"RAY TRACING" OTIMIZADO DE SÓLIDOS CSG USANDO OCTREES

João Luiz Dihl Comba

Ronaldo C. Marinho Persiano

COPPE-Sistemas, UFRJ
Caixa Postal 68511
Rio de Janeiro - RJ - 21945

RESUMO: A obtenção de cenas realistas é uma das tarefas mais importantes em ambiente de síntese de imagens. O tradicional algoritmo de "Ray Tracing" mostra-se, na sua versão canônica, muito ineficiente. Neste trabalho é proposta uma otimização deste algoritmo para cenas descritas em CSG ("Constructive Solid Geometry"). Uma estrutura híbrida entre Octree e CSG é utilizada e o processo de visualização é tornado mais eficiente utilizando-se uma variação do algoritmo Bresenham para traçado de linhas (Bresenham Multi-Resolução).

1. INTRODUÇÃO

'"Rendering" é a área da Computação Gráfica que estuda o processo de produção de imagens ou figuras realistas' (Rogers [1]). O interesse por esta área tem crescido a medida que diversas aplicações começam a fazer uso de imagens reais produzidas por computador. As mais conhecidas aplicações são encontradas nos campos da Publicidade, Cinema, Televisão, Arquitetura, etc. Estudar as diversas alternativas de produção de imagens realistas torna-se, portanto, objeto de alto interesse.

"Ray Tracing" é o tradicional algoritmo para a produção de cenas realistas. Inicialmente introduzido por Appel [2] como um método para remoção de superfícies ocultas, foi revisado por Whitted [3] para trabalhar com modelos globais de iluminação, e assim permitir a geração de cenas realistas. A idéia principal deste método consiste na simulação da interação da luz com uma determinada cena, usando os conceitos da Ótica Geométrica.

Os resultados produzidos por este algoritmo são excelentes, entretanto o alto custo computacional exigido tornou-se sua maior desvantagem. Diversas propostas de aceleração do método podem ser encontradas na literatura, desde o uso de técnicas de coerência até o uso de arquiteturas paralelas.

Neste artigo iremos apresentar um método de otimização para cenas compostas de sólidos descritos no modelo de representação CSG ("Constructive Solid Geometry" [4]). A idéia central do método consiste em subdividir a cena de forma a simplificá-la, utilizando para isto uma estrutura híbrida entre CSG e Octree [5]. A visualização desta estrutura será feita utilizando uma variação do algoritmo Bresenham de traçado de linhas, estendido para o espaço tridimensional e trabalhando com células de tamanho variável (*Bresenham Multi-Resolução*).

2. ALGORITMO CANÔNICO PARA "RAY TRACING" DE SÓLIDOS CSG

2.1. O Modelo CSG para representação de sólidos

O modelo CSG para representação de sólidos é um dos mais usados em vários sistemas de modelagem de sólidos. Seus objetos são descritos segundo composições de sólidos primitivos, combinados segundo os operadores booleanos *União, Interseção* e *Diferença* (Requicha [4]). A figura 1 mostra um exemplo de um sólido CSG.

"RAY TRACING" OTIMIZADO DE SÓLIDOS CSG USANDO OCTREES

João Luiz Dihl Comba

Ronaldo C. Marinho Persiano

COPPE-Sistemas, UFRJ
Caixa Postal 68511
Rio de Janeiro - RJ - 21945

RESUMO: A obtenção de cenas realistas é uma das tarefas mais importantes em ambiente de síntese de imagens. O tradicional algoritmo de "Ray Tracing" mostra-se, na sua versão canônica, muito ineficiente. Neste trabalho é proposta uma otimização deste algoritmo para cenas descritas em CSG ("Constructive Solid Geometry"). Uma estrutura híbrida entre Octree e CSG é utilizada e o processo de visualização é tornado mais eficiente utilizando-se uma variação do algoritmo Bresenham para traçado de linhas (Bresenham Multi-Resolução).

