Rudi Lopes Bravo de Andrade

Um estudo sobre técnicas de renderização volumétrica com implementação e otimização do algoritmo Ray Tracing

Florianópolis

Junho de 2007

Rudi Lopes Bravo de Andrade

Um estudo sobre técnicas de renderização volumétrica com implementação e otimização do algoritmo Ray Tracing

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharelado em Ciências da Computação na Universidade Federal de Santa Catarina

Orientador:
Prof. Dr. rer.nat. Aldo von Wangenheim

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis

Junho de 2007

Resumo

Esse trabalho se trata de um levantamento de diversas técnicas de renderização volumétricas, com ênfase em volumes construídos a partir de imagens de Raio-X obtidas em scanners de tomografia. As técnicas podem ser divididas em duas categorias, as técnicas de aproximação do volume para uma malha de polígonos e as de renderização direta do volume, com a idéia de renderizar pontos na tela para obter uma imagem completa. Dentro dessa segunda categoria, existe a técnica de Ray Tracing, onde a partir do ponto de observação da cena, são lançados raios, ou segmentos de reta, através do volume. Esses raios correspondem à um pixel na imagem final e a cor desse pixel vai variar de acordo com a cor e opacidade dos pontos pelos quais o raio vai colidir. Por ter essa característica de percorrer o volume guardando a cor e opacidade dos pontos, existe a possibilidade de fazer determinadas superfícies transparentes ou opacas, diferente de técnicas de aproximação por polígonos em que toda malha será sempre opaca. A técnica Ray Tracing será implementada e testada, constando durante o trabalho as etapas em que ela foi separada e também novas contribuições para melhorar a performance em relação à como ela foi proposta. Para melhorar ainda mais a performance, no final da implementação foi utilizada a estrutura de dados octree para substituir a representação direta do volume.

Abstract

This work consists in a study concerning volume data rendering. Volume data is a structure that stores three-dimensional data on and it is used in a widely set of applications. One kind of volume data are x-ray sliced images acquired from computer tomography, which will be the used as input on this work. There are two different ways to display volumetric information. The first one is to approximate the volume to a mesh of polygons. However this is performed through approximations and may lead to representation errors. Another way to display these data is to render every small part of the total volume, using point as primitive, and those techniques are called direct volume rendering. Ray tracing is a direct volume rendering, and it consists in casting rays from the observer point of view to the volume, checking the points it cross on the volume, and rendering each ray as a pixel on the screen. The Ray Tracing technique takes advantage on this kind of volume pipelining possibilities and it can blend the volume in a way to create transparency on some selected parts. After the implementation of the ray tracing algorithm, it will be changed to take advantages of the volume data redundancy, using a special data structure named Octree.

List of Algorithms

1 Calculando Variação Vertical e Horizontal. . . . . p. 21
2 Aplicando variação nos raios. . . . . p. 21
3 Etapa de composição . . . . . p. 25
4 Etapa de classificação . . . . . p. 25
5 Etapa de shading . . . . . p. 27

Sumário

Lista de Figuras

1 Introdução p. 9

1.1 Motivação e escopo . . . . . p. 10

1.2 Objetivos e metodologia . . . . . p. 10

2 Revisão Bibliográfica p. 12

2.1 Renderização de Volumes . . . . . p. 12

2.2 Aproximação com Cuberille . . . . . p. 12

2.2.1 Marching cubes . . . . . p. 13

2.2.2 Ray Tracing . . . . . p. 15

2.2.3 Algoritimo Shear Warp . . . . . p. 15

2.2.4 Texture Mapping . . . . . p. 16

2.3 Métricas de Performance . . . . . p. 16

2.4 Otimização . . . . . p. 17

2.4.1 Hardware . . . . . p. 17

3 Desenvolvimento p. 19

3.1 Justificando a escolha . . . . . p. 19

3.2 Ray Tracing . . . . . p. 20

3.3 Classificação . . . . . p. 24

3.4 Shadding . . . . . p. 25

3.5 Junção da etapa de shading e classificação . . . . . p. 28

3.6 Octree . . . . . p. 29

4 Resultados p. 30

4.1 Octree . . . . . p. 33

5 Conclusão e Trabalhos futuros p. 36

Referências p. 38

Anexo A – Diagramas de atividade p. 40

Lista de Figuras

1 Reflexão da luz nas quinas dos cubos. . . . . p. 13

2 Aproximação das quinas para um superfície lisa. . . . . p. 13

3 Construção do cubo a partir de dois planos quadriculados. . . . . p. 14

4 Tabela de construção dos planos segundo os vértices. . . . . p. 14

5 Fatias de texturas. . . . . p. 16

6 Diagrama geral de atividades . . . . . p. 20

7 Frustrum da cena. . . . . p. 20

8 Dados para cálculo da direção do raio. . . . . p. 21

9 Composição dos voxels de trás pra frente. . . . . p. 22

10 Composição da imagem com 80 amostras por raio. . . . . p. 23

11 Composição da imagem com 256 amostras por raio. . . . . p. 23

12 O pior caso para o comprimento de um raio. . . . . p. 24

13 Composição da imagem com número variável de amostras. . . . . p. 24

14 Crânio renderizado com shading plano. . . . . p. 26

15 Crânio renderizado utilizando modelo de Phong. . . . . p. 27

16 Crânio com pele transparente. . . . . p. 28

17 Phantoms simulando artérias. . . . . p. 30

18 Crânio renderizado com Marching Cubes . . . . . p. 31

19 Crânio renderizado com Ray Tracing. . . . . p. 31

20 Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, com opacidade 0.2 . . . . . p. 32

21 Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, com opacidade 0.4 . . . . . p. 32

22 Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, completamente opaca. . . . . p. 33

23 Crânio renderizado utilizando a estrutura de dados Octree. Com espaçamento entre voxels (2,2,12). . . . . p. 34

24 Crânio renderizado utilizando a estrutura de dados Octree. Com espaçamento entre voxels (4,4,24). . . . . p. 35

25 Diagrama de atividades composição . . . . . p. 40

26 Diagrama de atividades classificação. . . . . p. 41

27 Diagrama de atividades shading. . . . . p. 42

28 Diagrama de atividades do cálculo da cor de saída do raio. . . . . p. 43

1 Introdução

Renderização de imagens gráficas, no contexto computacional, é o processo de criação de imagens digitais a partir de modelos tridimensionais. Esses modelos normalmente representam objetos geométricos; por exemplo, cubos, cones ou esferas; são objetos que não podem ser representados no modelo bidimensional convencional, utilizado pelos atuais displays gráficos, como monitores e televisores. Renderizar imagens gráficas é transformar esses objetos 3D, junto com informações de luz, de texturas e de materiais, em imagens bidimensionais que possam ser exibidas sem perder informações fundamentais sobre o modelo original.

Renderização é a etapa final de processamento nos programas gráficos, é o momento em que toda informação ganha uma representação visual que o ser humano é capaz de interpretar. As características deste processo, vão atribuir diferentes nuances à imagem gerada, e assim é possível que existam interpretações diferentes do mesmo conjunto de dados representado.

Essa variação ocorre porque cada técnica possui uma abrangência limitada de representações de aspectos do mundo real, seja porque alguns fenômenos físicos são muito caros, em termos de processamento, para serem representados, ou porque o modelo físico no mundo real ainda não possui nenhuma representação computacional boa o suficiente para representá-lo. Vale ressaltar que segundo [Foley et al.(1990)Foley, van Dam, Feiner, and Hughes] as técnicas podem ser mais ou menos fotorealistas e um dos fatores determinantes é o tratamento que ela dá para iluminação.

A representação geométrica dos objetos é a forma mais usada para compor o modelo. Um cubo por exemplo, poderia ser descrito como oito vértices interligados por arestas, estas formando seis planos. Outra maneira de descrever o modelo é o uso de informações volumétricas. Análogo aos pixels, o voxel é a representação de um pequeno volume no espaço tridimensional, contendo informações de cor e opacidade.

Uma abordagem para renderização volumétrica é a aproximação do volume para uma representação geométrica compatível, como a proposta por [Lorensen and Cline(1987)], que pode ser utilizada em técnicas convencionais de renderização. Outra abordagem é a renderização

do volume de forma direta, esta será explorada no decorrer do trabalho, com a técnica de renderização Ray Tracing de [Levoy(1988)].

Ray Tracing, ou "Seguir os Raios" em tradução literal, consiste na técnica de calcular o comportamento dos feixes de luz na cena 3D, a partir da propriedade óptica dos objetos da cena. A idéia básica por traz do Ray Tracing é que os raios de luz, depois de sairem da fonte de luz, possivelmente encontrarão objetos da cena pelo seu caminho, e esses objetos possuem propriedades físicas que irão fazer o raio mudar seu estado, como reflexos, refrações ou ofuscação, e em algum momento, finalmente, irão incindir na representação computacional da visão humana, a câmera virtual.

O maior problema da utilização de Ray Tracing para a renderização de volumes, é o custo computacional que a técnica necessita para o processamento de imagens. Existem muitas pesquisas indo na direção de melhorias dos algoritmos, melhorias de hardware e aproveitamente da infraestrutura atual, em termos de processamento, para fazer do Ray Tracing uma técnica capaz de processar imagens em tempo real, tornando a técnica hábil ao uso em programas interativos.

1.1 Motivação e escopo

O trabalho será desenvolvido como parte integrante de um software da área médica, que tem como propósito reconstruir imagens geradas por scanners de tomografia. O software serve de auxílio a médicos, que podem visualizar o exame com informações extras graças à representação tridimensional dos dados.

Esse programa já possui um método de renderização das imagens em tempo real, mas pouco realista. As técnicas que existem hoje podem servir de apoio ao processo, incrementando a qualidade do software.

1.2 Objetivos e metodologia

É o objetivo do trabalho implementar um método de renderização de informações volumétricas, com algumas características realistas de iluminação. Otimizá-lo de alguma forma e avaliar a sua viabilidade para uso em aplicações de tempo real.

Nesse processo será necessário estudar técnicas existentes e justificar a escolha de uma. Para a elaboração da implementação serão necessários diagramas de atividades, pelos quais o desenvolvimento será guiado. A implementação do modelo, utilizará dados de uma reconstrução tridimensional de uma série de dados médicos, obtidos através de scaners de tomografia. Uma

vez que o processo de renderização esteja implementado e validado, a próxima etapa será estudar maneiras de acelerar o processo, através de otimização de código ou paralelismo de instruções.

