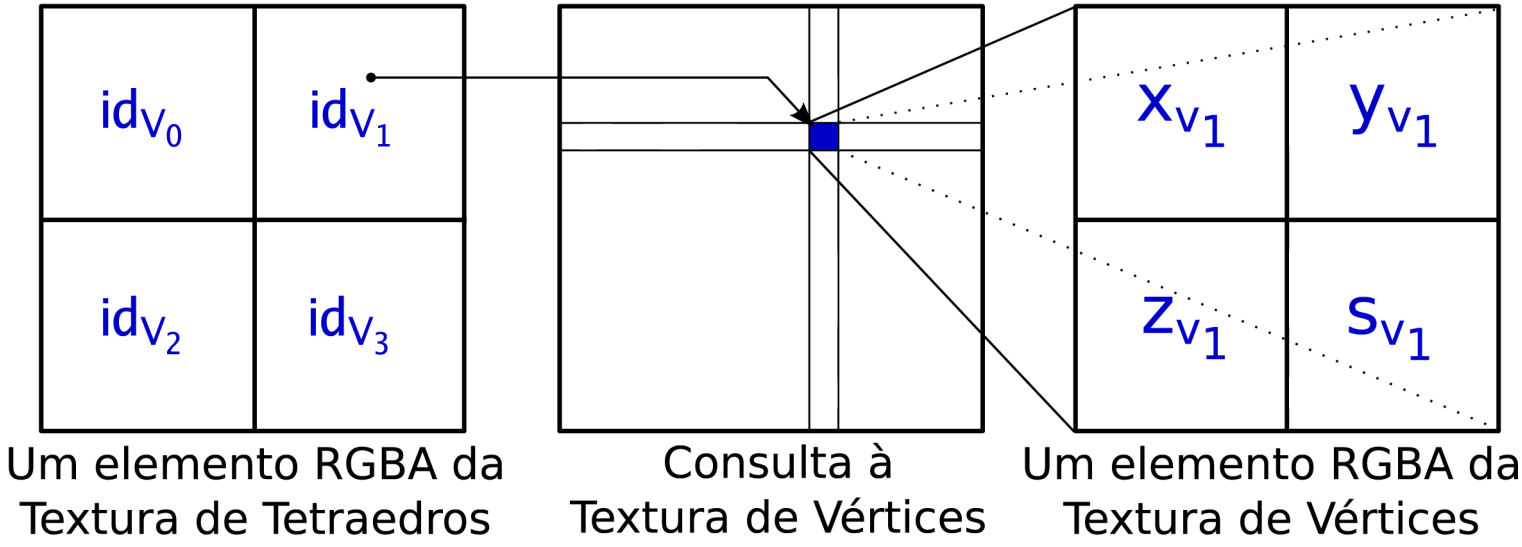


Passos de pré-processamento

- Preparação da memória da GPU
  - Construção de texturas a partir do modelo:
    - Textura de Vértices
    - Textura de Tetraedros
    - Textura da Função de Transferência
  - Pré-compilação de texturas independentes:
    - Textura de Classificação
    - Textura de Exponenciais
    - Textura de Pré-Integração Parcial (tabela Ψ)

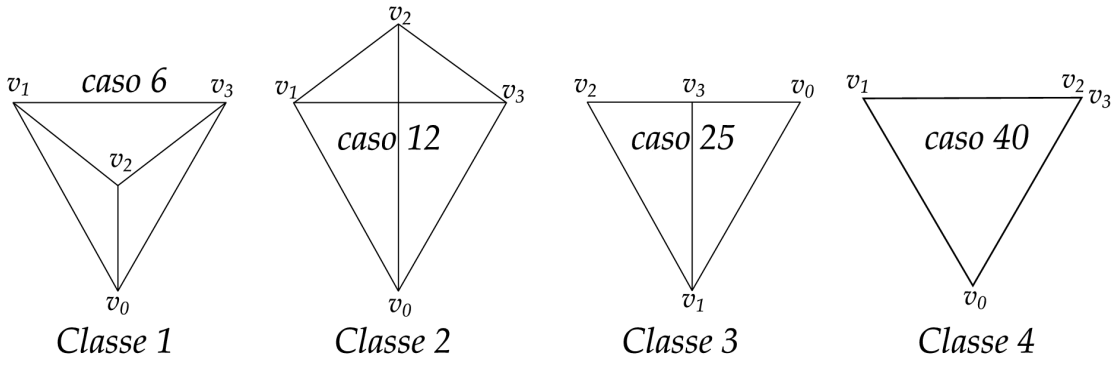
Algoritmo proposto – 1° passo

Esquema de consulta às texturas



Algoritmo proposto – 1° passo

Nova classificação dos tetraedros



$$vec_1 = v_1 - v_0$$
$$vec_2 = v_2 - v_0$$
$$vec_3 = v_3 - v_0$$
$$vec_4 = v_1 - v_2$$
$$vec_5 = v_1 - v_3$$

$$teste_1 = sign((vec_1 \times vec_2).z) + 1$$
$$teste_2 = sign((vec_1 \times vec_3).z) + 1$$
$$teste_3 = sign((vec_2 \times vec_3).z) + 1$$
$$teste_4 = sign((vec_4 \times vec_5).z) + 1$$

