UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANZ JOSEF FIGUEROA FERREIRA DA SILVA

Síntese de Fenômenos Naturais através do Traçado de Raios usando "Height Fields"

Dissertação submetida como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Anatólio Laschuk
Orientador

Porto Alegre, janeiro de 1996

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Figueroa, Franz Josef Ferreira da Silva

Síntese de Fenômenos Naturais através do Traçado de Raios usando "Height Fields". – Porto Alegre: CPGCC da UFRGS, 1995.

111p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1996. Orientador: Laschuk, Anatólio.

Dissertação: Computação Gráfica.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Dr. Helgio Casses Trindade
Pró-Reitor de Pesquisa e Pós-Graduação: Prof. Dr. Cláudio Scherer
Diretor do Instituto de Informática: Prof. Dr. Roberto Tom Price
Coordenador do CPGCC: Prof. Dr. José Palazzo Oliveira
Bibliotecária-chefe: Zita Oliveira

"Porque o Senhor dá a sabedoria:

da sua boca vem o conhecimento e o entendimento.

Ele reserva a verdadeira sabedoria para os retos.

Porquanto a sabedoria entrará no teu coração,

e o conhecimento será suave à tua alma."

Provérbios (2:6,7,10)

AGRADECIMENTOS

Quero em primeiro lugar agradecer ao Senhor Jesus Cristo, meu senhor e salvador, quem através do Seu Santo Espírito tem sido constante apoio, incentivo, exortação e consolo. A Ele devo tudo quanto sou, possuo, conheço e faço. Jesus, és a razão da minha vida.

Um agradecimento especial ao Prof. Laschuk, incansável na sua orientação, não apenas como mestre, mas também como amigo, sem cuja ajuda e dedicação este trabalho não teria sido possível.

A minha amada e fiel esposa Eliana Figueiró Figueroa, companheira e auxiliadora nas horas mais escuras da minha jornada. Amo-te.

Aos meus pais Jaime Figueroa e Maria Tereza de Figueroa pelo amor, carinho, lições de vida e por me mostrar como ser realmente um homem de Deus. Aos meus amados irmãos Maria Helena e Günther por saber que sempre e sob qualquer circunstância conto com o amor deles.

Ao meu querido Tio Olavo por ser um amigo em todas as horas e pelo apoio, carinho e amizade incondicional que sempre, inclusive nas madrugadas, tem me prestado.

Aos amigos da Internet pelo seu apoio e sugestões na elaboração deste trabalho: Juan Carlos Arévalo, Douglas Muir, Chris Young, Drew Wells, The Pov-Team e tantos outros.

Ao Curso de Pós-Graduação em Ciência da Computação e ao Instituto de Informática, pelo conhecimento, estrutura e recursos disponíveis. A todos os funcionários que contribuíram fazendo da UFRGS um ambiente agradável de estar.

Aos meus sogros Amaro Figueiró e Anna Maria Figueiró por serem como pais para mim. Aos meus irmãos em Cristo: Walter, Andreia, Isaias, Guilherme, e tantos outros pelas orações.

Aos meus alunos por serem um exemplo de que a felicidade esta em compartilhar o que se possui. Em especial ao Mauro (Mr. Milk) pela amizade e discussões sobre CG.

Finalmente, quero agradecer a CAPES, CNPQ, e todos os contribuintes deste maravilhoso Brasil por financiarem e tornarem possível este trabalho.

Que Deus os abençoe.

SUMÁRIO

LISTA DE FIGURAS **9**

LISTA DE ABREVIATURAS **11**

RESUMO **12**

ABSTRACT **13**

1 INTRODUÇÃO **14**

1.1 A Síntese de Fenômenos Naturais **14**

2 SÍNTESE DE FENÔMENOS NATURAIS **16**

2.1 Introdução **16**

2.2 Síntese de Montanhas e Terrenos. **17**

2.2.1 Geometria Fractal 17

2.2.2 Superfícies Quádricas Texturizadas 19

2.3 Síntese de Água. **19**

2.3.1 Mapeamento de Texturas 19

2.3.2 Modelos Procedurais 20

2.3.2 Sistema de Partículas 21

2.4 Síntese de Madeira **21**

2.4.1 Mapeamento Bidimensional 22

2.4.2 Mapeamento Tridimensional 22

2.5 Síntese de Fogo **23**

2.5.1 Mapeamento de Texturas 23

2.5.2 Sistema de Partículas 24

2.5.3 Modelos Baseados em Física 25

2.6 Conclusão **27**

3 SÍNTESE DE IMAGENS POR TRAÇADO DE RAIOS **28**

3.1 Introdução **28**

3.2 O Algoritmo de Traçado de Raios **29**

3.3 Um Modelo de Iluminação **30**

3.3.1 Componente de Luz Ambiente 30

3.3.2 Componente de Reflexão Difusa 31

3.3.3 Componente de Reflexão Especular 32

3.3.4 Reflexões 32

3.3.5 Transparência 33

3.3.6 Sombras 33
3.4 Desempenhos do Algoritmo 34
3.5 Conclusão 34

4 PROPOSTA DE MODELO DE TRAÇADO DE RAIOS DE FENÔMENOS NATURAIS MODELADOS POR CAMPOS DE ALTITUDES 35
4.1 Introdução 35
4.2 Campos de Altitude 35
4.2.1 Modelo de Imagens Utilizado 37
4.2.2 Seleção de Imagens como Campos de Altitude 37
4.2.2.1 Campos de Altitude Reais 37
4.2.2.2 Campos de Altitude Virtuais 39
4.3 Traçados de Raios de Campos de Altitude 40
4.3.1 Funcionamento do Algoritmo 41
4.3.1.1 Intersecção dos Raios com o Campo de Altitude 42
4.3.1.2 Determinação da Superfície 44
4.3.1.3 Nível do Mar. 46
4.3.1.4 Implementação Modular 47
4.3.2 Colorização do Campo de Altitude 57
4.4 Conclusão 59

5 PROTÓTIPO IMPLEMENTADO 60
5.1 Introdução 60
5.2 Módulos do Protótipo 60
5.3 Fluxo de Execução 63
5.4 Entrada de Dados 64
5.5 Saída de Dados 67
5.6 Conclusão 67

6 APRESENTAÇÃO E ANÁLISE DE RESULTADOS 68
6.1 Introdução 68
6.2 Síntese de Fenômenos da Natureza com Campos de Altitude 69
6.2.1 Síntese de Montanhas 69
6.2.1.1 Campos de Altitude Reais 69
6.2.1.2 Campos de Altitude Virtuais 74
6.2.2 Síntese de Água 79
6.2.3 Síntese de Madeira 81
6.2.4 Síntese de Fogo 82

6.3 Complexidade do Algoritmo 85
6.4 Considerações Importantes 87
6.4.1 Uso de Dados Bidimensionais 87
6.4.2 Pré-processamento Implícito 88
6.4.3 Eficiência na Síntese 88
6.4.4 Baixo Consumo de Recursos 89
6.4 Conclusão 90
7 CONCLUSÕES 91
7.1 Futuro 93
ANEXO A-1 Listagens 95
BIBLIOGRAFIA 105