2 *Revisão Bibliográfica*

2.1 Renderização de Volumes

Como relatado por [Elvins(1992)] os volumes podem ser renderizados utilizando duas abordagens diferentes. Uma é a renderização com aproximação de malhas, como os algoritmos propostos por [Che et al.(1985)Che, Herman, Reynolds, and Udupa] e [Lorensen and Cline(1987)], onde o volume em questão recebe um tratamento para detecção de formas geométricas e é renderizado através de algoritmos convencionais. A outra é a renderização direta do volume, como nos trabalhos de [Levoy(1988)] e [Lacroute and Levoy(1994)], onde os próprios voxels representam o volume, numa abordagem de renderização direta, usando pontos como primitiva. Dentro dessa última categoria também existe algumas técnicas particulares, como o trabalho de [Wilson et al.(1994)Wilson, VanGelder, and Wilhelms] que utiliza texturas para representar o volume, sendo este mais recente e otimizado para as arquiteturas atuais de vídeo.

2.2 Aproximação com Cuberille

Método proposto por [Che et al.(1985)Che, Herman, Reynolds, and Udupa], apresenta o *cuberille*, que é um conceito diferenciado de *voxel*. Ele utiliza um cubo para representar a menor unidade de volume, análogo a representar um pixel como um quadrado, para a menor unidade de espaço bidimensional.

A técnica consiste em renderizar diretamente os cubos do volume, utilizando a sua forma geométrica. O resultado não seria representativo se não houvesse algum tratamento antes da renderização. A imagem gerada teria diversas quinas, resultantes da composição dos cubos, isso porque as faces do cubo irão refletir a luz em direções distintas, como mostrado na imagem 1.

A proposta da técnica é detectar as superfícies de cubos, que podem descrever a superfície física do modelo, e aproximar a reflexão da luz, a partir de técnicas de *shading*, para refletir o

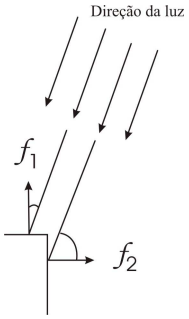


Figura 1: Reflexão da luz nas quinas dos cubos.

mais similar ao modelo físico. Essa aproximação pode ser observada na figura 2.

Figura 2: Aproximação das quinas para um superfície lisa.

2.2.1 Marching cubes

Marching Cubes é um algoritmo de reconstrução de malha a partir de um volume proposto por [Lorensen and Cline(1987)]. Reconstruir a malha provê a facilidade do modelo poder ser visualizado em procedimentos convencionais de renderização.

No Marching Cubes, cada uma das imagens que compõem o volume é dividia em espaços quadriculares regulares, formando uma grade. Duas imagens adjancentes, possuirão pontos adjacentes, que se observados em grupos de oito, formarão cubos. A figura 3 fornece um entendimento visual dos cubos. Cada vértice do cubo representa um pedaço da imagem original e possui um valor de cor. De acordo com o orgão do dados que se queira visualizar, é escolhida uma cor, que servirá como padrão. Se o vértice possui um valor inferior ou igual ao padrão, ele é considerado como fazendo parte do volume, caso contrário ele está do lado de fora do volume.

Os cubos então terão informações de vértices pertencentes ou não ao volume. Um estudo realizado pelos autores do algoritmo propõe que existem 14 disposições possíveis de vértices, e que a triangulação dos vértices pode ser substituida por uma referência a algum dos casos da figura 4.

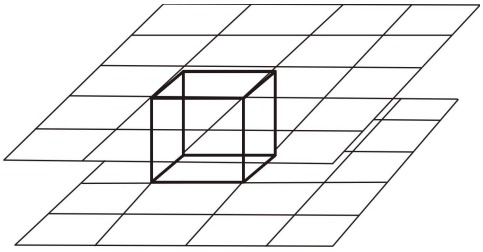


Figura 3: Construção do cubo a partir de dois planos quadriculados.

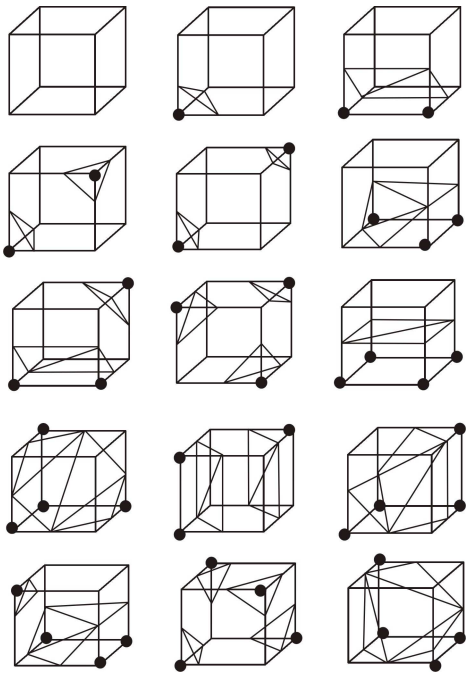


Figura 4: Tabela de construção dos planos segundo os vértices.

2.2.2 Ray Tracing

Uma abordagem de [Levoy(1988)] para realização de renderização volumétrica que não utiliza a aproximação geométrica do volume, mas renderiza o volume diretamente.

A partir de um conjunto de dados de informações volumétricas previamente obtido, como em um scanner de tomografia por exemplo, calcula-se a cor e opacidade para todos os *voxels* do volume, para depois renderizar a imagem final através da técnica *Ray Tracing*, ou *Ray Casting* para alguns autores.

As etapas de *shading* e classificação definem a cor e a opacidade, respectivamente, dos *voxels*. Na etapa de composição, raios são lançados a partir do ponto de vista do observador em direção ao volume, cada raio que é lançado representa um pixel da imagem final.

Os raios não irão necessariamente colidir com os voxels, isso porque os voxels são pontos discretos no espaço. Para contornar esse problema são coletadas amostras, em número significativo, de cor e opacidade, obtidas através da interpolação trilinear dos oito voxels mais próximos de cada amostra.

A opacidade e cor das amostras são mescladas de trás pra frente, da amostra mais distante do ponto de vista até a mais próxima, definindo a cor do pixel na imagem final.

2.2.3 Algoritimo Shear Warp

É um algoríitmo de [Lacroute and Levoy(1994)] mais recente e rápido, que não descarta a qualidade da imagem gerada em função do tempo. Para entender a idéia proposta, usa-se a analogia a um livro: Olhando um livro de cima, temos apenas a visualização da capa. Se quiseremos olhar a lateral do livro, sem rotacioná-lo, podemos incliná-lo, ”escorregando” as páginas pro lado.

Depois de conseguir obter a visualização desejada, a imagem é composta de trás pra frente, utilizando o operador *over* ([Porter and Duff(1984)]). Esse passo projeta o volume em uma imagem 2D que no passo seguinte sofre uma transformação para ganhar perspectiva.

Essa técnica já foi empregadada anteriormente por [Schröder and Stoll(1992)] e [Yagel and Kaufman(1992)]. O trabalho de [Lacroute and Levoy(1994)] aproveitou três características que essa técnica propicia, para otimizá-la, sendo a mais importante delas o fato do volume nunca sofrer rotação.

2.2.4 Texture Mapping

Por aproveitar o suporte que as placas de vídeo atuais dão para as texturas, é um processamento significantemente mais rápido em comparação aos demais.

A técnica proposta por [Wilson et al.(1994)Wilson, VanGelder, and Wilhelms] se baseia na criação de texturas 3D a partir das fatias originais do volume. Lembrando que as fatias são as imagens do scanner de tomografia, para cada uma delas são calculadas cor e opacidade, igual nos outros métodos de renderização direta de volumes, para então criar uma textura intermediária, chamado no trabalho de texel. Os texels terão que ser regerados a cada mudança nas informações do volume, ou adição de mais imagens, mas não terão que ser regerados caso aconteçam transformações como escala e rotação.

Depois que as texturas 3D são criadas, a renderização é feita aplicando planos paralelos à textura 3D, criando uma pilha de novas texturas que serão renderizadas alinhadas com o usuário, uma analogia que o autor usa são fatias de pão que compõe o pão, do mesmo jeito que as fatias de texturas compõe o volume. A figura 5 mostra uma visualização melhor da criação dessas texturas.

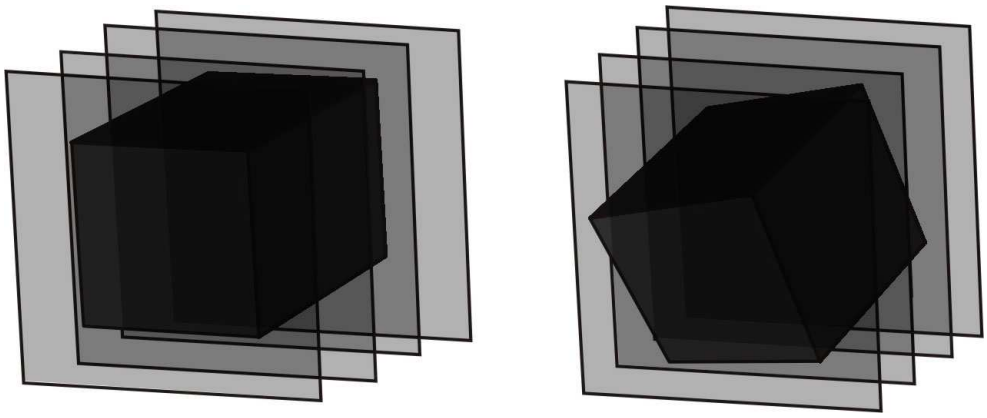


Figura 5: Fatias de texturas.

2.3 Métricas de Performance

As métricas de performance, para justificar a escolha de uma ou outra técnica no decorrer do trabalho, foi proposta por [Ray et al.(1999)Ray, Pfister, Silver, and Cook] e leva em conta os seguintes fatores, que ele considera determinantes.

- Frame Rate: O número de imagens que podem ser geradas por segundo (ou Hz). Estudos feitos por [Akenine-Müller and Haines(1999)] apontam que são necessários 30 frames

por segundo, para o olho humano captar a sensação de movimento. Se ficar muito abaixo disso a sensação que o usuário terá, são de imagens fixas sendo exibidas e não uma animação.

- Amostras Processadas Por Segundo(APPS): O número de amostras que tem seus valores interpolados. Diferente do frame rate o valor de APPS não depende da resolução da imagem. É um ponto crucial do programa já que um número insuficiente de amostras pode conduzir a resultados insatisfatórios na composição final.

- Latência: é o tempo entre alguma mudança nos dados do modelo, e a imagem gerada com essa mudança. É muito necessário em dados que estão sofrendo alterações, principalmente alterações em tempo real, como o trabalho no campo de Ultrasonografia 3D de [Ohbuchi et al.(1992)Ohbuchi, Chen, and Fuchs], mas não é um fator determinante em aplicações com dados imutáveis.