Exemplos de um caso de cada classe

Algoritmo proposto – 1° passo

Diferença entre as tabelas de classificação

Caso	$teste1$	$teste2$	$teste3$	$teste4$
1	1	1	X	1
2	1	1	X	0
3	1	0	0	0
4	1	0	0	1
5	0	1	0	1
6	0	1	0	0
7	0	0	0	1
8	0	0	0	0
9	1	0	1	0
10	1	0	1	1
11	0	1	1	1
12	0	1	1	0
13	0	0	1	1
14	0	0	1	0

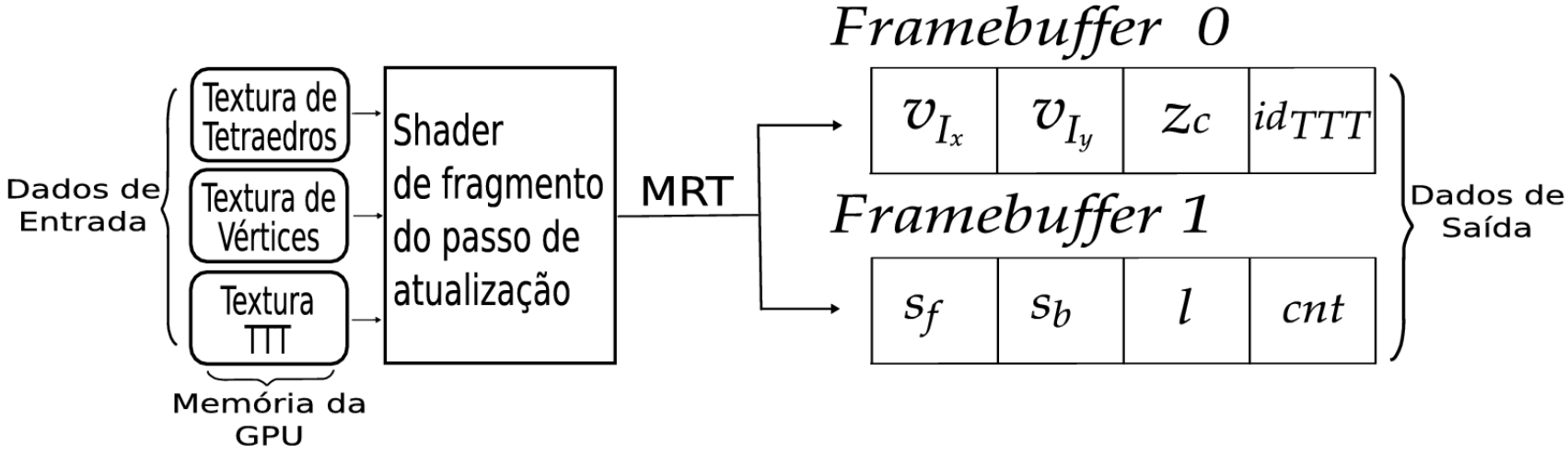
[Wylie et al. 2002]

id_{ttt}	$teste_{1234}$	$Caso$	$RGBA$
0	0 0 0 0	12	0-3-2-1
1	0 0 0 1	18	0-3-2-1
2	0 0 0 2	6	0-3-2-1
3	0 0 1 0	25	1-0-3-2
4	0 0 1 1	40	2-3-1-0
.	.	.	.
.	.	.	.
.	.	.	.
80	2 2 2 2	11	0-1-2-3