1. INTRODUÇÃO

'"Rendering" é a área da Computação Gráfica que estuda o processo de produção de imagens ou figuras realistas' (Rogers [1]). O interesse por esta área tem crescido a medida que diversas aplicações começam a fazer uso de imagens reais produzidas por computador. As mais conhecidas aplicações são encontradas nos campos da Publicidade, Cinema, Televisão, Arquitetura, etc. Estudar as diversas alternativas de produção de imagens realistas torna-se, portanto, objeto de alto interesse.

"Ray Tracing" é o tradicional algoritmo para a produção de cenas realistas. Inicialmente introduzido por Appel [2] como um método para remoção de superfícies ocultas, foi revisado por Whitted [3] para trabalhar com modelos globais de iluminação, e assim permitir a geração de cenas realistas. A idéia principal deste método consiste na simulação da interação da luz com uma determinada cena, usando os conceitos da Ótica Geométrica.

Os resultados produzidos por este algoritmo são excelentes, entretanto o alto custo computacional exigido tornou-se sua maior desvantagem. Diversas propostas de aceleração do método podem ser encontradas na literatura, desde o uso de técnicas de coerência até o uso de arquiteturas paralelas.

Neste artigo iremos apresentar um método de otimização para cenas compostas de sólidos descritos no modelo de representação CSG ("Constructive Solid Geometry" [4]). A idéia central do método consiste em subdividir a cena de forma a simplificá-la, utilizando para isto uma estrutura híbrida entre CSG e Octree [5]. A visualização desta estrutura será feita utilizando uma variação do algoritmo Bresenham de traçado de linhas, estendido para o espaço tridimensional e trabalhando com células de tamanho variável (*Bresenham Multi-Resolução*).

2. ALGORITMO CANÔNICO PARA "RAY TRACING" DE SÓLIDOS CSG

2.1. O Modelo CSG para representação de sólidos

O modelo CSG para representação de sólidos é um dos mais usados em vários sistemas de modelagem de sólidos. Seus objetos são descritos segundo composições de sólidos primitivos, combinados segundo os operadores booleanos *União, Interseção* e *Diferença* (Requicha [4]). A figura 1 mostra um exemplo de um sólido CSG.

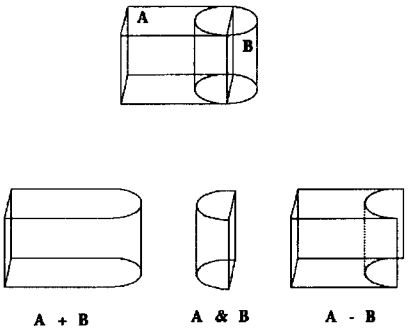

Figura 1 - Sólido descrito segundo o modelo CSG

A estrutura de dados mais adequada para armazenar esta composição de sólidos é uma árvore binária, cujas folhas são sólidos primitivos. Os nodos não terminais são operadores booleanos, e a raiz da arvore representa toda a composição. Esta estrutura é denominada Árvore CSG.

A implementação do modelo CSG segue o mesmo formato proposto por Roth [6]. Os sólidos primitivos que estão disponíveis são:

- BLOCO : Bloco cúbico de lado unitário, situado no primeiro octante e com um vértice na origem
- ESFERA : Esfera de raio unitário centrada na origem
- CILINDRO : Cilindro de raio e altura unitários com a base sobre o plano Z = 0 e centrado sobre o exio coordenado Z
- PLANO : Semiespaço plano correspondente à inequação Z >= 0.