- Qualidade da imagem: está relacionado com a resolução da imagem gerada, o filtro de interpolação usado e os modelos de iluminação aplicados. Dependendo do modelo de dados que serão exibidos, espera-se diferentes características de representação, que aqui estamos tratando como qualidade. A técnica escolhida deve além de ser capaz de gerar imagens em resolução adequada, possuir um filtro de interpolação que não crie ruídos e ser capaz de funcionar com diversos modelos de iluminação.

- Escalabilidade: a possibilidade de expandir a performance do método extendendo a quantidade de processamento e vazão de memória, colocando mais nós de processamento.

2.4 Otimização

Partindo do ponto em que foi decidido que a técnica a ser implementada seria o Ray Tracing, que está justificado no capítulo 3, de acordo com os critérios enumerados na seção 2.3, indo de acordo com as propostas do desenvolvimento e do objetivo. Algumas técnicas de otimização de desempenho foram estudadas, que são basicamente técnicas de aceleração por hardware, software, múltiplos processadores e computadores.

2.4.1 Hardware

As iniciativas de desenvolvimento de hardware partiram de universidades [Ray et al.(1999)], para suprir a necessidade de aplicar a técnica Ray Tracing em aplicações interativas. Os hardwares propostos incorporam o pipeline de renderização para chipsets e placas, que podem executar

as instruções muito mais rápido que quando são comparadas ao hardware de propósito geral, programado a partir de instruções de alto nível.

Um dos maiores problemas para a execução desse tipo de algoritmo é o uso excessivo de memória. A tabela 1 é uma estimativa de qual largura de fluxo de memória seria necessário para garantir a execução à 30hz.

Tamanho de Dados	Frame Rate (Hz)	Fluxo de Memória
$128^3 \times 16$	30	120 MB/s
$256^3 \times 16$	30	960 MB/s
$512^3 \times 16$	30	7.5 GB/s
$1024^3 \times 16$	30	60 GB/s

Tabela 1: Tabela de largura do fluxo de memória.

3 *Desenvolvimento*

3.1 Justificando a escolha

Dentre todas as técnicas estudadas e apresentadas nos capítulos anteriores, a que se mostrou mais apropriada ao desenvolvimento foi a técnica de renderização direta do volume, utilizando o algoritmo Ray Tracing.

Os fatores que levaram a essa escolha foram:

- A qualidade superior das imagens geradas pelo algoritmo, que é possível graças ao uso de técnicas de iluminação realistas, que reproduzem propriedades físicas dos objetos iluminados. O processo também provê a possibilidade de usar diversas opacidades diferente a diferente volumes, podendo aplicar transparência e aumentando o grau de entendimento da imagem gerada.

- A simplicidade e extensibilidade da técnica. A extensibilidade para mudar o método de iluminação foi um fator decisivo para a escolha da técnica, já que seriam necessários diversos testes com diferentes shadings até chegar num resultado satisfatório. A simplicidade permite a fácil detecção dos gargalos de processamento e análise do código atrás de otimizações.

- Possibilidade de paralelizar as informações. Uma grande fatia de otimizações pesquisadas, como a de [Parker et al.(1999)Parker, Parker, Livnat, Sloan, and Shirley], se baseiam na paralelização de informações, seja em clusters de computadores ou utilizando a GPU para cálculos específicos. No ray tracing o cálculo de cada raio é completamente independente de todos os outros raios e levando em conta as taxas de transmissões possíveis em diferentes arquiteturas, pode ser um campo explorado para otimização.

3.2 Ray Tracing

Como já foi dito anteriormente, ray tracing significa seguir a luz. São lançados no modelo, para cada pixel a ser renderizado, um raio reto que parte da câmera (ponto observador) em direção ao modelo. Essa etapa de lançar os raios no modelo é chamada de composição e é nesse momento que as cores dos raios serão computadas. Antes disso, os dados modelo são preparados e a partir do volume inicial são calculados dois novos volumes, referentes ao valor de opacidade e cor dos voxels. As etapas referentes ao cálculo de cor e opacidade são descritas nas sessões seguintes. Por hora admite-se que existam dois processos distintos que retornam essas informações. De um modo geral o pipeline do ray tracing pode ser sintetizado no diagrama de sequência da figura 6.

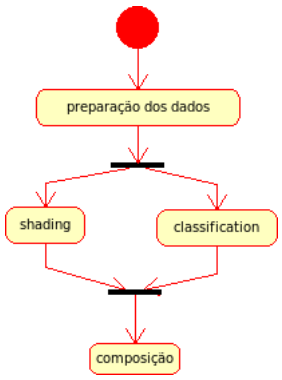


Figura 6: Diagrama geral de atividades

Para calcular a direção dos raios são utilizadas algumas informações da cena. A cena renderizada tem que estar dentro do *frustrum*, campo de visão do observador. Esse é composto por dois planos paralelos, near plane e far plane, que funcionam como limitadores do que é visto na imagem final, tudo que estiver fora desse limite é ignorado na renderização. A imagem 7 fornece uma visualização mais clara dos limitadores da cena.

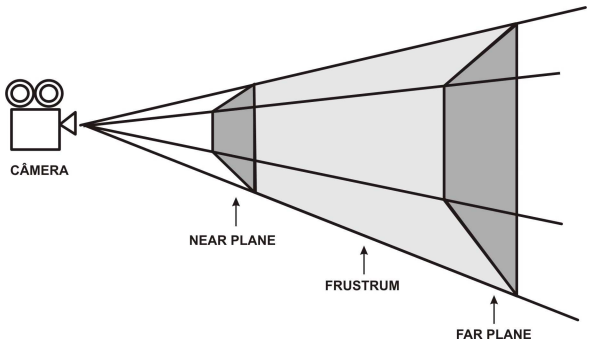
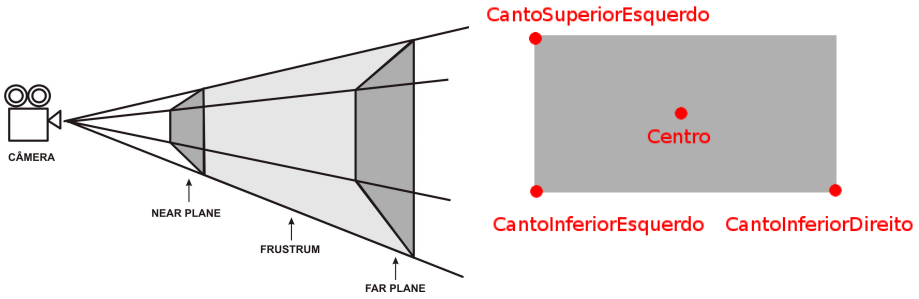


Figura 7: Frustrum da cena.

A câmera tem um vetor que indica a direção que ela está apontando, seu vetor normal, e um vetor que indica qual é o lado que está virado pra cima. Esses vetores estão representados

graficamente na imagem 8. Também é conhecido a distância da câmera até o *near plane* bem como as dimensões deste. Com essas informações é possível obter a posição espacial do centro e cantos do near plane, essa conta está expressa no algoritmo 1. Neste também é obtida a variação que cada raio deve ter no eixo (X,Y) para que o número de raios sejá igual ao tamanho da resolução, e cada raio represente um pixel na imagem final. Após obter a variação em (X,Y) dos raios é só aplicar a variação para cada raio que será jogado no modelo, conforme o algorítmo 2.



(a) Frustrum da cena com informações de câmera.

(b) Informações do Near Plane

Figura 8: Dados para cálculo da direção do raio.

Algoritmo 1 Calculando Variação Vertical e Horizontal.

```
Entrada(Posicao_camera, Camera_up, Camera_normal, Distancia_camera_nearplane)
  side = Camera_up ProdutoVetorial Camera_normal;
  Centro = Posicao_camera + (Camera_normal * Distancia_camera_nearplane);
  CantoSuperiorEsquerdo = Centro + (Camera_up * AlturaNearPlane/2) +
                                   (side * lefLarguraNearPlane/2t);
  CantoInferiorEsquerdo = Centro + (Camera_up * AlturaNearPlane/2) +
                                   (side * LarguraNearPlane/2);
  CantoInferiorDireito = Centro + (Camera_up * AlturaNearPlane/2) +
                                   (side * LarguraNearPlane/2);
  VariacaoVertical = (CantoSuperiorEsquerdo - CantoInferiorEsquerdo) /
                                   HEIGHT_RESOLUCAO;
  VariacaoHorizontal = (CantoInferiorDireito - CantoInferiorEsquerdo) /
                                   WIDTH_RESOLUCAO;
```

Algoritmo 2 Aplicando variação nos raios.

```
Entrada(Raio (u,v))
  PontoNoNearPlane = CantoInferiorEsquerdo + (VariacaoVertical * v) +
                                   (VariacaoHorizontal * u);
  Direção do Raio = (PontoNoNearPlane - Posicao_camera) normalizado;
```

Os raios são jogados na cena e é verificado se houve colisão com o volume. Se houve, são guardadas as posições de entrada e saída do raio, formando um segmento de reta. Esse

segmento de reta representa o caminho que o voxel perccoreu dentro do volume e é a partir dele que as amostras serão coletadas e a cor do voxel composta.

Para entender melhor como essas cores são compostas, assuma que cada voxel possui uma cor e uma opacidade referente a ele. Se olharmos dois voxels, um que esteja na frente e o outro atrás, dependendo da opacidade do voxel da frente, poderemos ver mais ou menos o que está atrás, esse efeito é mostrado pela figura 9. Pensando em vários voxels, a cor dos que estão atrás vai sempre influenciar a cor do que está na frente. Computacionalmente falando, a cor deve ser calculada seguindo uma ordem de trás pra frente e pode ser expressa utilizando a seguinte fórmula:

$$Cor_{frente} = Cor_{tras}(1 - Opacidade_{voxel}) + Cor_{voxel} * Opacidade_{voxel}$$

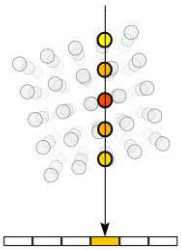


Figura 9: Composição dos voxels de trás pra frente.

A Cor_{frente} da primeira amostra de um raio é a própria cor do raio. Ela é composta por todas as outras cores, mescladas uma a uma de traz pra frente. É trivial a partir da fórmula chegar a conclusão que se um voxel ou amostra tem $Opacidade_{voxel} = 1$, ou seja, ele é completamente opaco, não existirá a necessidade de calcular a cor das amostras atrás dele e isso pode ser explorado como uma terminação prematura do raio, conforme levantado por [Yagel and Kaufman(1992)]. O resultado dessa otimização foi o ganho de 90% no tempo médio de composição do raio em relação ao sem ao mesmo modelo sem término prematuro do raio sem influenciar no resultado visual da imagem gerada. A tabela 2 mostra esse ganho em números absolutos de um pequeno escopo de teste.