LISTA DE FIGURAS

Figura 2.1 - Perfil de montanha criado através de fractais 17
Figura 2.2 - Síntese de montanha usando fractais 18
Figura 2.3 - Montanha gerada por subdivisão de triângulos incluindo parâmetro estocástico 18
Figura 2.4 - Imagem simulando o mar através do mapeamento de texturas 20
Figura 2.5 - Bloco de madeira gerado através de procedimentos matemáticos 23
Figura 2.6 - Textura de fogo.....24
Figura 2.7 - Textura de fogo após a aplicação de uma função de turbulência 24
Figura 2.8 - Imagem produzida para o filme "A Ira de Khan" através de sistema de partículas 25
Figura 2.9 - Simulação de fogo através de sistema de partículas (filme "A Ira de Khan") 25
Figura 2.10 - Síntese de chamas levando em consideração princípios físicos 26
Figura 3.1 - Princípio básico de "Ray Tracing" 30
Figura 3.2 - Vetores existente num sistema de traçado de raios.....31
Figura 4.1 - Imagem de satélite de uma região montanhosa 36
Figura 4.2 - Campo de altitude obtido de um arquivo DEM 38
Figura 4.3 - Imagem produzida no sistema de processamento de imagens Adobe Photoshop e que pode ser utilizada como campo de altitude.....40
Figura 4.4 - Raio atravessando um campo de altitude.....42
Figura 4.5 - Divisão de uma imagem num conjunto de triângulos.....45
Figura 4.6 - Superfície definida por um campo de altitude. 45
Figura 4.7 - Testes de intersecção de um raio com um paralelepípedo 48
Figura 4.8 - Identificação das células de entrada e saída de um raio num campo de altitude.....51
Figura 4.9 - Células marcadas pelo algoritmo DDA padrão. 51
Figura 4.10 - Células determinadas pelo algoritmo DDA modificado.....52
Figura 4.11 - Parte de uma imagem e a sua semelhança nas cores produzidas na superfície do campo de altitude. 58
Figura 5.1 - Arquitetura do protótipo implementado. 61
Figura 6.1 - "Crater Lake", Oregon, Estados Unidos 70
Figura 6.2 - Visão tridimensional do "Crater Lake".....71
Figura 6.3 - Pikes Peak, Colorado 72

Figura 6.4 - "Pikes Peak" sintetizado sem levar em consideração as cores da "palette".....72
Figura 6.5 - "Pikes Peak" sintetizado considerando-se as cores na tabela de "palette".....73
Figura 6.6 - Síntese da região leste do "Grand Canyon", Estados Unidos, usando campos de altitude. 74
Figura 6.7 - Campo de altitude gerado por rotina fractal de plasma no sistema Adobe Photoshop. 75
Figura 6.8 - Imagem sintetizada de uma montanha junto ao mar.....76
Figura 6.9 - Campo de altitude criado no sistema de processamento imagens Adobe Photoshop. 77
Figura 6.10 - Cenário de região montanhosa sintetizado utilizando campos de altitude virtuais.....77
Figura 6.11 - Cenário sintetizado através do método proposto com imagem de fundo (nuvens) acrescentada. 78
Figura 6.12 - Síntese de ilha no mar usando campos de altitude virtuais.....79
Figura 6.13 - Síntese de água (cena inicial de uma animação).....80
Figura 6.14 - Síntese de água.....80
Figura 6.15 - Campos de altitudes para a animação de pingos caindo num recipiente com água. 81
Figura 6.16 - Síntese de madeira quebrada. 82
Figura 6.17 - Imagem bidimensional simulando fogo.....83
Figura 6.18 - Síntese de fogo utilizando-se campos de altitude. 84
Figura 6.19 - Exemplo de síntese de fogo. 84

LISTA DE ABREVIATURAS

3DDDA	- *Three-Dimensional Digital Differential Analyser*
ASCII	- *American Standart Code for Information Interchange*
CAD	- *Computer Aided Design*
DDA	- *Digital Differential Analyzer*
DEM	- *Digital Elevation Map*
GIF	- *Graphics Interchange Format*
POV	- *Persistence of Vision*
TIFF	- *Tagged Image File*
UFRGS	- Universidade Federal do Rio Grande do Sul
USGS	- *United States Geological Survey*

RESUMO

A síntese de imagens é uma ferramenta valiosa na compreensão de diversos fenômenos da natureza. Nos últimos anos várias abordagens têm sido propostas para sintetizar tais fenômenos. A grande maioria de tais abordagens têm se centralizado no desenvolvimento de modelos procedurais. Porém, cada uma destas técnicas simula exclusivamente um fenômeno natural.

Um dos métodos de síntese de imagens fotorealísticas mais proeminente é denominado de *Traçado de Raios* (*Ray Tracing)*. Contudo, apesar de produzir imagens de excelente qualidade, este método é computacionalmente muito oneroso. A síntese de fenômenos naturais utilizando-se o traçado de raios é um desafio. É importante que este problema seja abordado, apesar da sua complexidade, pois a simulaçao fotorealista da natureza é muito importante para os cientistas e pesquisadores desde o surgimento dos computadores.

Um algoritmo versátil e rápido para a síntese de fenômenos da natureza através do traçado de raios utilizando campos de altitude é proposto. O algoritmo utiliza uma modificação do algoritmo do Analisador Diferencial Digital de Bresenham para atravesar uma matriz bidimensional de valores de altitude. A determinação das primitivas geométricas a serem interseccionadas por um raio é obtida num tempo $O(\sqrt{N})$, sendo *N* o número de altitudes no campo de altitude.

Este trabalho faz uma comparação em termos de velocidade e realismo deste método com outras abordagens convencionais; e discute as implicações que a implementação deste método traz.

Finalmente, destaca-se a simplicidade e versatilidade que este método proporciona devido à pequena quantidade de parâmetros necessária para a síntese de fenômenos naturais utilizando o traçado de raios. Para a criação de animações basta a especificação de novos parâmetros num intervalo de tempo diferente.

Palavras-chave: Traçado de raios, fenômenos naturais, DDA, campos de altitude.

ABSTRACT

Title: "NATURAL PHENOMENA SYNTHESIS THROUGH RAY TRACING USING HEIGHT FIELDS"

Visualization is a powerful tool for better undestanding of several natural phenomena. In recent years, several techniques have been proposed. Considerable interest in natural scene synthesis has focused on procedural models. However, these techniques produce synthetic scenes of only one natural phenomenon.

Ray tracing is one of the most photorealistic methods of image syntesis. While providing images of excellent quality, ray tracing is a computationally intensive task. Natural scene synthesis is a challenging problem within the realm of ray tracing. It is important to tackle this problem, despite of its complexity, because photorealistic simulation have been important to scientific community since the appearance of computers.

A fast and versatile algorithm for ray tracing natural scenes through height fields is presented. The algorithm employs a modified Bresenham DDA to traverse a two dimensional array of values. The objects tested for intersection are located in $O(\sqrt{N})$ time where N is the number of values in the field.

This work compares the speed-up and photorealism achieved in natural scene synthesis using this method with other algorithms and discusses the implications of implementing this approach.

As a final point, the simplicity and versatility of synthesizing complex natural scenes from a few parameters and data is especially attractive. Animated sequences require only the additional specifications of time modified parameters or data.

Keywords: Ray tracing, natural phenomena, DDA, height field.