Ós sólidos primitivos em suas posições e tamanhos originais não são gerais o suficiente. Portanto, e necessário que se disponha de operadores de instanciação capazes de deformá-los e posicioná-los dentro do universo de interesse. Para tanto, podem ser aplicadas as seguintes transformações geométricas:

- Translação (x, y, z) : Mover a primitiva no espaço x unidades no eixo X, y unidades no eixo Y e z unidades no eixo Z
- Rotação (α, β, γ) : Rotacionar a primitiva α graus em torno do eixo X, β em torno do eixo Y e γ em torno do eixo Z
- Escala (x, y, z) : Escalar a primitiva x unidades no eixo X, y unidades no eixo Y e z unidades no eixo Z

Sólidos mais complexos, como por exemplo elipsóides, podem ser gerados ao se aplicar tais tranformações geométricas. Inicialmente cada primitiva está definida no Sistema de Coordenadas da Primitiva (SCPrimitiva). Através da aplicação da Matriz de Transformações Geométricas (MTG), a primitiva é levada para o Sistema de Coordenadas da Cena (SCCena). Para facilitar o trabalho do algoritmo de visualização, a matriz inversa da MTG é calculada, responsável por levar uma primitiva do SCCena para o SCPrimitiva.

2.2. "Ray Casting" de sólidos CSG

"Ray Casting" pode ser considerado como o "Ray Tracing" somente para remoção de superfícies ocultas. A utilização deste algoritmo para sólidos CSG foi proposta inicialmente por Roth [6]. A idéia básica do algoritmo pode ser explicada pela figura 2.

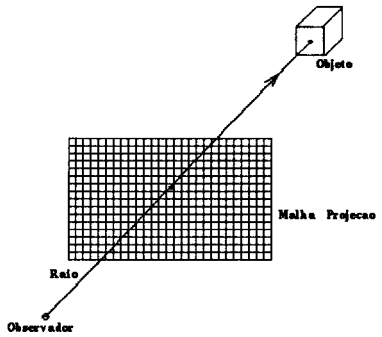

Figura 2 - Idéia básica do algoritmo de "Ray Casting"

O plano de projeção é uma malha retangular, e para cada célula desta malha um raio é atirado segundo os parâmetros de projeção definidos. Busca-se descobrir para cada raio a primeira interseção deste com a cena. Para tanto, são necessários testes de interseção entre raios e os diversos componentes da cena.

Dois problemas surgem para se jogar um raio em um árvore CSG. O primeiro refere-se ao cálculo de interseção entre o raio e a primitiva. Como as primitivas são passíveis de transformações geométricas, podemos ter testes de interseção de raios contra sólidos mais complexos (elipsóides, romboedros). A idéia de Roth [6] simplificou bastante o problema. Sua proposta consiste em se fazer o teste de interseção no SCPrimitiva ao invés de se fazer no SCCena. Para tanto, basta aplicarmos a MTG Inversa sobre o raio, levando-o para o SCPrimitiva. Neste sistema o teste torna-se bastante simples.

O segundo problema consiste em descobrir qual ponto da cena CSG foi primeiramente interceptado pelo raio. É necessário classificar cada primitiva contra o raio em questão, gerando assim uma lista de interseções. Esta lista de interseções define os intervalos em que o raio está dentro da primitiva.

O algoritmo básico consiste em combinar estas listas de forma que tenhamos os intervalos em que o raio esta dentro do sólido CSG. Para tanto, usa-se uma aritmética intervalar para combinar as listas. Este é um algoritmo essencialmente recursivo, cujo núcleo básico deve ser capaz de conbinar duas subárvores segundo um operador booleano (figura 3).

2.3. Usando um modelo de iluminação global

Um modelo de iluminação permite a determinação da luz refletida para o observador em cada ponto da imagem. Este pode ser invocado localmente ou globalmente. Invocado localmente, somente a luz incidente de uma fonte de luz e a orientação da superfície são usados pelo modelo de iluminação para determinar a intensidade de luz refletida ao observador. Invocado globalmente, a luz que alcança este ponto por reflexão ou refração, bem como a contribuição de diversas fontes de luz, são também consideradas (Rogers [1]).