	Com término prematuro	Sem término prematuro
Média do tempo de composição	1m11s	2m15s

Tabela 2: Média do tempo de composição em função do término do raio.

Outro fator crítico notado durante a implementação do trabalho foi o número de amostras. Se o número de amostras for muito pequeno o resultado da imagem final será muito pouco satisfatório, como pode ser observado na imagem 10, por outro lado se o número de amostras for

muito grande, a imagem fica satisfatória (11), mas a performance do método fica prejudicada. Notou-se que o número de amostras não podia ser fixo, mas variável e um bom parâmetro encontrado para mensurar o número de amostras que devia ser coletado, foi o comprimento do raio, dentro do volume. Quanto maior o raio mais amostras devem ser coletadas.

Figura 10: Composição da imagem com 80 amostras por raio.

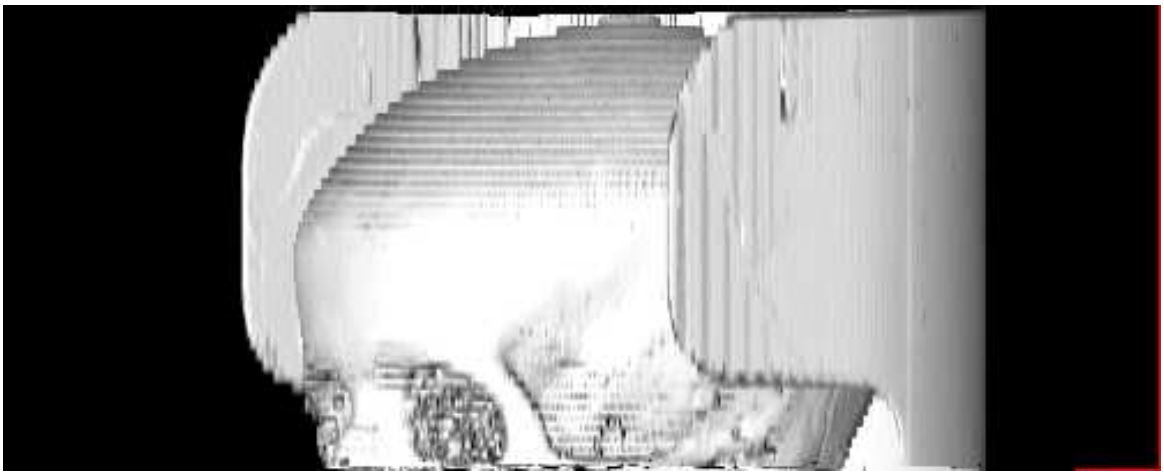


Figura 11: Composição da imagem com 256 amostras por raio.

A proposta sugerida para resolver esse impasse é calcular o número de amostras a partir do tamanho e segmento de reta que interceptou o volume. Como não existe garantia de qual a distância entre os pixels, apenas o tamanho do segmento de reta não é informação o suficiente para decidir quantas amostras serão coletadas para cada raio. A partir de um segmento de reta que corta o volume na diagonal, indo do canto inferior na origem até o canto superior, demonstrado na figura 12, e com o número de voxels que esse segmento colidiu é possível com uma equação linear calcular quantas amostras um raio de qualquer comprimento deve ter comparado à quantidade de voxels que o pior caso teve. Foram realizados alguns testes num escopo reduzido do problema e os resultados de performance podem ser observados na tabela

3, bem como o resultado visual na imagem 13. O uso de amostras variáveis se mostrou muito eficiente e com um resultado gráfico muito superior ao número de amostras fixa de desempenho semelhante.

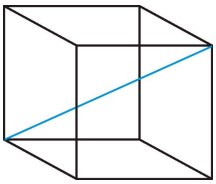


Figura 12: O pior caso para o comprimento de um raio.

	80 amostras	256 amostras	amostras variáveis
Tempo médio de composição	50s	2m27s	1m11s

Tabela 3: Tempo médio de composição em função do número de amostras

Figura 13: Composição da imagem com número variável de amostras.

A etapa de composição pode ser finalmente sintetizada em pseudo-código no algoritmo 3 e o diagrama UML de atividades está anexado na figura 25.

3.3 Classificação

[Levoy(1988)] apresenta um modelo de classificação dos voxels que leva em conta o limite de superfícies, que é mais específico ao tipo de informação adquirida por exames de tomografia é o que se encaixa nesse estudo. Algumas características dos orgãos humanos são levadas em conta e é proposto um método que pode ser descrito da seguinte forma: Uma associação linear, em que cada valor de voxel f_{vn} pode ser convertido discretamente para um valor de opacidade α_{vn}, e cada f_{vn+1} pode ser convertido para α_{vn+1}. Valores intermediários possuem opacidades intermediárias, que podem ser obtidas através do vetor gradiente dos voxels, pela fórmula:

Algoritmo 3 Etapa de composição

```
fatorPiorCaso = numeroDeVoxelsPiorCaso/TamanhoDoPiorCaso
para cada linha
 para cada coluna de cada linha
  raio = aplicando variacao nos raios(linha, coluna)
  segmentoDeReta = interseccao raio com volume
  tamanho da amostra = tamanhoDoRaio * (fatorPiorCaso)
  para k=0 a tamanho da amostra
   opacidade[k] = opacidade da amostra
   cor[k] = cor da amostra
   se opacidade da amostra = 1
    sai do loop
  fim para
 fim para
fim para
```

$$\alpha(x) = \alpha_{vn+1}\left[\frac{V(x) - f_{vn}}{f_{vn+1} - f_{vn}}\right] + \left[\frac{f_{vn+1} - V(x)}{f_{vn+1} - f_{vn}}\right]$$

Toda a etapa de classificação está descrita no algorítmo 4 e no diagrama de atividades anexado, na figura 26.

Algoritmo 4 Etapa de classificação

```
se fn está entre fmin e fmax
 alpha = 1
senao se fn é menor que fmax+tolerancia ou menor que fmin-tolerancia
 alpha = ((valorDoVoxel-fmin)/(fmax-fmin))
senao
 alpha = 0
```

3.4 Shadding

O shading garante a representatividade da figura. Para demonstrar esse conceito, numa primeira abordagem foi utilizado uma iluminação plana, sem levar em consideração nenhuma informação de iluminação nem de superfícies. A imagem gerada utilizando esse processo de shading pode ser verificada na figura 14.

Outra abordagem é utilizar o modelo de iluminação de [Phong(1975)] , mas com alugumas características dos dados volumétricos. Como modelo de iluminação de Phong leva em conta a direção da normal da superfície, e os voxels não tem a noção de superfície, é utilizando o vetor gradiente do voxel para representar a sua superfície. O vetor gradiente do voxel pode ser nulo,

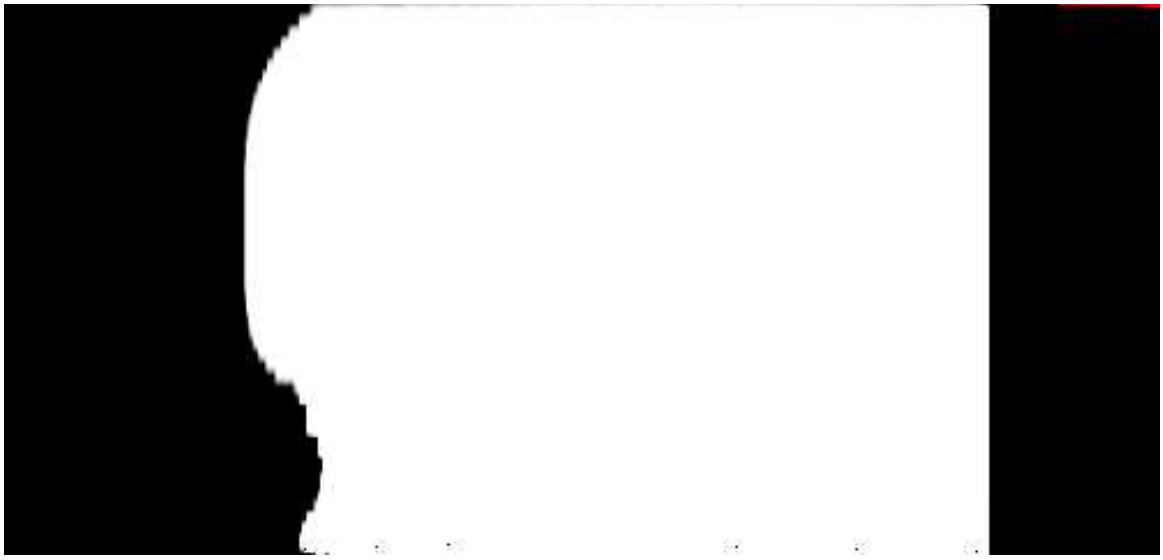


Figura 14: Crânio renderizado com shading plano.

que siginifica que o voxel está cercado por outros voxels com a mesma intensidade e cor, e não faz parte da superfície. Ter o vetor nulo significa que o voxel só vai ser influenciado pela luz ambiente e aplica a equação:

$$c_\lambda(x_i) = c_{p\lambda}k_{a\lambda}$$

Onde o termo $c_\lambda(x_i)$ é o componente λ de cor final do voxel, $c_{p\lambda}$ é a intensidade do λ componente de cor e k_a é a porcentagem refletida. Se for levada apenas em conta a luz ambiente, e sua característica de intensidade igual em todas as direções, todos os objetos da cena ficarão sem a noção de perspectiva. Seria impossível visualizar as arestas dos objetos, isso porque todas as faces estariam iluminadas por igual.

Aos demais voxels se aplica o modelo de iluniação de Phong com as variantes de especular e difusa, que é a equação:

$$c_\lambda(x_i) = c_{p\lambda}k_{a\lambda} + \frac{c_{p\lambda}}{k_1 + k_2 d(x_i)}\left[k_{d\lambda}(N(x_i) \bullet L) + k_{s\lambda}(N(x_i) \bullet H)^n\right]$$

Onde os termos k_1 e k_2 são constantes usadas para depth cueing, $d(x_i)$ é a distância perpendicular do near plane ao voxel x_i, $k_{d\lambda}$e $k_{s\lambda}$ são os coeficientes de reflexão difusa e specular, respectivamente, do \lambda componente de cor, $N(x_i)$ é a normal do voxel x_i, L é o vetor normalizado do voxel em direção à luz e H é o vetor normalizado na direção do highlight máximo.

O algorítmo 5 sintetiza a etapa de shading, que deve ser executada para cada canal de cor

(RGB por exemplo). Em anexo está o diagrama de atividades na figura 27. A figura 15 mostra o resultado final com o novo modelo de iluminação.