Tabela do algoritmo proposto

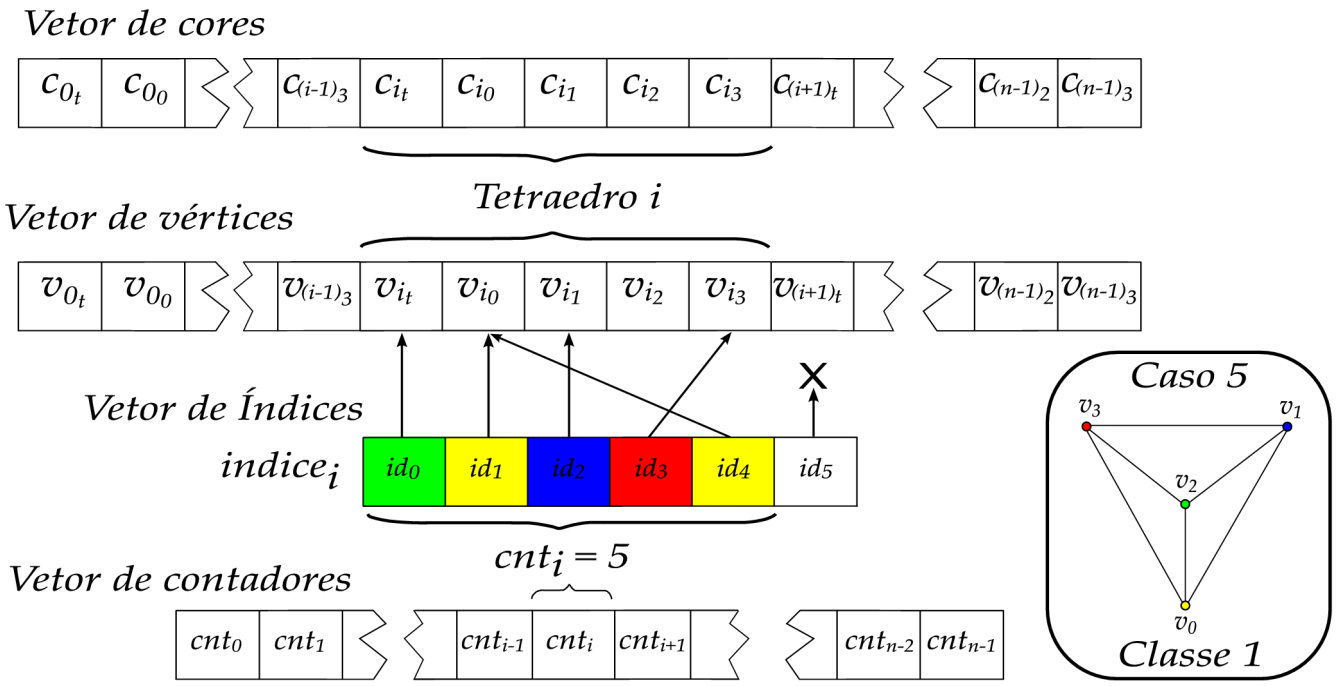
Algoritmo proposto – 1° passo

Esquema de entrada/saída do primeiro passo



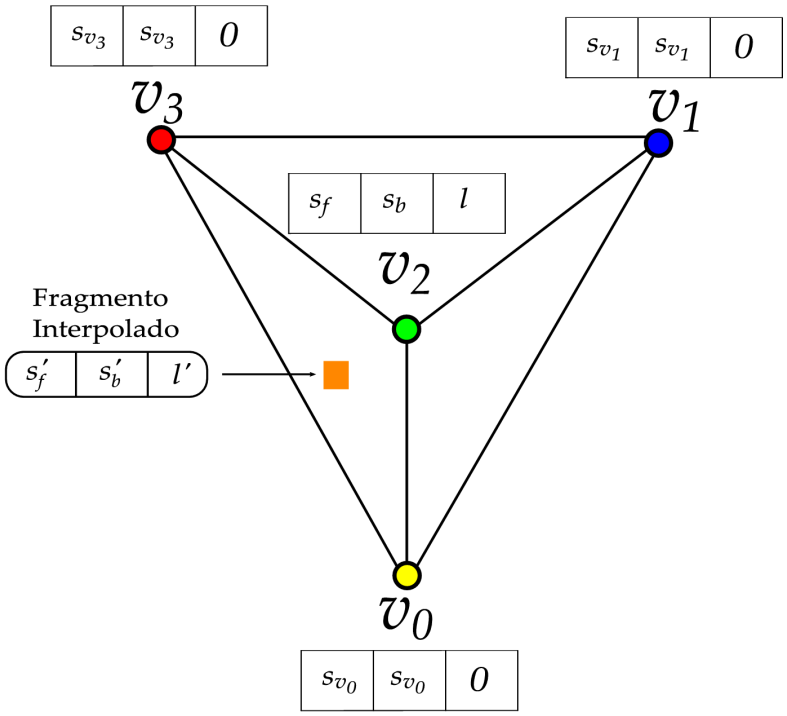
Algoritmo proposto – CPU

Ordenação das células e organização dos vetores



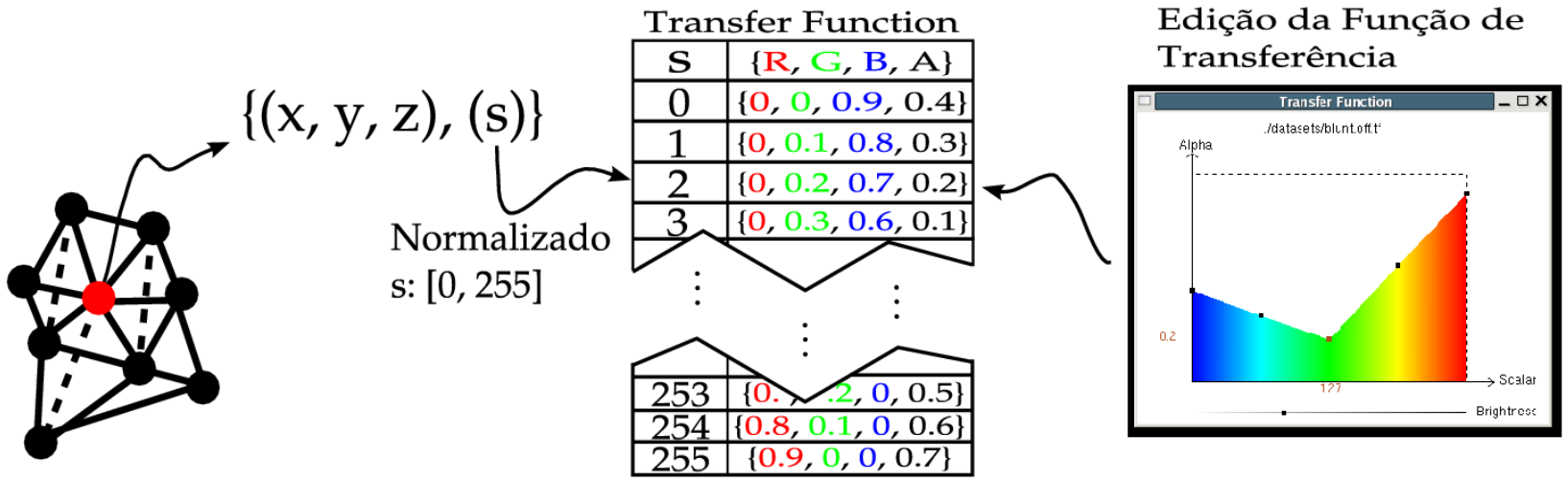
Algoritmo proposto – 2° passo