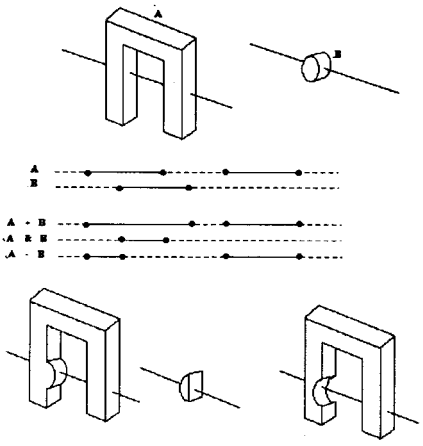

Figura 3 - Exemplo de "Ray Casting" de sólidos CSG

O primeiro algoritmo de "Ray Tracing" que utilizava modelos de iluminação global foi proposto por Whitted [2]. A intensidade de luz que atinge um determinado ponto é calculada neste modelo da seguinte forma (para mais informações ver Rogers [1]):

$$I = K_a I_a + K_d \sum_{j=1}^{NroFL} I_{lj}(N \cdot L_j) + K_s \sum_{j=1}^{NroFL} I_{lj}(S \cdot R_j)^n + K_s I_s + K_t I_t$$

Onde:

K_a, K_d, K_s *e* K_t = *Coeficientes de Reflexao Ambiente, Reflexao Difusa, Reflexao Especular e Refracao*

I_a = *Intensidade Ambiente*

I_{lj} = *Intensidade da j-esima fonte de luz*

I_s = *Intensidade proveniente do acompanhamento de reflexoes,*

I_t = *Intensidade proveniente do acompanhamento de refracoes*

N = *normal a superficie*

Lj = *j-esima fonte de luz*

$NroFL$ = *numero de fontes de luz*

n = *constante especular de Phong*

Diferentemente do "Ray Casting", os cálculos do modelo de iluminação não terminam no primeiro cálculo de interseções. Neste ponto de interseção dois raios adicionais são gerados, referentes a reflexão e refração do raio neste ponto. Estes dois raios são acompanhados para encontrar suas interseções com outros objetos na cena. O processo é essencialmente recursivo, e termina quando os raios saem da cena.

Portanto, a cada interseção dois novos raios podem ser gerados. Isto faz com que tenhamos uma árvore de raios refletidos e refratados (Árvore RT). Após a montagem desta árvore, o modelo de iluminação é aplicado recursivamente em cada nodo da árvore (figura 4).

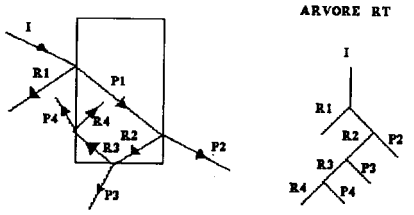

Figura 4 - Arvore de Reflexões e Refrações

3. GERAÇÃO DA ESTRUTURA HÍBRIDA OCTREE-CSG

O uso de Octrees para acelerar o algoritmo de "Ray Tracing" já foi proposto por Glassner [7], Fujimoto et all [8] e Wyvill e Kunii [9]. Octree é um modelo para representação de sólidos baseado na subdivisão recursiva do espaço em oito octantes que são armazenados em uma árvore octária (figura 5).

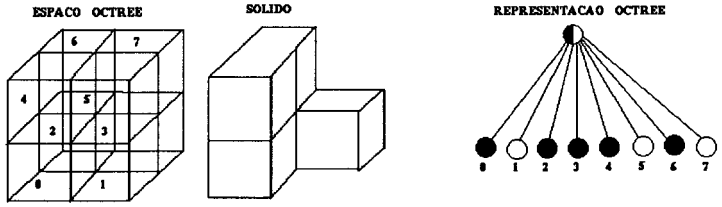

Figura 5 - Sólido descrito segundo o modelo Octree

Na sua proposta mais simples, existem três tipos de nodos na Octree:

- Nodo Misto: Indica que a porção do espaço faz interseção parcial com o objeto, sendo subdivida em oito octantes
- Nodo Cheio: Indica que a porção do espaço é totalmente preenchida pelo objeto
- Nodo Vazio: Indica que a porção do espaço não faz interseção com o objeto

A utilização de Octrees tradicionais em "Ray Tracing" possue sérios problemas, pois a informação da normal ao sólido é perdida no processo de sua geração (pois o sólido obtido é apenas uma discretização do sólido original). A saída para tal problema consiste na criação de tipos de nodos mais

complexos, onde poderão ser obtidas informações à respeito das normais dos sólidos.

As propostas de Glassner e Fujimoto são parecidas na medida que estes não trabalham com modelos de representação de sólidos, enquanto que o trabalho de Wyvil e Kunii trabalha com o modelo CSG. Suas idéias compartilham o fato de que é necessário criar um novo tipo de nodo na Octree, agindo como um ponteiro para informações sobre o objeto.

Glassner propôs que a Octree seja dividida até que um número x de objetos estivesse dentro de uma célula, associando a este nodo a lista para os descritores físicos de tais objetos. Fujimoto propôs que fosse criado um caso particular de Octree (uma malha cúbica), associando a cada célula desta malha a lista de objetos que fazem interseção com ela.

A proposta de Wyvill e Kunii é a de se criar uma estrutura híbrida de Octree com CSG. Esta estrutura foi chamada de RCSG ("Reduced CSG") pois para cada voxel da Octree temos apenas uma subárvore da Árvore CSG, com as primitivas que fazem alguma interseção com ele. A estrutura gerada combina, portanto, características dos modelos Octree e CSG (figura 6).

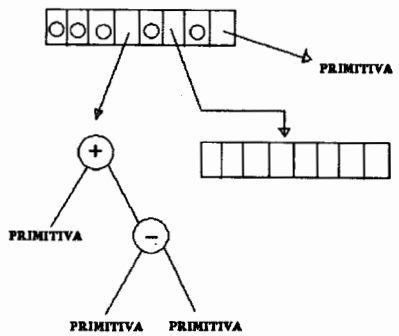

Figura 6 - Descrição da estrutura híbrida Octree-CSG

O processo de subdivisão de uma Árvore CSG consiste em classificar cada voxel da Octree contra a Árvore CSG. Esta classificação é feita com as primitivas da Árvore CSG, verificando os seguintes casos:

- Primitiva não intercepta voxel: A primitiva é substituída na Árvore CSG como sendo uma primitiva vazia
- Primitiva engloba o voxel: A primitiva é substituída na Árvore CSG como sendo uma primitiva universo
- Primitiva faz interseção parcial com o sólido: A primitiva é mantida inalterada na árvore CSG

A classificação entre voxel e primitiva não é um processo trivial. Usando o mesmo artifício apresentado anteriormente, os testes de interseção são feitos no SCPrimitiva. Os testes de interseção necessários são os seguintes: Romboedro x Esfera, Romboedro x Cilindro e Romboedro x Cubo. Verificar se a primitiva engloba o rombuedro (todos os vértices do rombuedro estão dentro da primitiva), ou se seguramente fazem interseção parcial (alguns vértices estão fora, outros dentro) é trivial. O mais complicado neste processo consiste em distinguir entre outra possibilidade de interseção parcial e a interseção vazia (todos vértices do romboedro estão fora da primitiva). Neste caso deve-se descobrir a face ou aresta do romboedro mais próxima da primitiva, calculando-se o teste de interseção contra a primitiva (reta x primitiva, plano x primitiva).