1 INTRODUÇÃO

"Adquire a sabedoria, adquire a inteligência...
a sabedoria é a coisa principal:
adquire pois a sabedoria; sim, com tudo
o que possuis adquire o conhecimento.
Exalta-a, e ela te exaltará;
e, abraçando-a tu, ela te honrará"
(Provérbios 4:5,7,8)

1.1 A Síntese de Fenômenos Naturais

A síntese de imagens tem se tornado muito popular na sociedade atual, sendo utilizada no cinema (REEVES 1985) (FIGUEROA 1992), nos simuladores de vôo, no planejamento arquitetônico (ERVIN 1993), no entretenimento e na representacão gráfica de entidades matemáticas. Na procura do realismo fotográfico, a arte da síntese de imagens tem progredido desde o simples desenho de linhas até complexos cenários aeroespaciais. Contudo, a síntese de fenômenos naturais é um passo a mais na busca do realismo fotográfico.

A motivação para o estudo da síntese de fenômenos naturais deriva-se de duas propriedades que estes possuem: complexidade e beleza. A complexidade deve-se à enorme quantidade de detalhes gerados espontaneamente pela natureza. Pequenos objetos, tais como um floco de neve, podem conter milhares de pequenas partes que formam o todo. Por outro lado, objetos maiores possuem também grande complexidade, como por exemplo, a superfície de uma montanha que é o resultado de complexos processos geológicos.

Finalmente, a beleza dos fenômenos da natureza é um estímulo para seu estudo. Pela compreensão de como trabalha a natureza, um animador terá em mãos as ferramentas necessárias para criar imagens fotorealísticas com variações e usos praticamente ilimitados. Poderá assim, por exemplo, ser estudada a influência dos ventos num incêndio florestal ou ainda visualizar um produto não fabricado como se tivesse sido produzido com materia prima natural (madeira, mármore, etc).

Este trabalho visa proporcionar uma ferramenta computacional para a síntese de fenômenos naturais utilizando uma das técnicas de maior realismo fotográfico denominada de Traçado de Raios.

No capítulo 2 são apresentadas as principais abordagens utilizadas atualmente pela Computação Gráfica para a síntese de diversos fenômenos da natureza.

No capítulo 3, apresentam-se os conceitos necessários sobre os sistemas de traçado de raios, com o intuito de criar o contexto onde será desenvolvido este trabalho.

No capítulo 4, o modelo desenvolvido é apresentado em detalhes. As diferentes etapas de implementação assim como as características do mesmo são descritas neste capítulo também.

No capítulo 5, apresenta-se detalhes de implementação do modelo proposto.

No capítulo 6, apresenta-se uma análise dos resultados obtidos com o modelo, levando-se em consideração tanto o aspecto visual (fotorealismo) como o seu custo computacional.

Nos capítulos 7 e 8, são apresentadas, respectivamente, as conclusões e as bibliografias consultadas.

2 SÍNTESE DE FENÔMENOS NATURAIS

"E chamou Deus à porção seca Terra;
e ao ajuntamento das águas chamou Mar.
E viu Deus que era bom."
(Gênesis 1:10)

2.1 Introdução

A modelagem, visualização e síntese de objetos regulares (edificios, peças mecânicas, etc) podem ser facilmente simuladas através de recursos computacionais. Os simuladores de vôo são umas das aplicações mais antigas da Computação Gráfica, e a maioria dos sistemas de CAD (*Computer Aided Design*) são utilizados para modelar equipamentos. Contudo, os especialistas em Computação Gráfica têm sido sempre cativados pela possibilidade de criarem cenas tridimensionais envolvendo fenômenos naturais. Estes fenômenos da natureza são sem dúvida alguma os objetos mais difícies de serem sintetizados através de processos computacionias devido à sua irregularidade e constante modificação. O custo computacional pode tornar-se excessivamente alto se forem levadas em consideração as leis físicas e matemáticas destes fenômenos.

Os modelos e técnicas utilizados na síntese de tais fenômenos são na sua grande maioria modelos procedurais e foram desenvolvidos para simular especificamente um determinado fenômeno. Este capítulo apresenta de forma sucinta as principais abordagens utilizadas na representação de vários fenômenos naturais tais como: montanhas, água, madeira e fogo.

2.2 Síntese de Montanhas e Terrenos.

2.2.1 Geometria Fractal

A geometria fractal de Benoit Mandelbrot tem sido amplamante utilizada na síntese de montanhas (MANDELBROT 1982) (PEITGEN 1988), por prover não

apenas uma representação realística das mesmas mas principalmente por fornecer um procedimento matemático capaz de modelar figuras complexas como montanhas, terrenos, crateras, etc. Estes modelos baseiam-se na repetição sucessiva e reduzida da mesma figura onde é acrescido um parâmetro estocástico para criar irregularidades na mesma.

A forma mais simplista de simulação de montanhas através de fractais consiste na subdivisão de triângulos ou quadriláteros. Um processo recursivo de subdivisão é aplicado a cada face, até um deterninado nível de refinamento, resultando assim num modelo que simula uma região montanhosa. A figura 2.1 mostra o perfil de uma provável montanha através de subdivisões sucessivas.

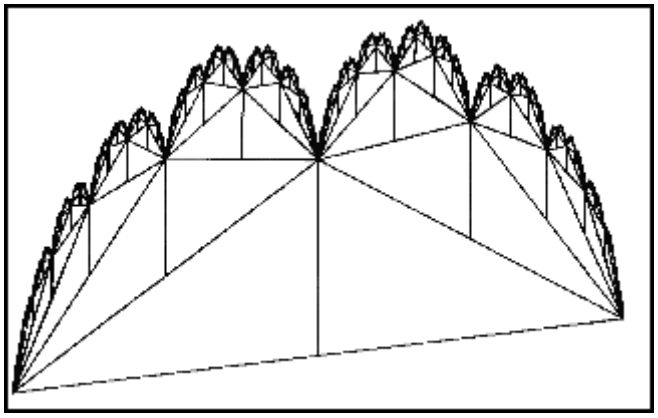


Figura 2.1 - Perfil de montanha criado através de fractais (PEITGEN 1988)

Este processo pode ser estendido para um espaço tridimensional conforme ilustra a figura 2.2.

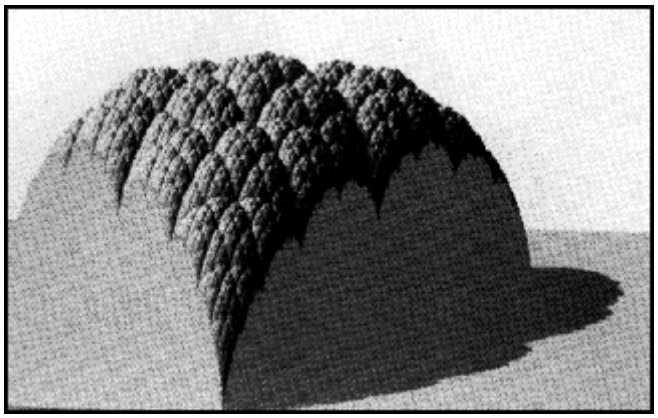


Figura 2.2 - Síntese de montanha usando fractais (PEITGEN 1988)

Contudo, a figura 2.2 não condiz com a representação característica de uma montanha por apresentar um padrão repetido ao longo da superfície. Por este motivo é inserido um fator estocástico que visa minimizar a repetição do padrão e criar irregularidades existentes nas montanhas reais. A figura 2.3 mostra o resultado deste processo.