Algoritmo 5 Etapa de shading

```
voxel Normal = vetorGradiente.normalize()
se voxelNormal == vetor nulo
  cor = shading ambient
senao
  cor = shading ambient + shading specular e difusa
```

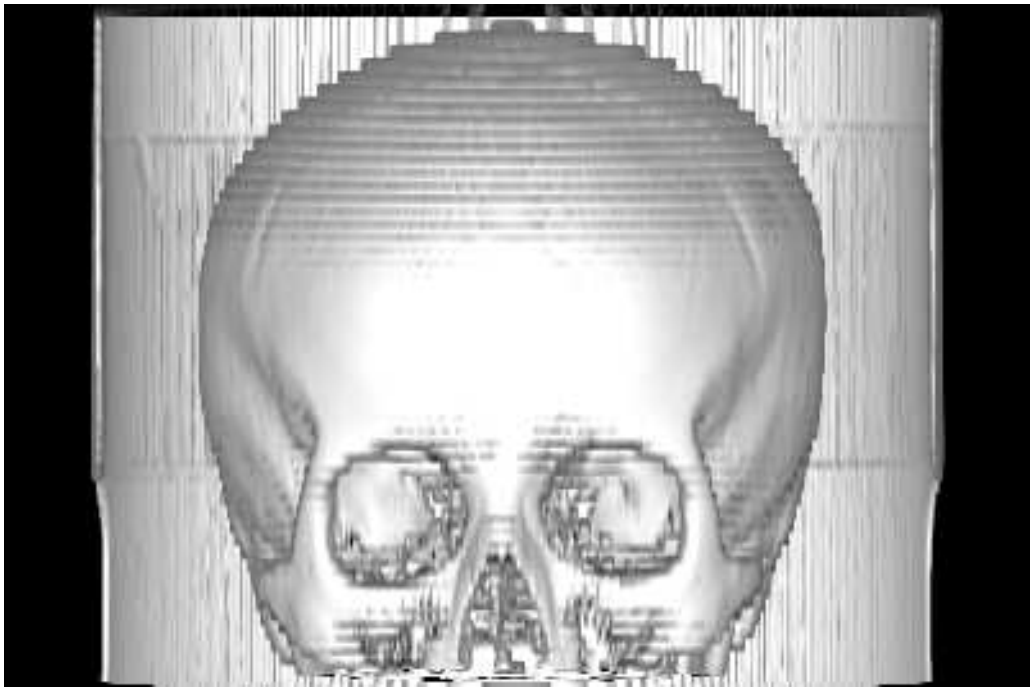


Figura 15: Crânio renderizado utilizando modelo de Phong.

Em uma segunda etapa do desenvolvimento de shading, foram atribuidas cores às diferentes faixas de opacidade. Essa abordagem traz um maior aproveitamento das características do Ray Tracing porque é possível visualizar as superfícies opacas atrás das semi transparentes. Este é um dos grandes ganhos da abordagem de seguir os raios em relação à outras, como a aproximação de malhas. O efeito pode ser observado na figura 16, onde o crânio está na cor azul, a pele está em outra faixa de cor verde.

Para que esse efeito fosse possível e não consumisse muito tempo de processamento, durante a etapa de classificação a cor do voxel é definida de acordo com a opacidade do mesmo.

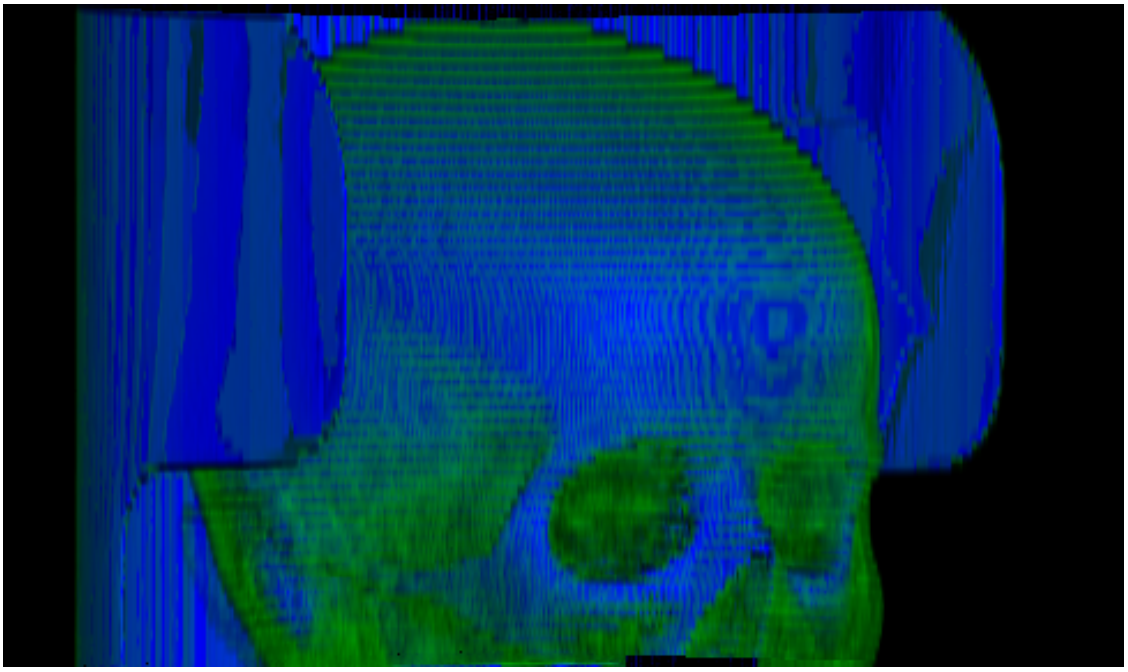


Figura 16: Crânio com pele transparente.

3.5 Junção da etapa de shading e classificação

As etapas de shading e classificação estão em processos separados no trabalho original de [Levoy(1988)] porque a possibilidade de paralelizar o código implicava numa otimização que no escopo desse trabalho não está sendo abordada. A junção das duas etapas em um único método se tornou necessária depois de constatada a necessidade de terem cores diferentes para diferentes opacidades. Como o cálculo da opacidade a partir desse momento teve que ser feito antes do shading, a junção das duas etapas em um único método algoritmico, melhorou a performance significativamente, como pode ser observado na tabela 4.

	Em diferentes métodos	No mesmo método
Média das duas etapas	4min25s	3min12s

Tabela 4: Média de tempo nas etapas de shading e composição.

Os ganhos foram conseguidos porque como a biblioteca utilizada para o desenvolvimento do trabalho é orientada a objetos, alguns métodos fazem verificações computacionalmente custosas para garantir a integridade dos dados, principalmente quando existe acesso e escrita nas informações do volume. Como essas informações são acessadas em ambos os métodos, o ganho é a redução de chamadas críticas da biblioteca usada.

3.6 Octree

A implementação do método utilizando a estrutura de dados *octree* será feita para acelerar o tempo de composição da imagem, aproveitando a grande redundância de informação que os volumes possuem, efetuando menos cálculos e conseqüentemente diminuindo o gargalo de tempo de processamento existente na aplicação. O uso dessa estrutura de dados vai acelerar as as etapas de *shading*, classificação e composição, sendo necessário um pré-processamento na etapa de preparação do volume para montar a estrutura interna da *octree*.

Na etapa de praparação do volume, após a leitura do volume do disco para a memória, será criada uma *octree* a partir desse volume. A octree vai recursivamente dividir o volume em nodos, levando em consideração um valor de espaçamento entre os pixels que será passado como parâmetro no construtor da classe. Esse espaçamento dos pixels determinará o grau de fidelidade que a octree representará o volume original, quanto mais próximo o espaçamento da octree corresponder ao do volume, mais semelhantes à imagem final estão. Em contrapartida, quanto mais nodos são criados, maior é o consumo de memória da estrutura, sendo este um fator determinante no desempenho da aplicação.

Na etapa de preparação do volume então não houveram muitas mudanças, a biblioteca de computação gráfica do laboratório Telemedicina, forneceu um construtor trivial que foi utilizado nessa implementação.

Na etapa de *shading* e composição, a mudança fundamental foi adaptar os métodos para trabalharem sobre os nodos, e não sobre os voxels. Isso porque no volume original, a menor parte do volume era o voxel, que possui uma representação espacial nas coordenadas (x,y,z), analogamente ao pixel nas imagens bidimensionais. Mas a menor parte da octree é um nodo, que é composto por diversos voxels, e se comporta de maneira semelhante a este. O tratamento adotado para adaptar o método ao uso da octree, foi mudar o método de iteração dos voxels do volume original, para iterar sobre os nodos da octree. O ganho em performance desse método depende muito do espaçamento de *pixels* utilizados e da quantidade de memória disponível no hardware, mas é significativo quando pensamos que as chamadas dos métodos são as mesmas, só que para um conjunto muito menor de dados.

Na etapa de composição da imagem, a mesma heurística anterior foi mantida. O número de amostras por raio continua sendo determinado em função do pior caso, conforme explicado na página 3.2, mas com uma checagem a mais: Se a amostra anterior está no mesmo nodo da anterior, então os valores de cor e opacidade são iguais.

4 Resultados

Com a implementação da técnica é possível renderizar informação volumétricas. Exames de tomografia foram renderizados usando a técnica para avaliar se os resultados eram os esperados ou não, como a figura 17 que é o exame de tomografia de um corpo de prova, com as artérias renderizadas na cor branca e opaca.

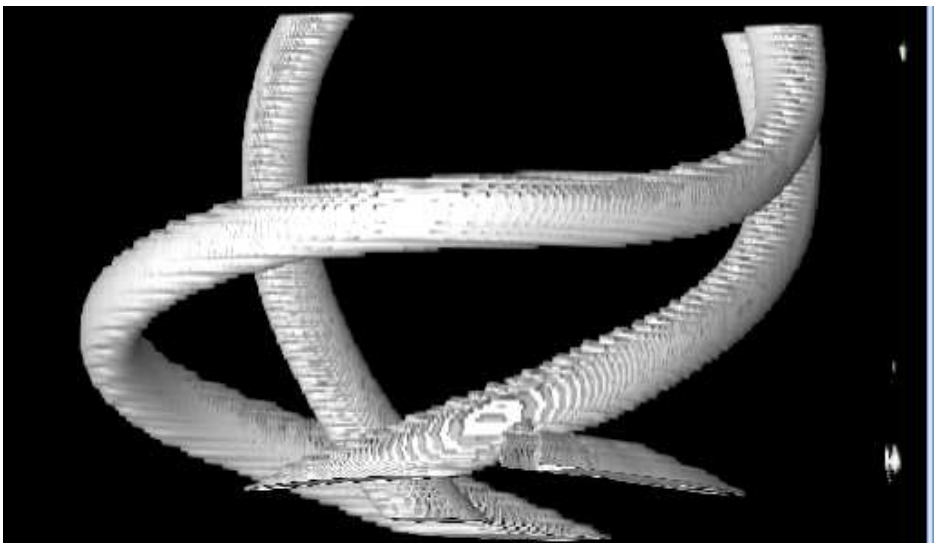


Figura 17: Phantoms simulando artérias.