Feito esta classificação, devemos submeter a nova Árvore CSG a um algoritmo de simplificação, que reduzirá a árvore CSG. Basicamente este algoritmo usará os conceitos de simplificação da álgebra

booleana. Ao final, teremos a Árvore CSG localizada em relação ao voxel. Ápos o proceso três casos podem acontecer:

- A árvore CSG está vazia = Gerar o nodo da Octree como Vazio
- A árvore CSG não está vazia mas contém mais de *p* primitivas = Gerar o nodo da Octree como Misto e processar seus descendentes
- A árvore CSG não está vazia mas contém *p* primitivas ou menos = Gerar o nodo da Octree como Cheio, guardando um ponteiro para esta árvore CSG reduzida

4. VISUALIZAÇÃO DA OCTREE-CSG

4.1. Resumo das propostas existentes

O problema de visualizar a estrutura híbrida entre Octree-CSG consiste em percorrer um raio por uma Octree, passando por nodos vazios até encontrar um nodo cheio. As propostas para este tipo de problema (conhecido na literatura como o problema de Pular Voxels Vazios) estão intimamente ligadas com a estrutura de dados utilizada para representar a Octree. Iremos discutir as soluções já conhecidas, apresentando uma nova solução : o algoritmo Bresenham Multi-Resolução.

Glassner usou uma tabela de hash para armazenar sua Octree. Como forma de identificar um nodo da árvore univocamente foi utilizado o caminho descrito desde a raiz até o próprio nodo (representação linear). Este representação linear (um número Octal) é usada como argumento na função de hash. Desta forma somente os nodos cheios são armazenados, resultando em grande economia de memória.

O problema chave no Algoritmo para pular Voxels Vazios consiste em encontrar o próximo nodo na Octree no caminho do raio. Este problema é resolvido por técnicas de Busca de Vizinho. Este problema ocorre porque nodos adjacentes no espaço da imagem podem não ter representação contígua na memória.

O método para busca de vizinho proposto por Glassner consiste em achar um ponto que seguramente esteja dentro do voxel vizinho procurado. Para tanto as seguintes etapas são executadas:

- Fazer testes de interseção do raio contra os 6 planos que definem o voxel corrente e descobrir o ponto e o plano por qual o raio está saindo
- Fazer um deslocamento do ponto descoberto paralamente ao plano descoberto de metade do menor lado do menor voxel da Octree, calculando um novo ponto
- Localizar a célula que contém este último ponto calculado. Esta célula será a célula vizinha

Wyvill e Kunii usam a representação em árvore para representar sua RCSG. A busca de vizinho e similar à proposta por Glassner, calculando-se um ponto que esteja seguramente dentro do vizinho.

Fujimoto possue o método mais rápido para descobrir um vizinho. Pelo fato de usar uma malha regular (Octree com todos filhos) muitas simplificações são feitas. Entretanto o uso de memória é extremamente abusivo, pois o número de nodos da Octree é muito grande. Todos estes são armazenados em uma tabela, cuja entrada é a representação linear do nodo.

Para descobrir o nodo vizinho é usado o método de translação de Octrees proposto por Gargantini [10]. Na verdade buscar um nodo vizinho nada mais é do que transladar um nodo segundo a direção do raio. Neste caso o algoritmo fica extremamente simplificado, visto que todos os nodos possuem o mesmo tamanho. Um cálculo é executado sobre a representação linear do nodo corrente, obtendo-se a representação linear do nodo vizinho. Com esta representação linear indexa-se diretamente a tabela que contém os nodos da Octree. Para descobrir sucessivamente as direções pelas quais o raio está saindo da célula, foi utilizado um algoritmo de traçado de linhas DDA, que informa a discretização do raio no Espaço Octree.

Uma outra alternativa proposta para busca de vizinhos foi proposta por Samet [11]. O algoritmo consiste em percorrer a Octree para encontrar o vizinho (subindo e descendo na árvore). Este algoritmo percorre no pior caso duas vezes a altura da Octree, e no melhor caso (quando o vizinho é um irmão

direto do nodo) somente um acesso é necessário.