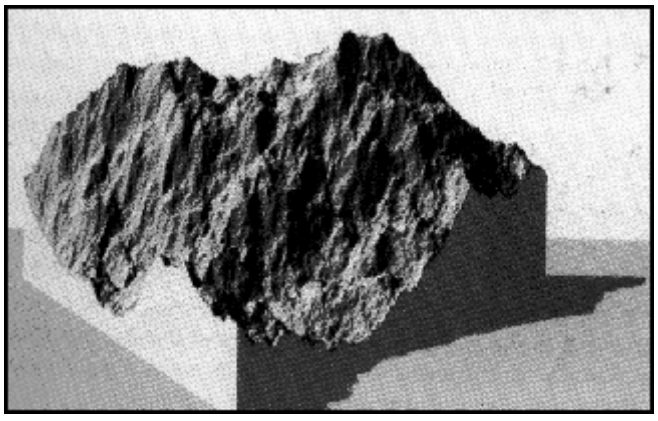


Figura 2.3 - Montanha gerada por subdivisão de triângulos incluindo parâmetro estocástico (PEITGEN 1988)

O nível de realismo das imagens é diretamente proporcional ao número de faces criadas pela dimensão fractal utilizada, podendo chegar a centenas e até milhares de primitivas geométricas, o que traz como conseqüência um enorme consumo de memória assim como um tempo de síntese muito alto (independente do algoritmo utilizado).

2.2.2 Superfícies Quádricas Texturizadas

O realismo obtido em imagens de montanhas produzidas através de fractais é atingido através da síntese de centenas e até milhares de triângulos, o que é, em extremo, oneroso em termos de tempo de processamento. Por outro lado, limitar o número de triângulos reduz o realismo da imagem gerada. Para evitar estes problemas, Gardener (GARDENER 1984) propos outro método para a síntese de terrenos. Neste método as imagens são obtidas através da síntese de superfícies curvas e o uso de texturas.

As cenas são compostas por um conjunto de objetos convexos, cada um destes definido por uma superfície quádrica e *N* planos de envelope, podendo *N* ser igual a zero. O principio básico consiste na determinação das partes das superfícies curvas que estarão visíveis na imagem final. Para se obter isto são projetadas todas as superfícies curvas em cada um dos planos que as envolvem, criando-se assim uma lista de segmentos visíveis para cada linha de varredura do vídeo (*scan-line*). Para a representação de uma superfície natural são mapeadas texturas sobre as superfícies.

Embora este método minimize em muito o custo computacional da síntese de montanhas, limita-se à síntese de cenários curvos, tais como dunas ou pradeiras.

2.3 Síntese de Água.

Cenas naturais em muitos casos envolvem regiões com água; contudo a representação de água através de meios computacionais é uma tarefa de extrema complexidade e dificuldade. A principal razão é que a água está constantemente em movimento e por isso a sua forma depende do seu movimento o que torna difícil a modelagem da mesma pois as leis e os princípios hidrodinânicos são muito complexos para serem utilizados.

2.3.1 Mapeamento de Texturas

Diversas abordagens têm sido propostas para simular ondas na água. Um dos primeiros a simular água foi Whitted (WHITTED 1980) na criação da animação intitulada *The Compleat Angler*. Utilizando um sistema de traçado de raios ele animou realisticamente ondas numa pequena piscina. As ondas foram criadas atraves de *Bump Mapping*, um tipo de mapeamento de texturas, onde o vetor normal à superfície sofria perturbações de acordo com uma função sinusoidal.

A figura 2.4 mostra uma imagem simulando o mar, obtida através de *bump-mapping*, neste método o vetor mormal a superfície (um plano neste caso) é ligeiramente modificado a fim de produzir perturbações na superfície.

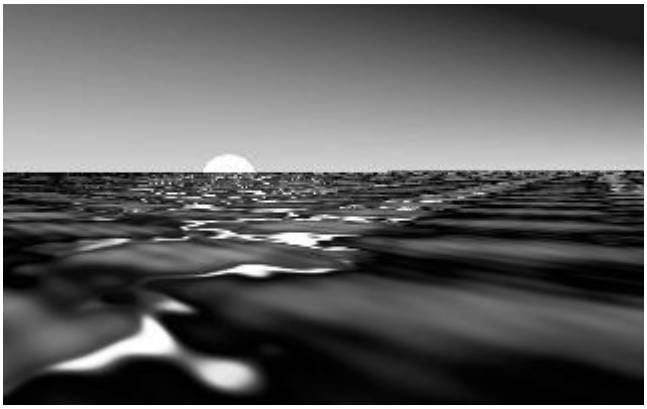


Figura 2.4 - Imagem simulando o mar através do mapeamento de texturas

2.3.2 Modelos Procedurais

Diversos modelos para a síntese de ondas e a arrebentação das mesmas foram propostos por Peachey (PEACHEY 1986) e Fournier (FOURNIER 1986) durante o ano de 1986. O modelo mais interessante (FOURNIER 1986) considerava partículas em movimento numa superfície, definindo o plano XY na horizontal e o eixo Z na vertical. Cada partícula descrevia um movimento circular ao redor do seu ponto de partida P_0. Para a definição do movimento das mesmas foram utilizadas equações senoidais.

Através da variação de parâmetros temporais e alguns estocásticos, as partículas descreviam o movimento característico das ondas do mar, permitindo inclusive a criação de animações.

2.3.2 Sistema de Partículas

Um modelo sistemático para a síntese de água (e outros fenômenos) desenvolvido por Reeves (REEVES 1983) é denominado de *sistema de partículas*. Um sistema de partículas é uma coleção de partículas que juntas representam um objeto de aparência não muito definida. Ao longo de determinado período de tempo as partículas nascem, movem-se, modificam-se e morrem.

Reeves descreveu de forma simples o cálculo de cada cena de uma seqüência animada utilizando-se um sistema de partículas, conforme descrito a seguir:

1. Geração das novas partículas através de processos controlados ou estocásticos e determinação de atributos individuas (cor, tempo de vida, posição),
2. Extinção das partículas cujo tempo de vida foi esgotado,
3. Movimentação e transformação das partículas restantes de acordo com seus atributos (dinâmicos, hidrodinâmicos, etc) e
4. Síntese das partículas "vivas" no ambiente.

Sobre as partículas podem atuar parâmetros que levem em consideração aspectos físicos dos fluídos (MILLER 1989) como também as relações provenientes da colisão das mesmas, conforme demostrou Sims (SIMS 1990) na sua simulação realística de uma cachoeira e do choque d'água com as pedras.

2.4 Síntese de Madeira

Para a síntese de madeira tem sido utilizado o mapeamento de texturas, tanto bidimensional como tridimensionalmente.

2.4.1 Mapeamento Bidimensional

O mapeamento bidimensional de texturas consiste na transformação de uma imagem bidimensional para uma superfície tridimensional (ROGERS 1985) (WATT 1989). Este processo é bastante conhecido pela comunidade de Computação Gráfica. Para este fim é necessária a existência de uma imagem de um pedaço de madeira digitalizada da vida real.

O principal problema que este método apresenta é a dificuldade de se mapear uma imagem de duas dimensões para um objeto tridimensional cuja forma não é nada regular, ou contém centenas de faces.