A imagem 19 mostra um exame de tomografia com dimensões 512x512x50 de um crânio renderizado com o algoritmo de Ray Tracing, mas ainda sem nenhuma configuração especial de opacidade. O mesmo volume renderizado com a técnica Marching Cubes pode ser observado na imagem 18.

Um grande passo em direção à qualidade foi obtido quando a seleção de opacidade para as diferentes sessões do volume foi definitivamente desenvolvida. As imagens 20, 21 e 22 mostram o mesmo volume renderizado em 19 mas, respectivamentee, com a progressão da opacidade da pele indo de 0.2, para 0.4 e 1.0, que é totalmente opaco.

Em relação à performance da composição das imagens, deve-se ressaltar que todos os testes de foram realizados num AMD64 3200+ com 1GB de memória DDR e placa de vídeo nVidia

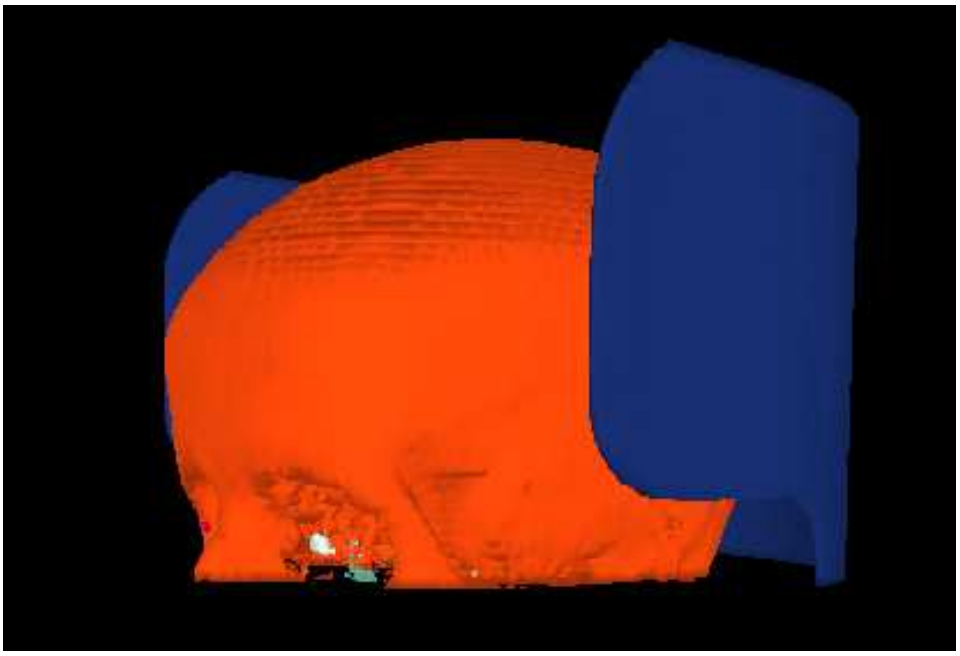


Figura 18: Crânio renderizado com Marching Cubes

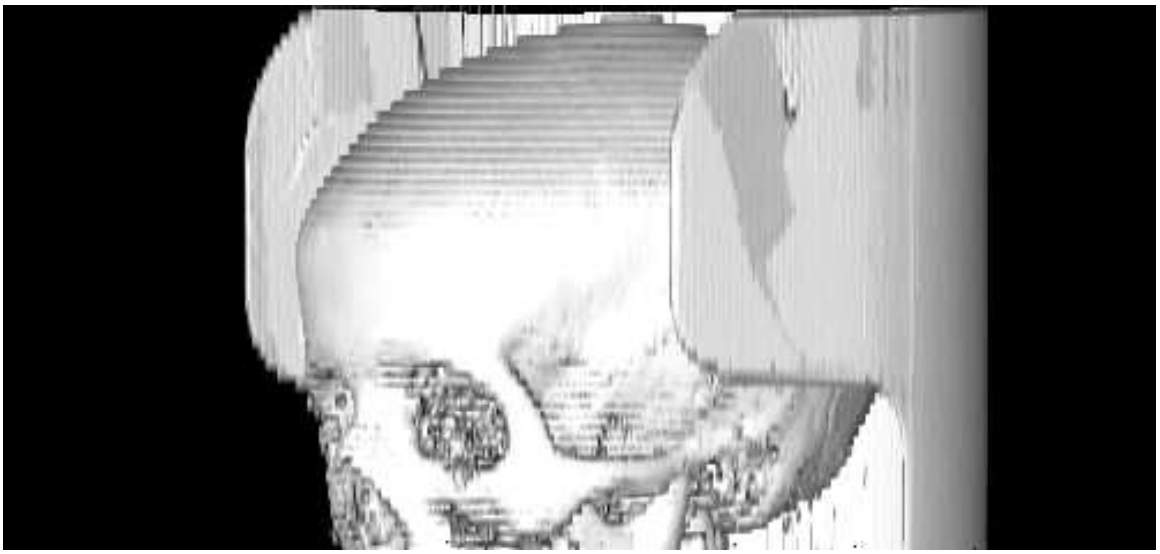


Figura 19: Crânio renderizado com Ray Tracing.

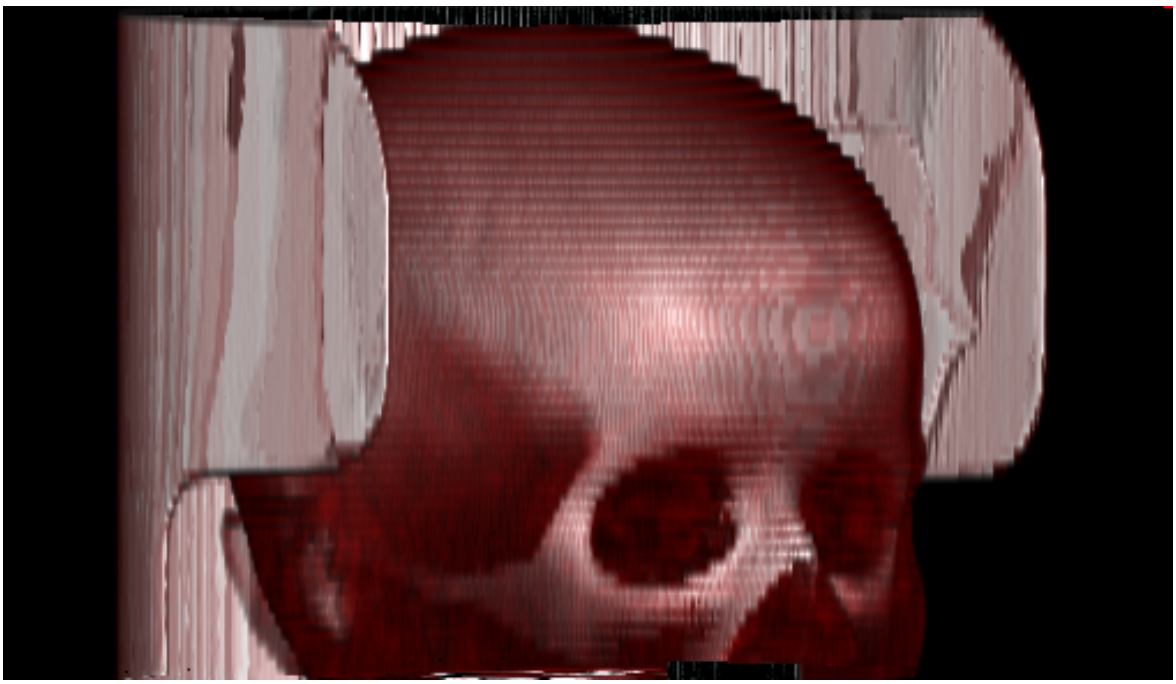


Figura 20: Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, com opacidade 0.2

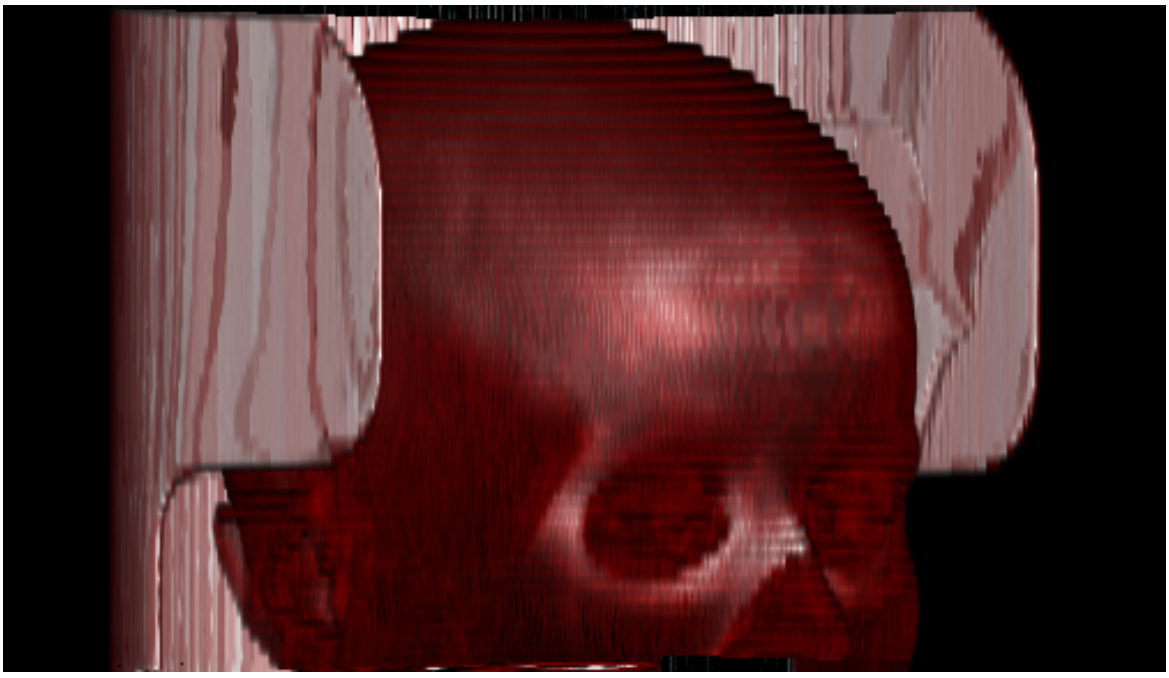


Figura 21: Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, com opacidade 0.4

Figura 22: Crânio renderizado com Ray Tracing. A pele está renderizada em vermelho, completamente opaca.

GeForce 6200 64MB.