Interpolação dos fragmentos $\{s_f, s_b, l\}$



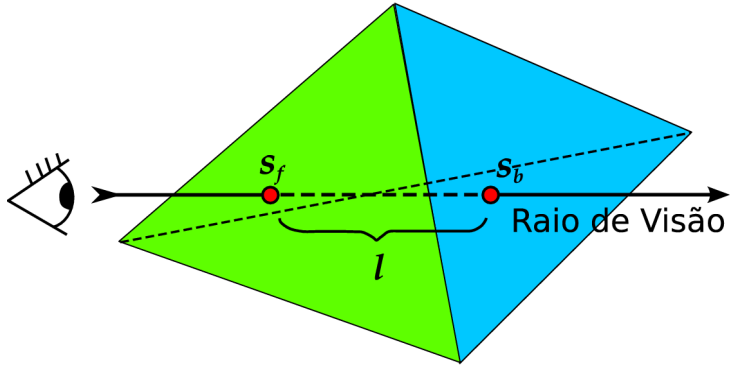
Algoritmo proposto – 2° passo

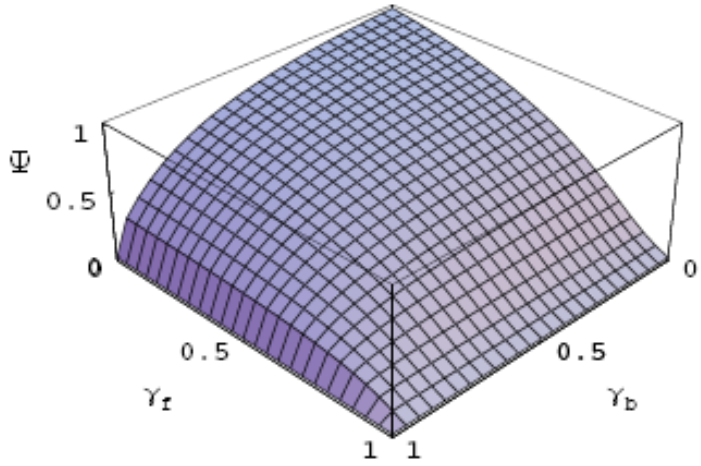
Consulta à textura de função de transferência



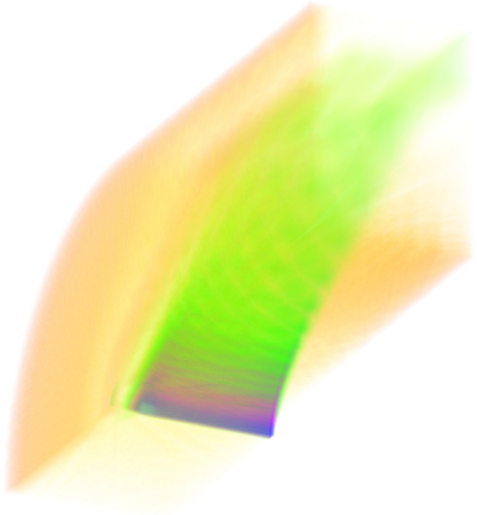
Algoritmo proposto – 2° passo

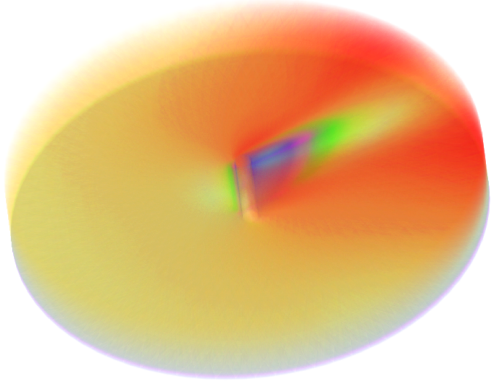
Consulta à textura com a tabela ψ de pré-integração parcial