2.4.2 Mapeamento Tridimensional

Neste procedimento são considerados dois espaços "paralelos", onde o primeiro contém a cena a ser sintetizada e o segundo é composto por um bloco sólido de uma textura definida proceduralmente (WATT 1989). Assim, qualquer ponto $P_o(X_{o,}Y_o,Z_o)$ na cena terá um ponto $P_t(X_t,Y_t,Z_t)$ equivalente no bloco de textura.

Desta forma a madeira pode ser síntetizada através de um conjunto de cilindros concêntricos, onde um dos seus eixos é perturbado por uma função harmônica. Uma descrição detalhada das equações utilizadas para perturbar os cilindros assim como uma rotina exemplo em "alto nível" podem ser obtidas em Watt (WATT 1989). A figura 2.5 mostra um bloco de madeira gerado através de procedimentos matemáticos.

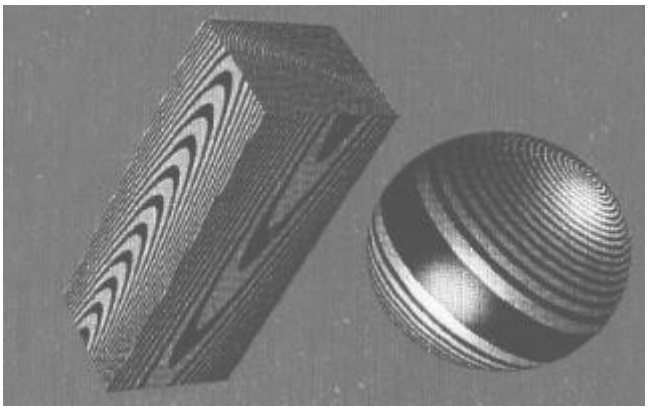


Figura 2.5 - Bloco de madeira gerado através de procedimentos matemáticos (FOLEY 1990)

Estes procedimentos determinam a coloração e aparência de madeira em qualquer objeto, mas não resolvem o problema da definição da forma (modelagem) do objeto.

2.5 Síntese de Fogo

Para a síntese de fogo são empregados básicamente os mesmos procedimentos utilizados na síntese de água.

2.5.1 Mapeamento de Texturas

Este procedimento foi proposto por Perlin (PERLIN 1985) como uma derivação do uso de uma função de turbulência sobre determinada textura ou cor. A figura 2.6 mostra uma textura de fogo no seu estado original e a figura 2.7 mostra a mesma imagem após a aplicação da função de turbulência.

Figura 2.6 - Textura de fogo (WATT 1992)

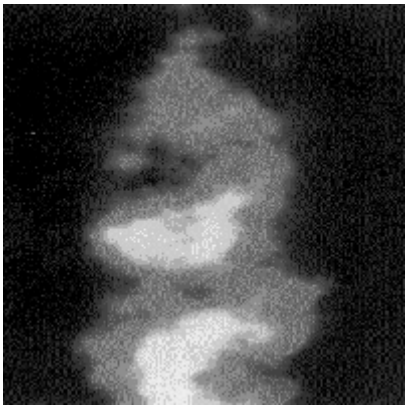


Figura 2.7 - Textura de fogo após a aplicação de uma função de turbulência (WATT 1992)

Perlin mostrou que outras fenômenos da natureza podem ser simulados desta forma (mármore, etc) como também é possível variar os parâmetros de turbulência num intervalo de tempo a fim de permitir a criação de seqüências de imagens (animação).

2.5.2 Sistema de Partículas

Os sistemas de partículas têm sido utilizados com muito sucesso na simulação de fogo, sendo inclusive responsáveis pela seqüência do *Gênesis* do filme *Star Trek II: The Wrath of Khan* (Jornada nas Estralas II: A Ira de Khan) (REEVES 1983). A seqüência simulava a explosão de um planeta, através de partículas que partiam de um ponto na superfície de uma esfera e se expandiam em círculos

concêntricos sobre a superfície. Para tal animação foram utlizadas 750.000 partículas (THALMANN 1987). As figura 2.8 e 2.9 mostram partes das seqüencias produzidas para o filme utilizando-se sistemas de partículas.

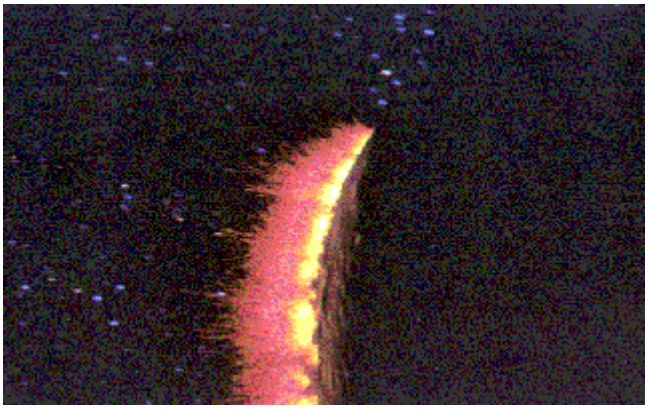


Figura 2.8 - Imagem produzida para o filme "*A Ira de Khan*" através de sistema de partículas (REEVES 1983)

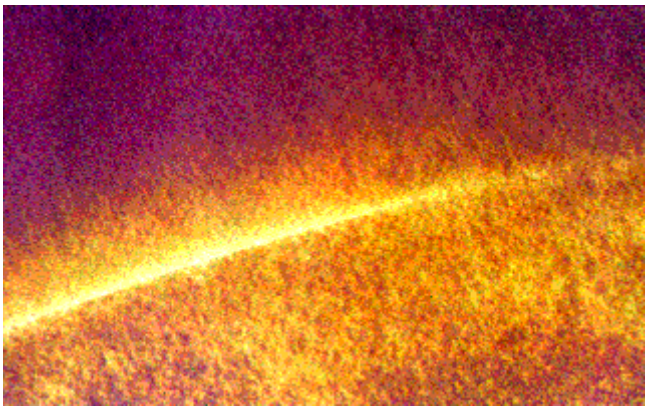


Figura 2.9 - Simulação de fogo através de sistema de partículas (filme "*A Ira de Khan*") (REEVES 1983)

2.5.3 Modelos Baseados em Física

O processo de combustão é bastante complicado, pois envolve leis físicas e químicas. Estes aspectos têm despertado recentemente o interesse de diversos pesquisadores, conforme mostra a síntese de chamas realizada por Inakage (INAKAGE 1990).

Inakage propõe modelar chamas laminares através de elipsóides, e chamas turbulentas adicionando apenas uma função estocástica para perturbar a geometria básica do objeto. A cor das chamas é determinada pelo tipo de combustível e é dependente também da região no interior da chama. O trabalho realizado leva em consideração os seguintes aspectos:

- presença de atmosfera (chamas laminares e turbulentas),
- representação volumétrica (consideração do interior do objeto) e
- reações químicas (combustível, temperatura, velocidade de combustão, etc).

Para a síntese de tais imagens foi utilizado um sistema de traçado de raios aprimorado para levar em consideração não apenas a superfície dos objetos mas também o interior dos mesmos, que Inakage denomina de *volume tracing*. A figura 2.10 mostra alguns resultados obtidos, mais especificamente, duas velas com chamas laminares e uma vela com a chama turbulenta.