O tempo de geração de imagens com diferentes dimensões para diferentes resoluções podem ser observadas na tabela 5.

	Shading e classificação	Composição	Total
512 x 512 x 10 - 256 x 256	40s	10s	50s
512 x 512 x 50 - 256 x 256	2m53s	36s	3m29s
512 x 512 x 10 - 512 x 512	40s	20s	1m
512 x 512 x 50 - 512 x 512	3m	1m50s	4m50s

Tabela 5: Tempo de shading, classificação e composição para diferentes dimensões de volume e diferentes resoluções de imagem.

4.1 Octree

O maior limitante para utilização da estrutura octree é o consumo de memória. O mesmo computador com processador AMD64 3200+, placa de vídeo nVidia GeForce 6200 64MB e agora com 3GB de memória DDR foi utilizado para a realização dos testes.

As configurações da octree são fundamentais para o entendimeno do teste. Para gerar uma imagem com uma qualidade similar à gerada pelo algoritmo sem octree seria necessário configurá-la para levar em consideração todos os voxels do volume original no momento da criação

dos nodos. Como essa configuração faz com que o consumo de memória extrapole os 3GB para o volume da figura 21, com dimensoes 512x512x50, o teste será realizado com a diminuição da qualidade e consequentemente a diminuição da memória usada.

O espaçamento original dos voxels é (0.5, 0.5, 3), isto quer dizer que a distância entre um voxel e outro no eixo X é 0.5, no eixo Y é 0.5 e no eixo Z é 3.0 unidades de cena. Os testes foram realizados com espaçamento em (2,2,12) e (4,4,24). A tabela 6 mostra o tempo de processamento e o uso de memória para esses testes. O resultado gráfico do uso de espaçamento (2,2,12) pode ser analisado na imagem 23 e com espaçamento (4,4,24) na imagem 24.

	Uso de Memória (MB)	Tempo total
Tempo sem Octree	280	4m50s
Pixel Spacing = (2, 2, 12)	1447	2m27s
Pixel Spacing = (4, 4, 24)	283	58s

Tabela 6: Tempo de renderização e consumo de memória do modelo utilizando octree.

Figura 23: Crânio renderizado utilizando a estrutura de dados Octree. Com espaçamento entre voxels (2,2,12).

Figura 24: Crânio renderizado utilizando a estrutura de dados Octree. Com espaçamento entre voxels (4,4,24).

5 *Conclusão e Trabalhos futuros*

O trabalho cumpriu o seu compromisso em implementar a técnica. Os resultados gráficos foram satisfatórios uma vez que é a técnica proporciona uma maior representatividade de informações da cena, como transparência e faixa de transição entre diferentes intervalos do volume original. O resultado final ao ser comparado com a imagem gerada pelo algorimto de Marching Cubes conseguiu ter maior expressividade graças ao seu maior potencial de qualidade e representatividade das informações. A implementação modular é expansível a novas técnicas de iluminação que podem agregar maiores melhorias no sentindo de aumentar a representatividade das informações, de acordo com o critério a ser analisado na imagem gerada.

O tempo para geração das imagens ainda não atingiu resultados que a qualifiquem como apta ao uso em aplicações em tempo real, não da maneira como está implementada hoje, necessitando de maiores melhorias e otimizações além das que foram abordadas.

O uso da estrutura de dados octree melhorou o tempo de processamento mas consumiu muita memória, mesmo três gigabytes de memória não foram suficientes para realizar os testes com a qualidade máxima da imagem. Embora hoje o limitador para o uso da octree sejá a disponibilidade de memória, é factível dizer que quando a tecnologia evoluir e a quantidade de memória disponível for maior, a octree será um importante aliado para obtenção de tempos menores de processamento em busca da viabilidade do uso da técnica em aplicações em tempo real.

Uma possível melhoria para a implementação da técnica é o uso da capacidade computacional das placas de vídeo, através de programação voltada à GPU (*Graphics Process Unit*), que já é uma realidade a muitos sistemas de computação gráfica. Essa implementação pode melhorar muito o desempenho do algoritmo, aproveitando o paralelismo da técnica e os resultados esperados são promissores, dada a capacidade de processamento das GPUs atuais.

Um outro trabalho futuro seria utilizar as imagens geradas pela técnica e usar como textura 3D em aplicações que já são em tempo real, que utilizam técnicas como o Marching Cubes. Em uma etapa de pré-processamento imagens são geradas a partir da técnica implementada

nesse trabalho, e armazenadas para serema aplicadas ao modelo tridimensional de malha gerados pela aproximação do Marching Cubes. Essa aplicação aproveitaria a qualidade gráfica do RayTracing e ainda seria em tempo real e renderizada no pipeline convencional de malhas.

Referências

[Akenine-Müller and Haines(1999)] Tomas Akenine-Müller and Eric Haines. *Real-Time Rendering*. AK Peters, Ltd., Wellesley, MA, USA, 1999. ISBN 1568811829.

[Che et al.(1985)Che, Herman, Reynolds, and Udupa] Lih-Shyang Che, G.T. Herman, R.A. Reynolds, and J.K. Udupa. Surface shading in the cuberille environment. *Computer Graphics and Applications, IEEE*, 5(12):33–43, 1985. ISSN 0272-1716.

[Elvins(1992)] T. Todd Elvins. A survey of algorithms for volume visualization. *Computer Graphics*, 26(3):194–201, 1992. URL `citeseer.ist.psu.edu/elvins92survey.html`.

[Foley et al.(1990)Foley, van Dam, Feiner, and Hughes] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0-201-12110-7.

[Lacroute and Levoy(1994)] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-667-0. doi: http://doi.acm.org/10.1145/192161.192283.

[Levoy(1988)] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/38.511.

[Lorensen and Cline(1987)] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press. ISBN 0-89791-227-6. doi: http://doi.acm.org/10.1145/37401.37422.

[Ohbuchi et al.(1992)Ohbuchi, Chen, and Fuchs] Ryutarou Ohbuchi, David Chen, and Henry Fuchs. Incremental volume reconstruction and rendering for 3D ultrasound imaging. Technical Report TR92-037, 1, 1992. URL `citeseer.ist.psu.edu/ohbuchi92incremental.html`.

[Parker et al.(1999)Parker, Parker, Livnat, Sloan, and Shirley] Steven Parker, Michael Parker, Yarden Livnat, Peter-Pike Sloan, and Peter Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238–250, /1999. URL `citeseer.ist.psu.edu/article/parker99interactive.html`.

[Phong(1975)] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/360825.360839.

[Porter and Duff(1984)] Thomas Porter and Tom Duff. Compositing digital images. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, New York, NY, USA, 1984. ACM Press. ISBN 0-89791-138-5. doi: http://doi.acm.org/10.1145/800031.808606.

[Ray et al.(1999)Ray, Pfister, Silver, and Cook] Harvey Ray, Hanspeter Pfister, Deborah Silver, and Todd A. Cook. Ray casting architectures for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):210–223, /1999.

[Schröder and Stoll(1992)] Peter Schröder and Gordon Stoll. Data parallel volume rendering as line drawing. In *VVS '92: Proceedings of the 1992 workshop on Volume visualization*, pages 25–32, New York, NY, USA, 1992. ACM Press. ISBN 0-89791-527-5. doi: http://doi.acm.org/10.1145/147130.147142.

[Wilson et al.(1994)Wilson, VanGelder, and Wilhelms] Orion Wilson, Allen VanGelder, and Jane Wilhelms. DIRECT VOLUME RENDERING VIA 3D TEXTURES. Technical Report UCSC-CRL-94-19, 1994. URL `citeseer.ist.psu.edu/wilson94direct.html`.

[Yagel and Kaufman(1992)] Roni Yagel and Arie Kaufman. Template-based volume viewing. volume 11, pages 153–167, 1992. URL `citeseer.ist.psu.edu/yagel92templatebased.html`.