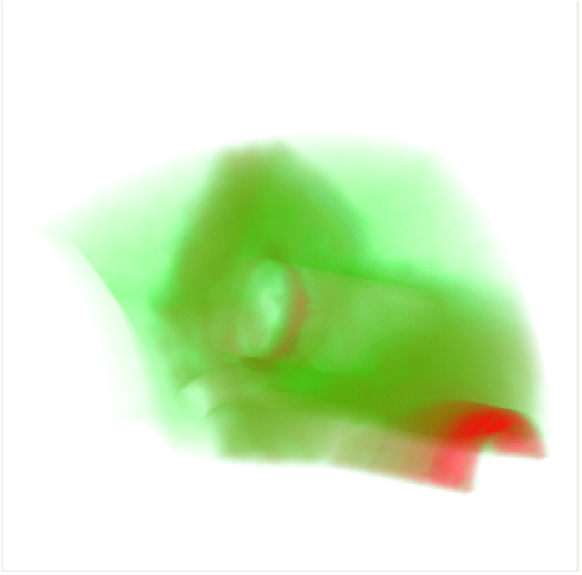


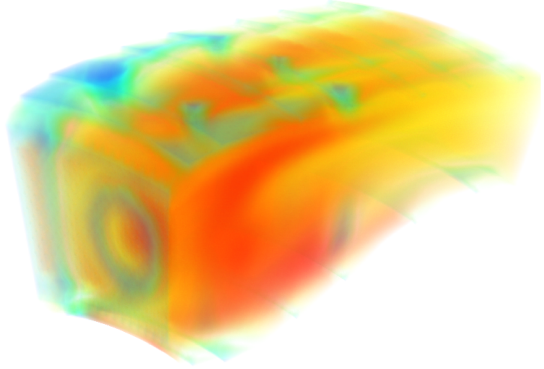


Resultados

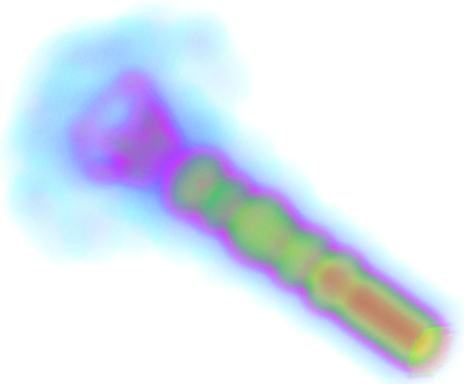


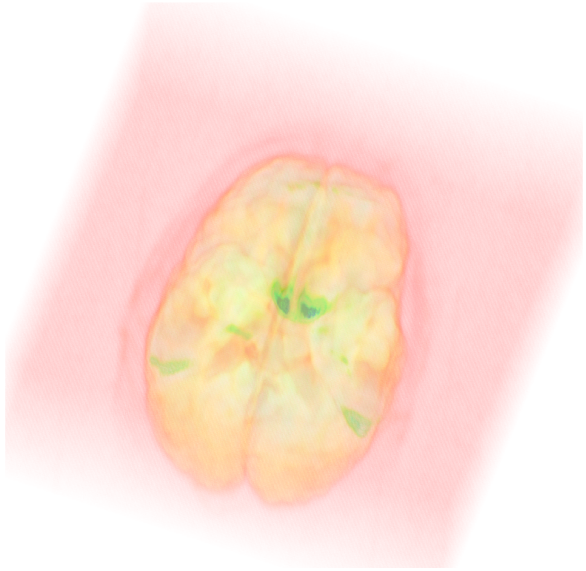






Resultados

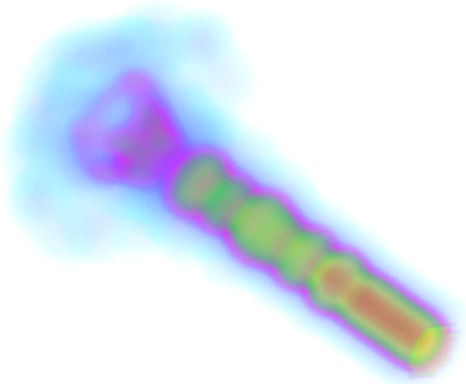


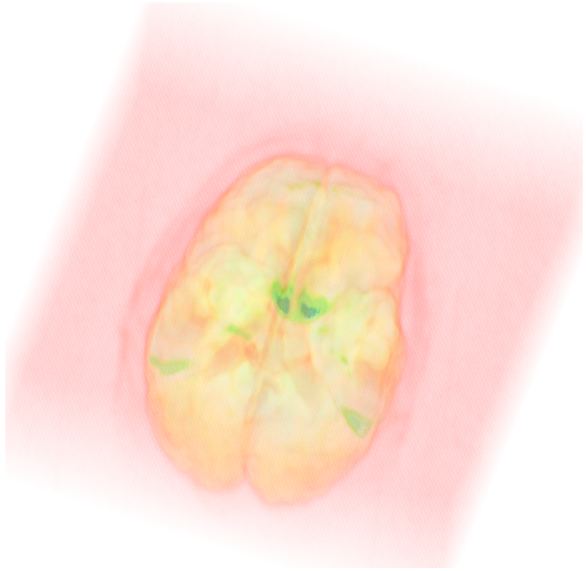


Algoritmo Proposto

Dado	Vértices	Tetraedros	fps	M tets/s
Blunt Fin	40 K	187 K	11,30	2,1197
Oxygen Post	110 K	513 K	4,49	2,3844
SPX	150 K	828 K	3,04	2,5269
Combustion Chamber	47 K	215 K	9,32	2,0054
Fuel Injection	262 K	1,5 M	1,49	2,2460
Brain Tomography	950 K	5,5 M	0,46	2,5608

Resultados





Algoritmo Proposto

Dado	Vértices	Tetraedros	fps	M tets/s
Blunt Fin	40 K	187 K	11,30	2,1197
Oxygen Post	110 K	513 K	4,49	2,3844
SPX	150 K	828 K	3,04	2,5269
Combustion Chamber	47 K	215 K	9,32	2,0054
Fuel Injection	262 K	1,5 M	1,49	2,2460
Brain Tomography	950 K	5,5 M	0,46	2,5608

Resultados

Comparação com outros algoritmos

Algoritmo / Dado	Blunt Fin	Oxygen Post
PTINT	11,30 fps	4,49 fps
GATOR	4,07 fps	1,51 fps
VICP (GPU)	5,20 fps	1,93 fps
VICP (CPU)	1,82 fps	0,57 fps
VICP (Balanced)	4,10 fps	0,11 fps
HARC	4,47 fps	8,63 fps
HARC (INT)	4,94 fps	5,93 fps
HAVIS	2,36 fps	0,79 fps

Resultados

PTINT x GATOR

Algoritmo / Dado	Blunt Fin	Oxygen Post
PTINT	11,30 fps	4,49 fps
GATOR	4,07 fps	1,51 fps
VICP (GPU)	5,20 fps	1,93 fps
VICP (CPU)	1,82 fps	0,57 fps
VICP (Balanced)	4,10 fps	0,11 fps
HARC	4,47 fps	8,63 fps
HARC (INT)	4,94 fps	5,93 fps
HAVIS	2,36 fps	0,79 fps