4.2. O algoritmo Bresenham Multi-Resolução

Após a análise destas diversas propostas, buscou-se uma proposta nova, que conjugasse a eficiência do método do Fujimoto com a economia de memória encontrada nas propostas de Glassner e Wyvill. Para tanto seria necessário que pudéssemos percorrer a Octree com um algoritmo incremental, de preferência com aritmética inteira.

O algoritmo Bresenham foi estudado e notou-se que sua utilização para percorrer uma Octree seria inteiramente viável. O problema pode ser pensado totalmente no plano, visto que a extensão deste algoritmo para três dimensões (Bresenham 3D) é costumeiramente implementado usando-se dois algoritmos Bresenham 2D.

O algoritmo Bresenham nada mais é do que um processo de discretização de um segmento de reta em um dispositivo matricial. Como estes dispositivos possuem todas as células de mesmo tamanho foi possível criar um algoritmo incremental e de aritmética inteira. Se estes dispositivos possuíssem regiões onde um maior número de células fosse encontrado, seria possível gerar uma aproximação melhor da reta nestas regiões. O que muda entre um Bresenham sobre uma Malha de n1 x m1 pixels e uma outra malha de n2 x m2 são suas variáveis internas (principalmente o erro acumulado).

Para o algoritmo Bresenham trabalhar com células de tamanho variável (Bresenham Multi-Resolução) é necessário modificar ou o valor do erro acumulado, ou o valor contra qual este erro é testado (valor limitante). Como as células possuem uma relação de tamanho múltiplas de 2, mudar de resolução significa simplesmente multiplicar ou dividir por dois.

Duas fases são importantes neste novo algoritmo:

- Detalhamento: Quando se está passando de uma célula de maior resolução para uma célula de menor resolução
- Integração: Quando se está passando de uma célula de menor resolução para uma célula de maior resolução.

A figura 7 mostra casos onde ocorrem Detalhamento e Integração

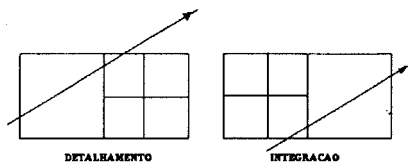

Figura 7 - Detalhamento e Integração

O algoritmo foi modificado basicamente em dois pontos: o erro acumulado não é mais medido em relação ao meio da célula mas em relação ao seu início. Outra modificaçao feita refere-se ao movimento em diagonal, que é desmenbrado em movimentos horizontais e verticais quando se fizer necessário (para permitir que todas as células no caminho do raio sejam percorridas). A estratégia de busca de vizinhos utilizadas foi àquela proposta por Samet [11]. Nesta estratégia o vizinho de um nodo é encontrado percorrendo a árvore ascendentemente e descendentemente.

O algoritmo de Bresenham Multi-Resolução possue a seguinte estrutura geral

```
PROCEDURE Bresenham Multi-Resolução (Raio)
BEGIN
  Entrada    = PontoEntradaQuadtree (Raio) ;
  Quadrante  = QuadranteEntradaQuadtree (Raio) ;
  Origem     = OrigemQuadrante (Quadrante) ;
  lado       = LadoQuadrante (Quadrante) ;
  a          = Entrada.X - Origem.X ;
  b          = Entrada.Y - Origem.Y ;
  slope      = CoeficienteAngular (Raio) ;
  erro       = slope * (lado - a) + b ;
  incremento = slope * lado ;
  WHILE (Valido (Quadrante)) DO
    IF (erro = lado) THEN
      MovimentoDiagonal (Quadrante)
    ELSE
      IF erro > lado THEN
        MovimentoVertical (Quadrante)
      ELSE
        MovimentoHorizontal (Quadrante)
END

PROCEDURE MovimentoHorizontal (Quadrante)
BEGIN
  q = VizinhoHorizontal (Quadrante) ;
  IF Nivel (q) < Nivel (Quadrante) THEN
    IntegraHorizontal
  ELSE
    IF TipoNodo (q) = MISTO THEN
      DetalhaHorizontal ;
  erro = erro + incremento ;
 END ;

PROCEDURE DetalhaHorizontal
BEGIN
  REPEAT
    lado := lado / 2 ;
    incremento := incremento / 2 ;
    IF erro >= lado THEN
      q = Filho (q, FilhodeCima) ;
      erro := erro - lado
    ELSE q = Filho (q, FilhodeBaixo) ;
  UNTIL TipoNodo (q) <> MISTO
END

PROCEDURE IntegraHorizontal
BEGIN
  FOR i = q.nivel to (Quadrante.nivel + 1) DO
    IF (Pai (Quadrante) = FilhodeCima) THEN erro = erro + lado ;
    lado := lado * 2 ;
    incremento := incremento * 2 ;
END
```