ANEXO A – Diagramas de atividade

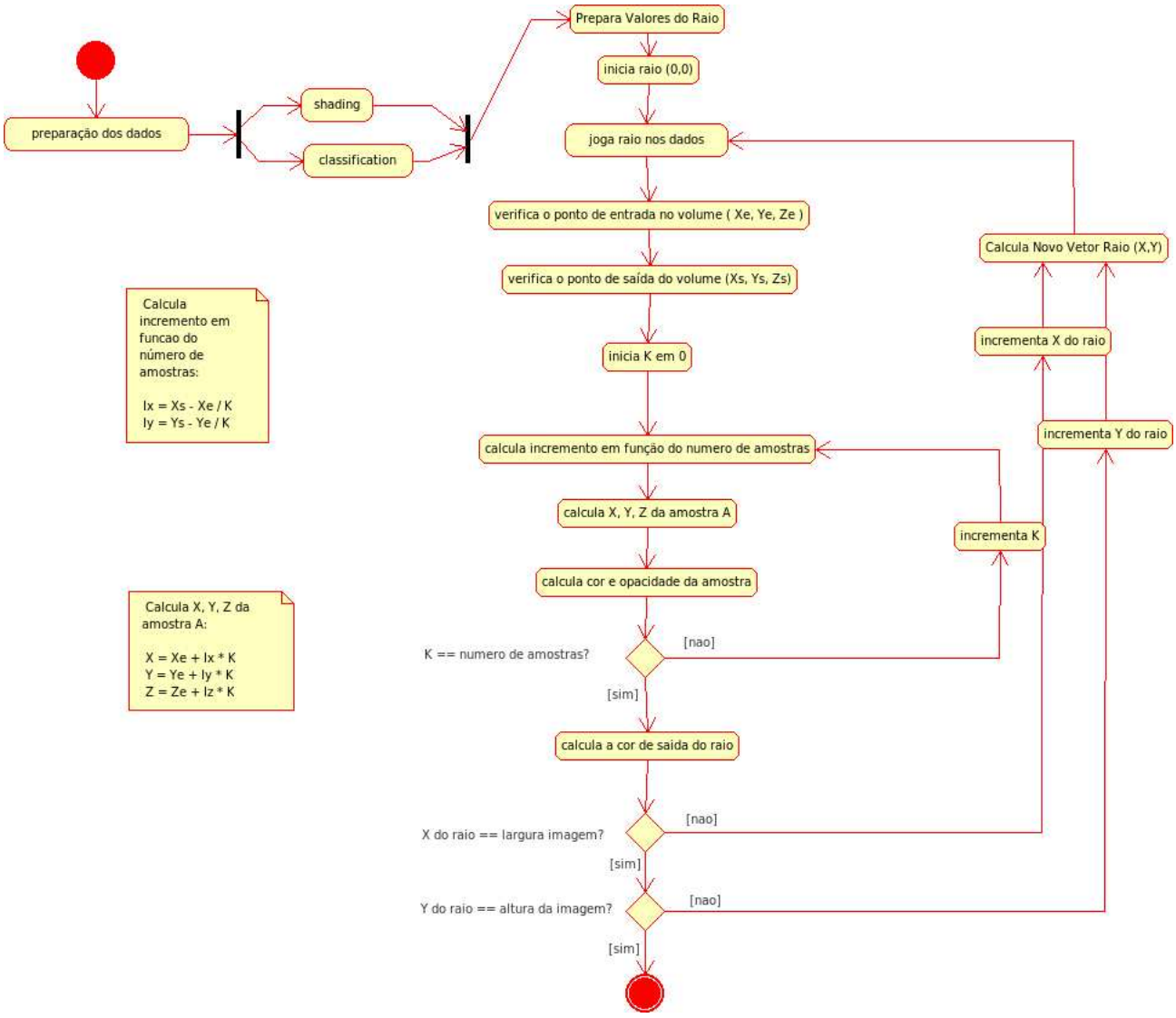


Figura 25: Diagrama de atividades composição

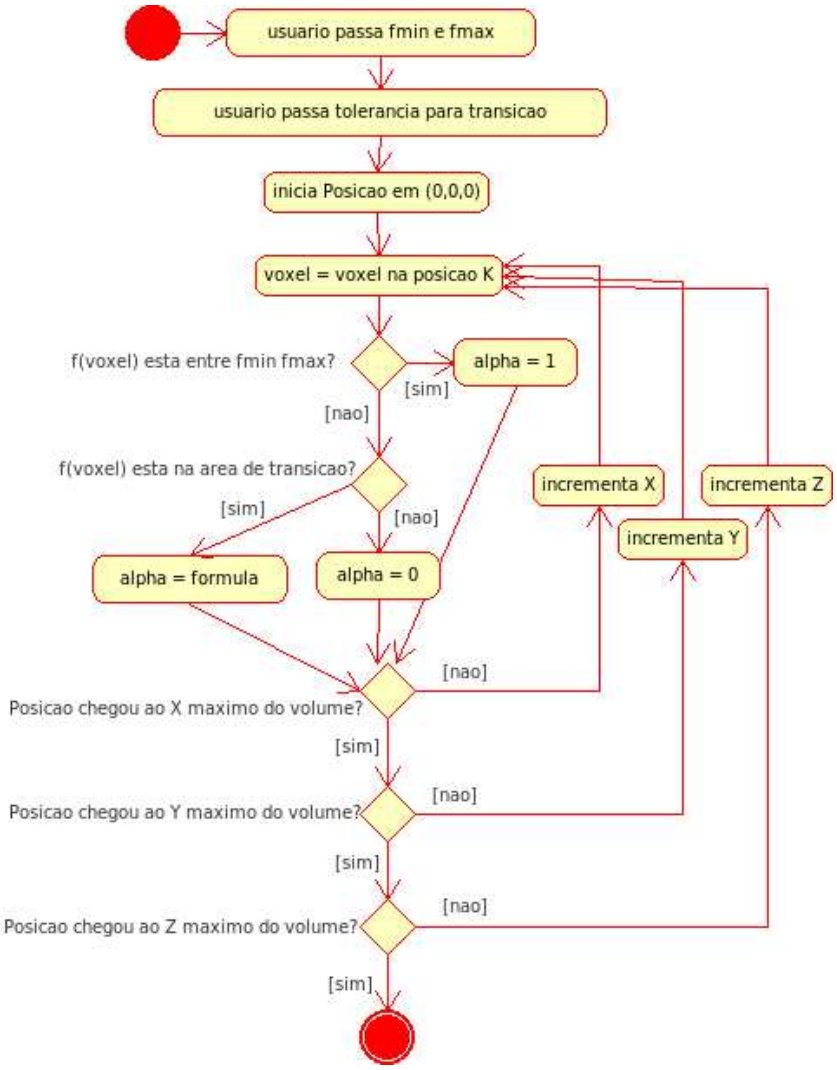


Figura 26: Diagrama de atividades classificação.

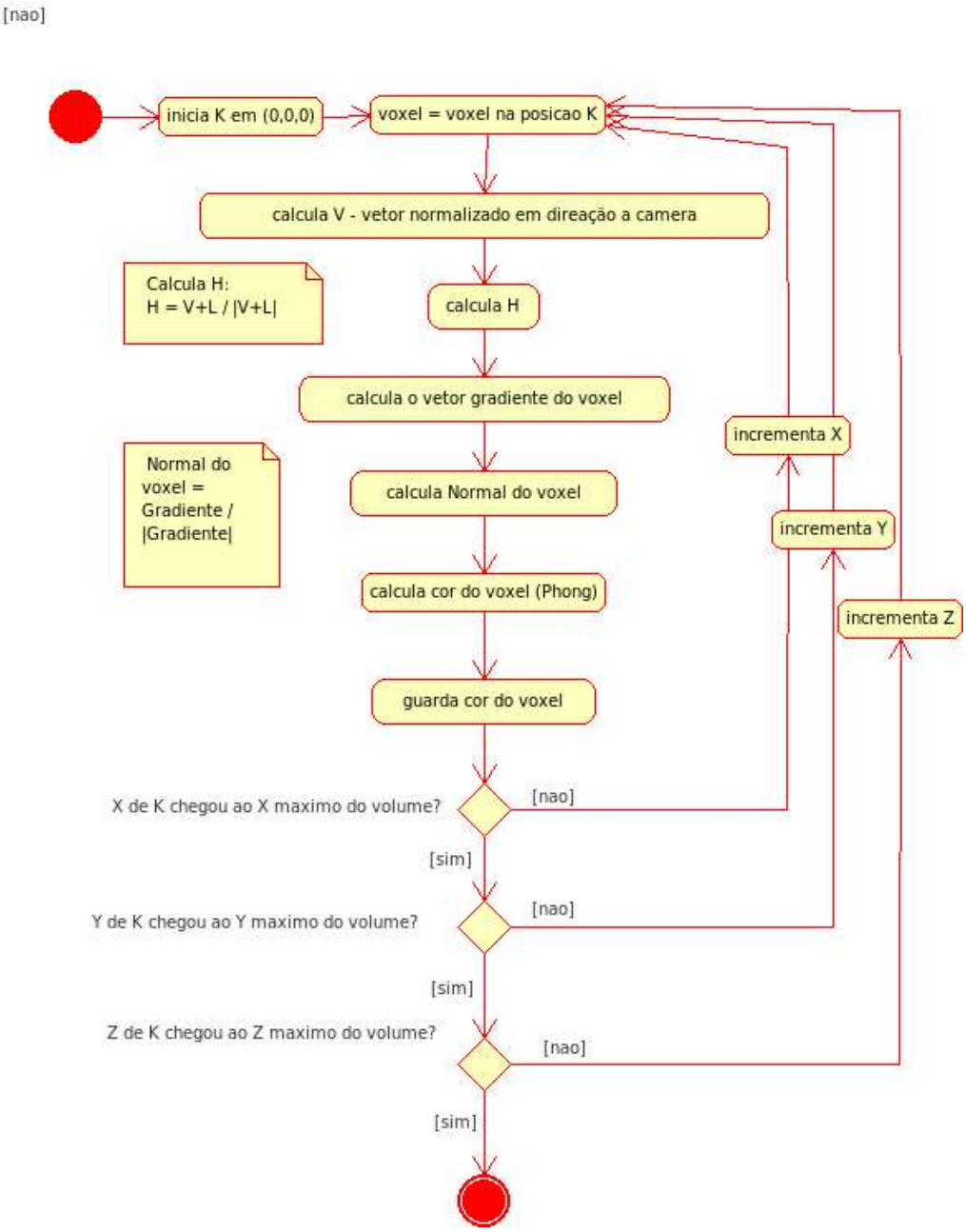


Figura 27: Diagrama de atividades shading.

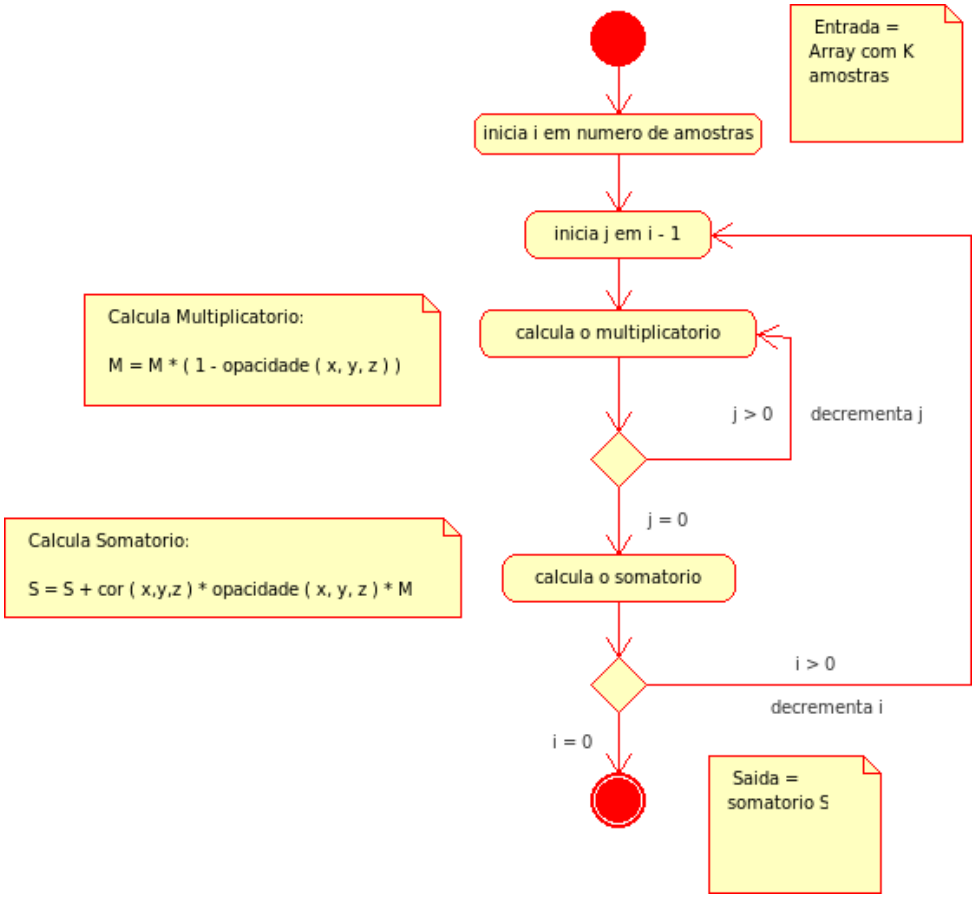


Figura 28: Diagrama de atividades do cálculo da cor de saída do raio.