Resultados

PTINT x VICP

Algoritmo / Dado	Blunt Fin	Oxygen Post
PTINT	11,30 fps	4,49 fps
GATOR	4,07 fps	1,51 fps
VICP (GPU)	5,20 fps	1,93 fps
VICP (CPU)	1,82 fps	0,57 fps
VICP (Balanced)	4,10 fps	0,11 fps
HARC	4,47 fps	8,63 fps
HARC (INT)	4,94 fps	5,93 fps
HAVIS	2,36 fps	0,79 fps

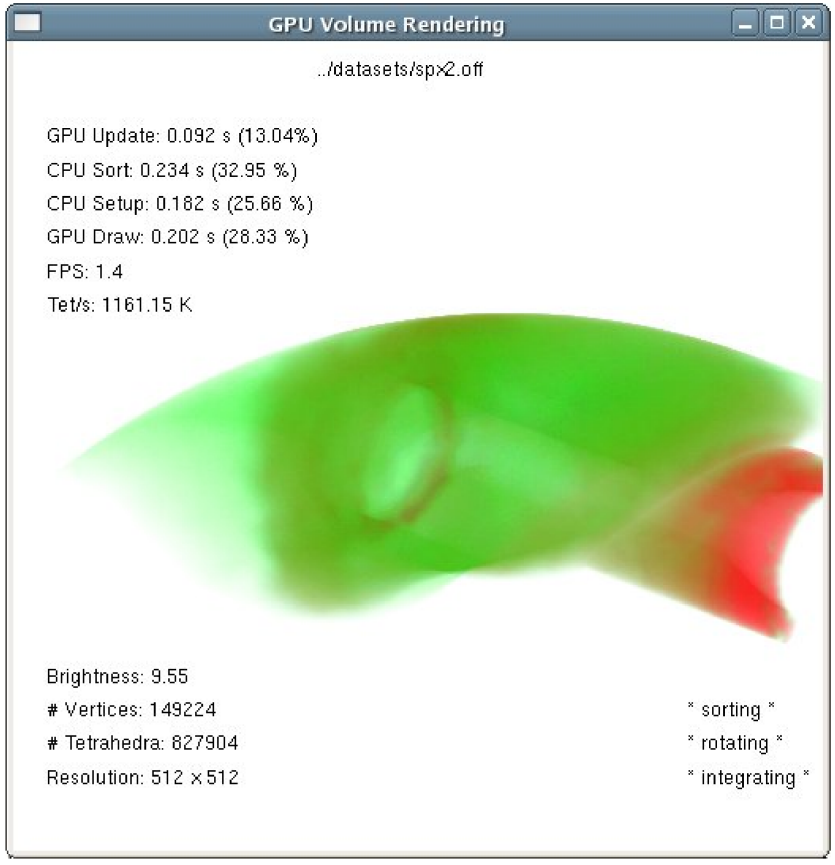
Resultados

PTINT x HARC

Algoritmo / Dado	Blunt Fin	Oxygen Post
PTINT	11,30 fps	4,49 fps
GATOR	4,07 fps	1,51 fps
VICP (GPU)	5,20 fps	1,93 fps
VICP (CPU)	1,82 fps	0,57 fps
VICP (Balanced)	4,10 fps	0,11 fps
HARC	4,47 fps	8,63 fps
HARC (INT)	4,94 fps	5,93 fps
HAVIS	2,36 fps	0,79 fps