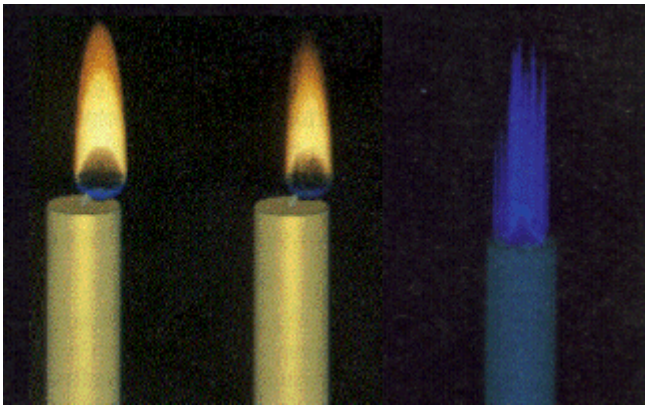


Figura 2.10 - Síntese de chamas levando em consideração princípios físicos (INAKAGE 1990)

Este tipo de síntese permite facilmente a geração de animações através da variação dos parâmetros físicos e químicos envolvidos.

2.6 Conclusão

Este capítulo teve por objetivo descrever os principais procedimentos utilizados atualmente na síntese de fenômenos naturais através de recursos computacionais e desta forma descrever o contexto em que este trabalho está inserido. É importante salientar que diversos aspectos dos fenômenos naturais não são totalmente compreendidos pelo ser humano na atualidade, e a sua simulação visa justamente aprimorar o seu conhecimento.

3 SÍNTESE DE IMAGENS POR TRAÇADO DE RAIOS

"E disse Deus: Haja luz. E houve luz.
E viu Deus que era boa a luz"
(Gênesis 1:3)

3.1 Introdução

Um dos objetivos mais importantes da Computação Gráfica é a geração de imagens que pareçam realistas quando mostradas no vídeo de um computador. Imagens realistas são amplamente utilizadas em diversas áreas tais como medicina, arquitetura, simulação, visualização, educação e publicidade.

As abordagens iniciais utilizadas para a síntese de imagens realisticas num computador foram a remoção de linhas ocultas e o sombreamento de superfícies sem levar em consideração o efeito dos objetos no ambiente. Contudo, para obter imagens mais realistas e detalhadas, um sombreamento global deve ser realizado.

O primeiro método introduzido para gerar imagens verdadeiramente realistas é conhecido como *Traçado de Raios* (*Ray Tracing*) (WHITTED 1980) e inclui efeitos como transparência, sombras e reflexão. Este método tem produzido imagens com um alto grau de realismo praticamente impossíveis de serem obtidas através de outras abordagens.

Este capítulo visa caracterizar o funcionamento de um sistema de traçado de raios. Serão descritos os seguintes conceitos com o objetivo de proporcionar o entendimento do tema: luz ambiente, componente difusa, componente especular, reflexão, transparência e sombras. Após estas definições serão apresentadas algumas considerações sobre o desempenho do algoritmo.

3.2 O Algoritmo de Traçado de Raios

Num modelo simplificado do algoritmo de traçado de raios, um raio é disparado para cada *pixel* do vídeo a partir do observador, num espaço tridimensional, como mostra a figura 3.1. Cada objeto da cena é testado para se determinar o primeiro ponto de alguma superfície atingida por este raio. A intensidade da cor no ponto de intersecção é calculada, o raio é então refletido a partir do ponto da superfície atingida para verificar se ele toca num outro objeto, ou numa fonte de luz. Se o raio refletido terminar em alguma fonte de luz, então um reflexo é visto na superfície. Se o raio refletido atinge a superfície de outro objeto, a intensidade da cor no novo ponto de intersecção é também levada em consideração. Isto produz o reflexo de uma superfície na outra. Quando um objeto é transparente, o raio é dividido em dois componentes, e cada novo raio é tratado individualmente. Conforme explicado acima, o valor da cor no ponto de intersecção determina o valor da cor do *pixel* associado a um raio. Desta forma, uma vez encontrado o ponto de intersecção, a cor deve ser calculada de acordo com o modelo de iluminação. A seguir será apresentado um modelo de iluminação bastante simples e que produz excelentes resultados.

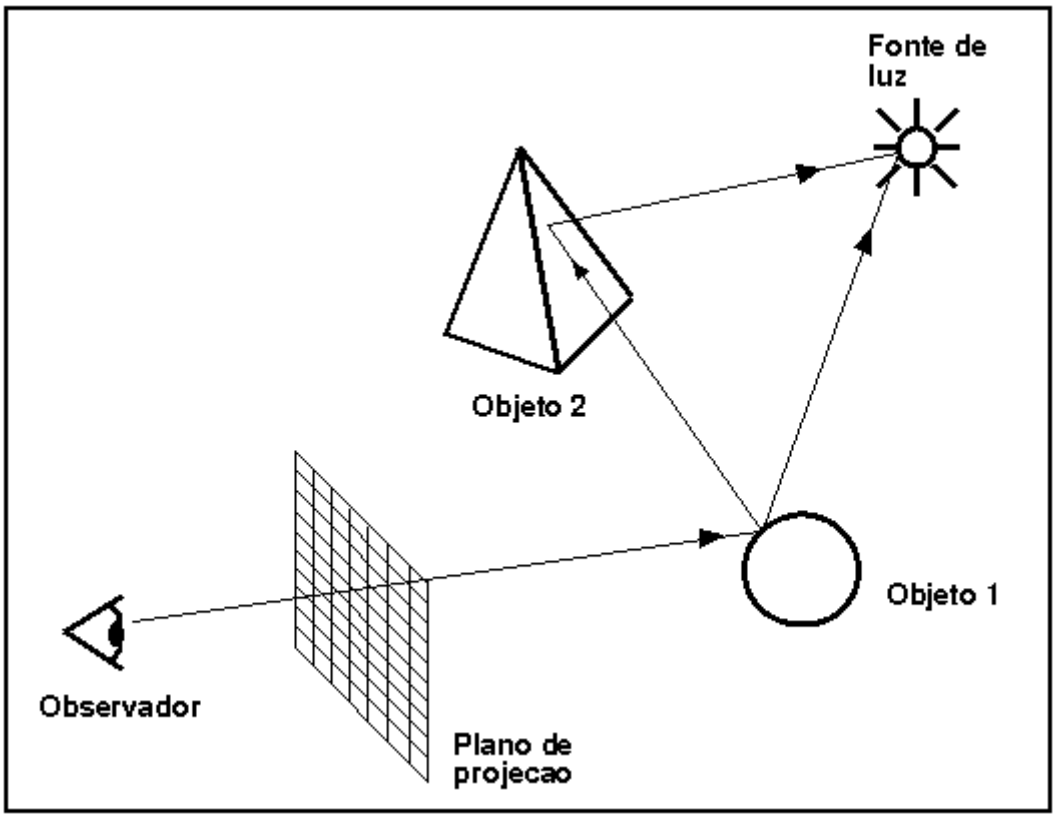


Figura 3.1 - Princípio básico de "Ray Tracing" (ISLER 1991).