OBS: As rotinas de Movimento Vertical e Diagonal tem codificação semelhante

5. CONCLUSÃO

Neste trabalho foi apresentada uma proposta para otimização do algoritmo de exibição de cenas realistas "Ray Tracing". A proposta consiste em subdividir uma cena complexa (segundo um modelo CSG) a fim de reduzir o tempo gasto com testes de interseções.

A estrutura resultante desta subdivisão é uma estrutura híbrida entre Octree e CSG (semelhante a estrutura proposta por Wyvill e Kunii [9]). Para exibir a imagem foi criado uma variação do algoritmo Bresenham, adaptado para percorrer Octrees. Este novo algoritmo, chamado Bresenham Multi-Resolução, permite o acompahamento do Raio sobre uma Octree segundo um algoritmo incremental, tornando o processo bastante eficiente.

No presente momento temos o algoritmo canônico de "Ray Tracing" para sólidos CSG implementados. Os algoritmos de geração da estrutura Octree-CSG estão em fases de testes, sendo que o algoritmo Bresenham Multi-Dimensional já se encontra concluído.

BIBLIOGRAFIA

[1] Rogers, David F. "Procedural Elements for Computer Graphics," *McGraw Hill Book Company*, 1985.

[2] Appel, A. "Some techniques for Shading Machine Rendering of Solids," *AFIPS 1968 Sprint Joint Comput. Conf.* pp 37-45

[3] Whitted, J. T. "An Improved Illumination Model for Shaded Display," *CACM, Vol 23*, pp 343-349 (Proc SIGGRAPH 79)

[4] Requicha, A.A.G. "Representation for rigid solids: Theory, methods and systems. *ACM Computing Surveys, 12, 4*, Dezembro 1980

[5] Tanimoto, S.L e Jackins, C.L. "Oct-trees and Their use in Representing Three Dimensional Objects", *Computer Graphics and Image Processing, Vol 14, No 3* , pp 249-270, Novembro 1980

[6] Roth, S. D. "Ray Casting for Modeling Solids", *Computer Graphics and Image Processing, Vol 18* , pp 109-144, 1982

[7] Glassner, A. S. "Space Subdivision for Fast Ray Tracing" *IEEE CG&A*, Outubro 1984

[8] Fujimoto, A. et all. "ARTS: Accelerated Ray-Tracing System *IEEE CG&A*, Abril 1986

[9] Wyvill, G.,Kunii, T. L. e Shirai, Y. "Space Division for Ray Tracing in CSG" *IEEE CG&A* , Abril 1986

[10] Gargantini, I. "Linear Octrees for Fast Processing of Three Dimensional Objects" *Computer Graphics and Image Processing*, pp 365-374, dezembro 1982

[11] Samet, H. "Neighbor Finding Techniques for Images Represented by Quadtrees" *Computer Graphics and Image Processing, 18*, pp 37-57, 1982.

Este trabalho foi parcialmente financiado pela CAPES, CNPQ, MCT, FINEP e FIPEC.