Resultados

Janela de interface do algoritmo proposto

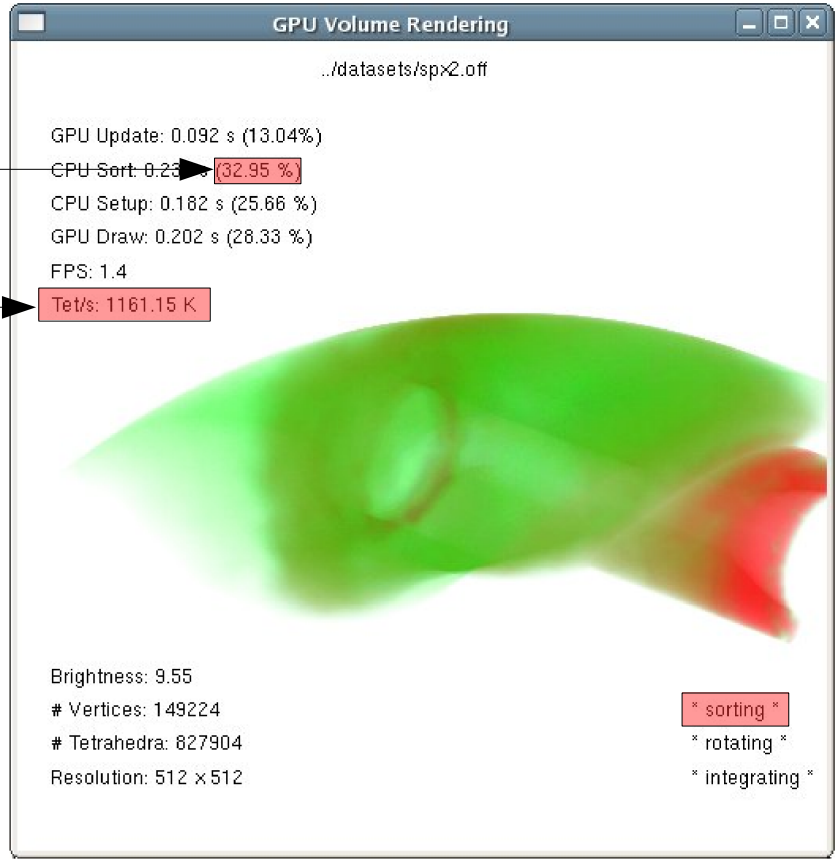


Resultados

Merge sort todos os quadros

32,95%

1.161,15 K Tet/s

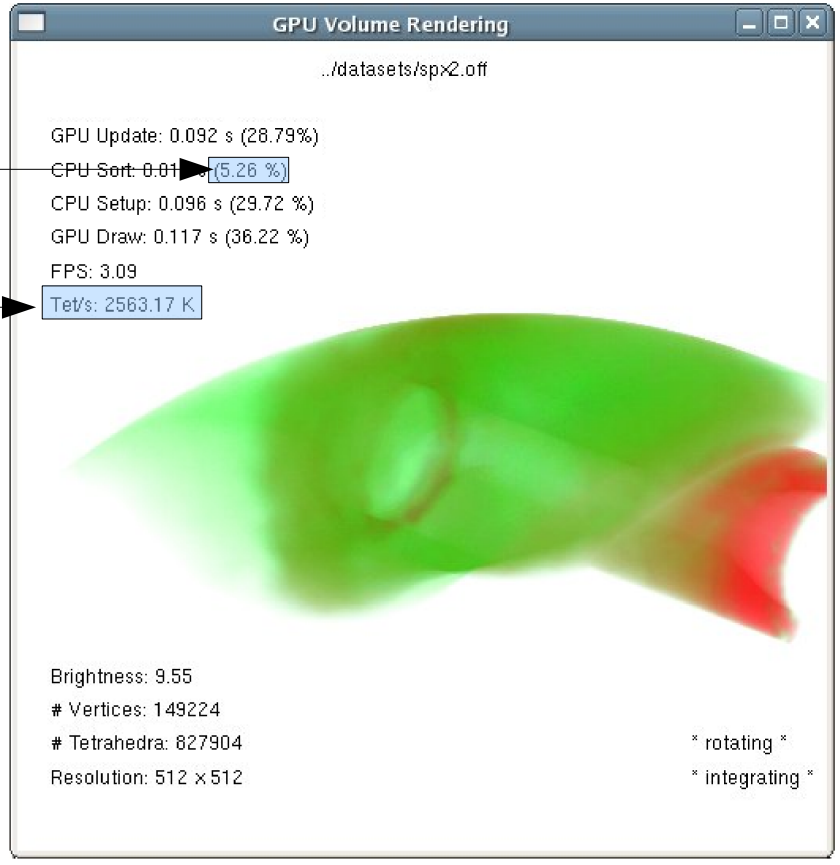


Resultados

Usando bucket sort

5,26%

2.563,17 K Tet/s

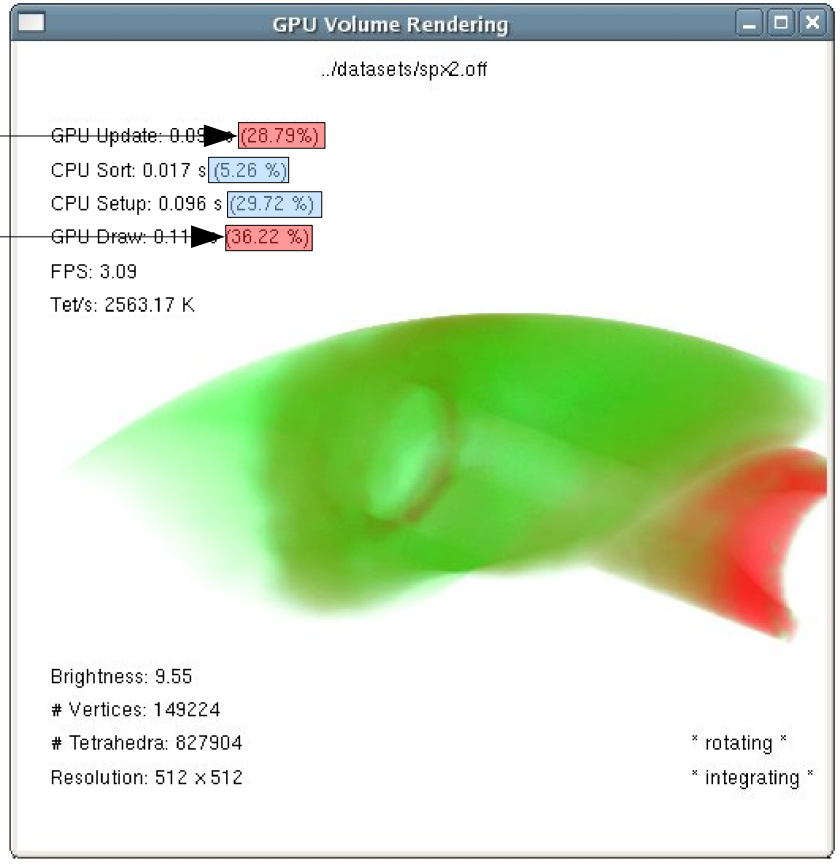


Resultados

Custo dos 2 passos em GPU

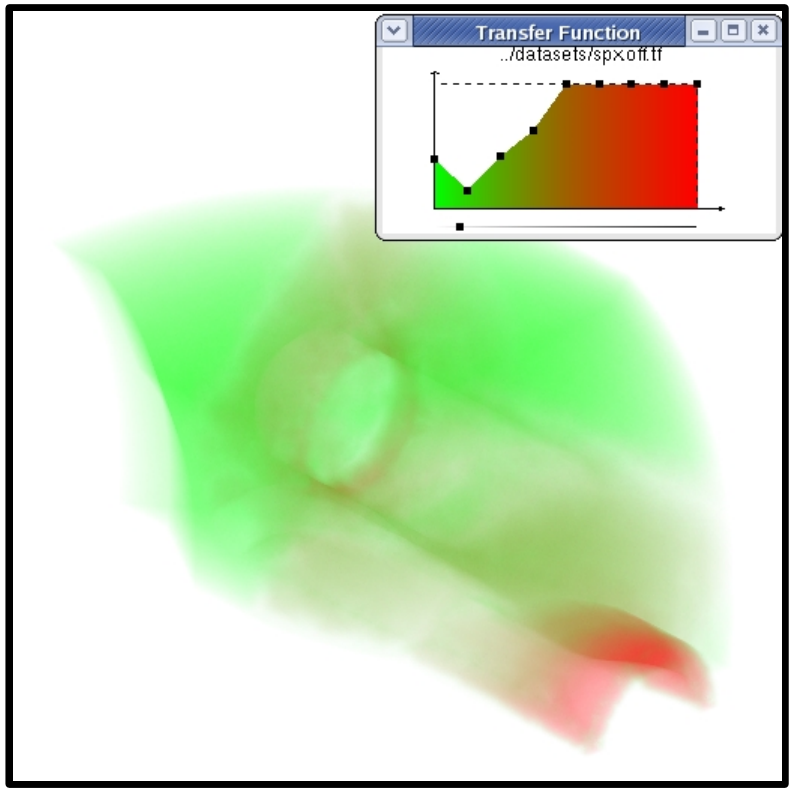
28,79%

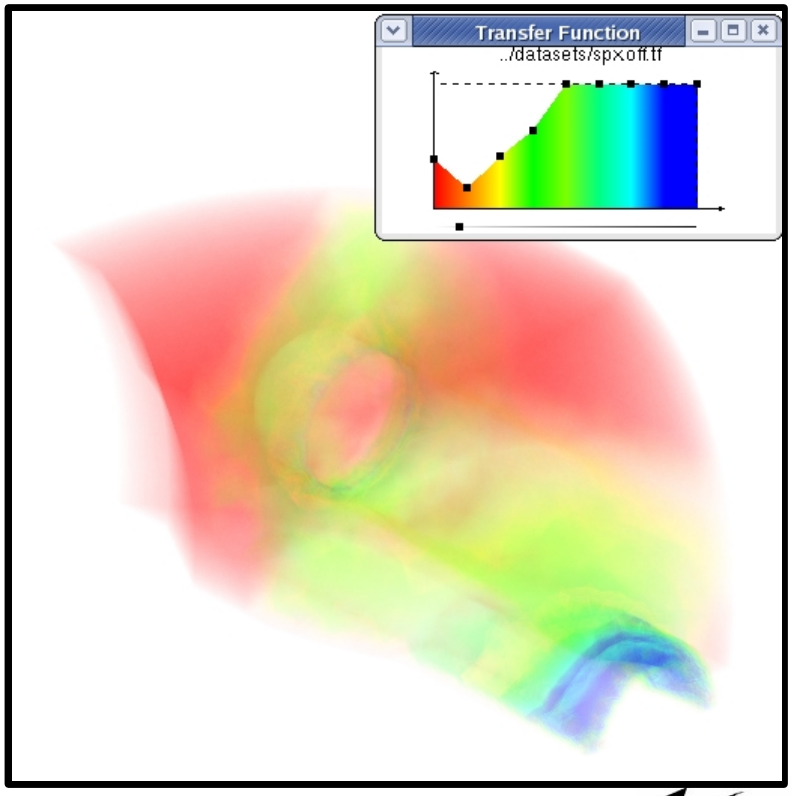
36,22%



Resultados

- Edição Interativa da Função de Tranferência
- Opção de diferentes Mapas de Cor





Conclusões

- Mais de 2 M tets/s
- Imagens de melhor qualidade
- Edição interativa da Função de Transferência
- Não sobrecarrega o barramento CPU – GPU
- Nenhuma estrutura de dados auxiliar na memória da GPU
- Ordenação por centróide causa artefatos

Publicações

- Artigo: *"GPU-Based Cell Projection for Interactive Volume Rendering"* aceito no [SIBGRAPI 2006] págs. 147-154
- Escolhido entre os 6 melhores do [SIBGRAPI 2006] e indicado para o [CGF Journal]
- Artigo: *"High-Performance Volume Rendering for 3D Heart Visualization"* aceito no [HPC Life 2006] workshop do [SBAC-PAD 2006]

Trabalhos Futuros

- Novas funcionalidades na GPU
- Algoritmo de ordenação em GPU (HAVS)
- Gradiente do campo escalar
- Visualização híbrida: volume + iso-superfície
- Modelo de iluminação mais complexo

Projeção de Células baseada em GPU para Visualização Interativa de Volumes

Obrigado!

andmax@cos.ufrj.br

http://www.lcg.ufrj.br/Projetos/volume_render