3.3 Um Modelo de Iluminação

3.3.1 Compomente de Luz Ambiente

Inicialmente uma superfície tem uma "cor ambiente", que é o resultado da luz ambiente emitida uniformemente pelos objetos existentes ao redor. Ou seja, uma superfície é visível mesmo que não esteja exposta diretamente a uma fonte de luz. Neste caso, a superfície é iluminada pelos objetos vizinhos. A cor ambiente é independente da posição do observador. Pode-se expressar a luz ambiente num determinado ponto a partir da equação (ISLER 1991)

$$Color_p = K_d \; Color_a$$

onde K_d é o coeficiente de reflexão e $Color_a$ é a intensidade da luz ambiente. K_d terá valores entre 0 e 1, sendo 1 para superfícies altamente refletoras. Apenas a cor ambiente

não dá resultados satisfatórios em termos de realismo. Por isto, o efeito das fontes de luz nas superfícies, conforme a orientação destas, também deve ser levado em consideração nos cálculos de cor, ou seja, os reflexos difusos e especulares contribuirão ao realismo de uma cena.

3.3.2 Componente de Reflexão Difusa

Os cálculos das reflexões difusa e especular usam diversos vetores conforme ilustrado na figura 3.2, onde $\vec{V}$ é o vetor que parte do observador até o ponto de intersecção *I* do raio com a superfície, $\vec{N}$ é o vetor normal à superfície no ponto de intersecção, $\vec{L}$ é o vetor que vai do ponto de intersecção até a fonte de luz e $\vec{R}$ é o vetor reflexão. O ângulo descrito por $\vec{R}$ e $\vec{N}$ é igual ao ângulo descrito por $\vec{V}$ e $\vec{N}$.

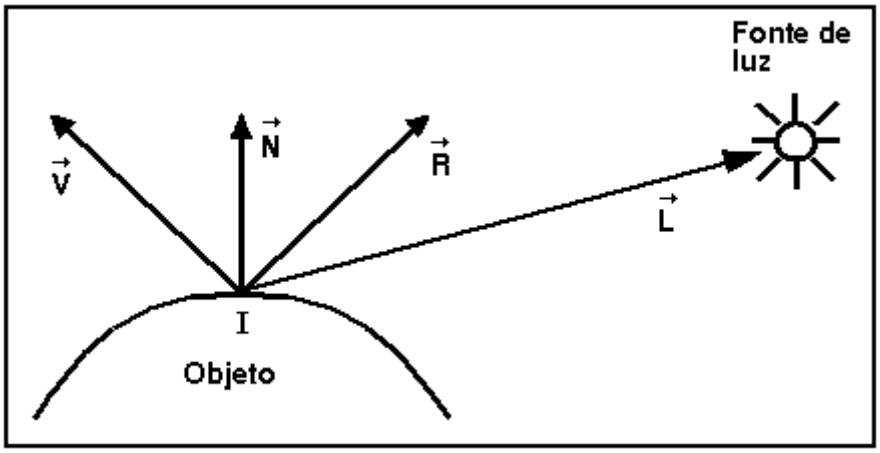


Figura 3.2 - Vetores existente num sistema de traçado de raios (ISLER 1991).

Os cálculos de iluminação difusa têm como base a lei do cosseno de Lambert, na qual a intensidade da luz refletida depende do cosseno do ângulo formado entre a normal à superfície e o raio de luz (PHONG 1975). O cosseno deste ângulo é o produto escalar de dois vetores normais nas mesmas direções que o vetor normal $\vec{N}$ e o vetor $\vec{L}$ da luz. A reflexão difusa pode ser calculada da seguinte forma (ISLER 1991)

$$Color_p = \frac{K_d \, Color_l}{d + d_0} (\vec{N} . \vec{L})$$

onde $Color_p$ é a intensidade previamente calculada para o ponto de intersecção I do raio com a supefície, K_d é uma constante para a reflexão difusa que depende das propriedades da superfície, $Color_l$ é a intensidade da fonte de luz. d representa a distância da fonte de luz ao ponto de intersecção e d_0 é uma constante para evitar que o denominador se torne nulo.

3.3.3 Componente de Reflexão Especular

Reflexos especulares são vistos pelo observador quando os raio de luz incidentes num objeto estão num determinado ângulo, e a superfície é refletora. A reflexão especular pode ser modelada da seguinte forma (ISLER 1991)

$$Color_p = \frac{K_s\ Color_l}{d + d_0} (\vec{R}.\vec{L})^n$$

onde n é uma constante relacionada com as propriedades óticas da superfície. É considerada nula (zero) quando a superfície é opaca, e possui um valor elevado quando a superfície é um "espelho perfeito". K_s é uma constante para o reflexo especular que depende das propriedades da superfície.

3.3.4 Reflexões

Para simular o efeito das reflexões ao redor do ponto de intersecção, um novo raio (raio refletido) é emitido a partir deste ponto e é testado com os objetos para verificar se atinge ou não algum destes. Se o raio atinge um objeto, a intensidade da cor no novo ponto de intersecção contribui para a determinação da cor no primeiro ponto de intersecção. Isto é feito através da expressão computacional abaixo (ISLER 1991)

$$Color_{p+1} = Color_p + K_r\ Color_r$$

onde $Color_p$ é a intensidade previamente calculada para o primeiro ponto e $Color_r$ é a intensidade no novo ponto de intersecção. K_r é uma constante relacionada às propriedades da superfície, que é o coeficiente de reflexão.

3.3.5 Transparência

Se o objeto atingido é transparente, os reflexos dos objetos por detrás do mesmo devem ser também levados em consideração. Esta contribuição é obtida da seguinte forma (ISLER 1991)

$$Color_p = (1-r)\ Color_t + r\ Color_b$$

onde $Color_t$ é a intensidade total no ponto de intersecção após a soma das intensidades da luz ambiente, do reflexo difuso e do especular. $Color_b$ é a intensidade do ponto de intersecção atrás do objeto transparente, e r é uma constante relacionada com a transparência do objeto. Se esta for zero o objeto será totalmente opaco. Em outras palavras, o raio que atinge a superfície continua se propagando através do objeto transparente até que atinge outro objeto. Este raio também pode ser refratado.

3.3.6 Sombras

Sombras, que aumentam o realismo da cena, podem ser obtidas enquanto se determinam as reflexões difusas e especulares. As regiões de uma superfície estarão na sombra se as fontes de luz são bloqueadas por qualquer objeto opaco ou semitransparente na cena. Isto é determinado enviando-se um raio do ponto de intersecção até a fonte de luz, e testando se existe ou não a intersecção deste raio com algum objeto. Se houver alguma intersecção, as reflexões difusas e especulares serão automaticamente zero.

3.4 Desempenho do Algoritmo

A generalidade e simplicidade do algoritmo de traçado de raios deve-se quase exclusivamente à sua dependência de duas únicas operações: o traçado de um raio (reta) para cada *pixel* do vídeo e o cálculo do ponto de intersecção do raio com as primitivas contidas na cena. Obviamente a segunda operação será a questão crítica na determinação do custo temporal na síntese de uma imagem. Segundo Whitted (WHITTED 1980) o tempo gasto no cálculo das intersecções em cenas complexas chega a superar os 95% do tempo total do algoritmo.

Embora elegantes soluções matemáticas tenham sido desenvolvidas para melhorar o cálculo das intersecções, a crescente complexidade das cenas tem criado uma constante demanda de procedimentos que visam a diminuição do número de intersecções calculadas para cada raio.

Diversos métodos criativos têm sido propostos para resolver esta questão envolvendo estruturas de dados (KAJIYA 1983) (KAY 1986), métodos numéricos (JOY 1986) (SWEENWY 1986) (TOTTH 1985) (VAN 1984), geometria computacional (MULLER 1986), ótica (SHINYA 1987), estatística (COOK 1986) (DIPPE 1985) (LEE 1985) (PURGATHOFER 1986) e processamento distribuído (CLEARLY 1985) (DIPPE 1984). Um levantamento destas técnicas foi apresentado por Figueroa (FIGUEROA 1993).

3.5 Conclusão

Neste capítulo foi explicado o funcionamento básico de um algoritmo de traçado de raios, indispensável para a compreensão deste trabalho. Foi destacado também que apesar da simplicidade e alto grau de realismo que os sistemas de traçado de raios apresentam, um fator crítico que determina o seu uso é a sua eficiência em termos de custo temporal; ressaltando assim a necessidade de implementações com baixo custo tanto em tempo quanto em flexibilidade.

4 PROPOSTA DE MODELO DE TRAÇADO DE RAIOS DE FENÔMENOS NATURAIS MODELADOS POR CAMPOS DE ALTITUDES

"Então desceu o Senhor...e disse...
não haverá restrição para tudo
o que eles intentarem fazer."
(Gênesis 11:5,6)

4.1 Introdução

Este capítulo tem por objetivo apresentar um modelo desenvolvido para a geração de imagens realísticas de fenômenos naturais através de um sistema de traçado de raios. Conforme visto anteriormente (capítulo 2), foi realizado um levantamento das principais abordagens utilizadas pelos sistemas de computação gráfica para a síntese de tais fenômenos. Foi também analisado o funcionamento do método conhecido como *Traçado de Raios (Ray Tracing)*, por ser um dos procedimentos computacionais que apresenta resultados com maior grau de realismo.

Diversas considerações foram feitas em ambos casos visando mostrar que: os atuais sistemas de simulação são, em extremo, específicos, implementando apenas um fenômeno em particular, sendo necessária uma nova implementação para a simulação de outro fenômeno; e que os sistemas de traçado de raios possuem um custo temporal muito alto quando utilizados em modelos complexos (tais como os existentes na natureza). As próximas seções apresentarão alguns conceitos relacionados com o exercício científico realizado neste trabalho e após fazer-se-á uma explanação do método proposto para a síntese de fenômenos naturais em sistemas de traçado de raios.

4.2 Campos de Altitude

Um campo de altitude (*height field*) é um conjunto de dados composto por uma matriz bidimensional de valores que representam altitudes. Um exemplo esclarecedor dos campos de altitudes são as imagens obtidas por satélites da superfície

terreste ou as imagens obtidas por ressonância da superfície no fundo do mar, onde pode-se observar o relevo da região pelas diferentes cores utilizadas. A figura 4.1 é um exemplo de tais imagens, onde os pontos mais escuros representam regiões mais baixas e os pontos mais claros regiões elevadas.

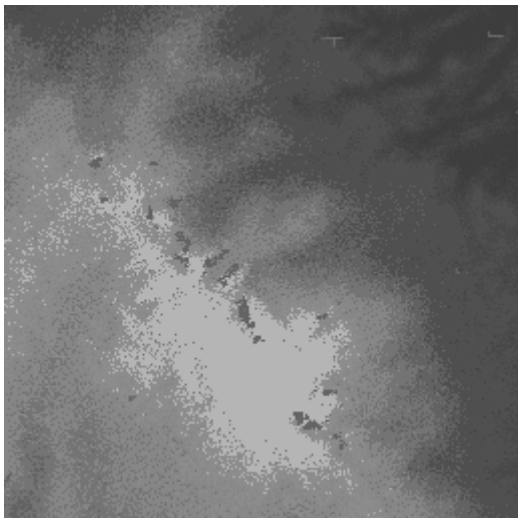


Figura 4.1 - Imagem de satélite de uma região montanhosa.

É importante que tais imagens possam ser visualizadas tridimensionalmente para se ter uma compreensão real da sua verdadeira forma.

A utilização de imagens como campos de altitude amplia os horizontes da Computação Gráfica. Imagens obtidas através de processos de digitalização podem ser visualizadas tridimensionalmente permitindo um estudo mais profundo de regiões geográficas existentes. Pesquisas oceanográficas, aero-espaciais, agrárias, populacionais, etc, podem ser enriquecidas, pois utilizam amplamente imagens bidimensionais.

Outras áreas de interesse como o cinema, a publicidade e a arte poderão criar cenários inexistentes e de alto grau de realismo a partir de imagens bidimensionais criadas em sistemas de desenho e/ou pintura.

4.2.1 Modelo de Imagens Utilizado

Neste trabalho os campos de altitude são considerados sempre como sendo imagens bidimensionais no formato *Graphics Interchange Format - GIF* desenvolvido pela Compuserve. Este formato foi escolhido devido a sua ampla divulgação na comunidade internacional, permitindo assim que qualquer imagem possa ser utilizada como um campo de altitude.

As imagens GIF são do tipo *bitmap* policromáticas e possuem uma tabela de *palette* de 256 cores. Neste tipo de imagens, cada ponto (*pixel*) da mesma possui uma informação associada indicando o índice, na tabela de *palette*, que contém a cor do ponto. O valor do índice na tabela de palete, independentemente da cor associada a ele, indicará a altitude da região, permitindo assim que os campos de altitude utilizados possuam altitudes que variam de 0 a 255 unidades.

O modelo proposto foi desenvolvido modularmente a fim de prever a inclusão de outros formatos de imagens, tais como o *Targa* ou *Tiff (Tagged Image File Format)*, permitindo assim que no futuro possam ser visualizados campos de altitudes com variações de altitude superiores a 256 níveis.

4.2.2 Seleção de Imagens como Campos de Altitude

A seleção das imagens a serem utilizadas como campos de altitude depende de diversos fatores, dentre os quais os mais importantes são: o objetivo da visualização e o resultado desejado. Neste trabalho foram utilizadas os dois tipos de imagens possíveis: tanto imagens com dados reais a respeito de regiões geográficas existentes, como também imagens criadas em sistemas de pintura para a geração de cenários virtuais e/ou resultados específicos.

4.2.2.1 Campos de Altitude Reais

Para a visualização de cenários reais foram utilizados arquivos da *United States Geological Survey (USGS)*, um orgão americano que tem criado mapas contendo dados tridimensionais de inúmeras regiões dos Estados Unidos, várias áreas submarinas

e inclusive de regiões extraterrestres (Lua, Marte). Grande parte do trabalho realizado pela USGS é de domínio público e pode ser facilmente obtido, principalmente através da rede internacional de comunicação Internet.

Este arquivos contém os dados tridimensionais em forma textual, padrão ASCII, mas podem ser facilmente transformados em imagens no padrão GIF. Existem diversos programas disponíveis para este fim.

Também foram utilizados arquivos no formato *DEM (Digital Elevation Map)*, uma derivação dos arquivos da USGS desenvolvida pela Virtual Reality Laboratories para uso pelo programa *Vista-Pro*. Estes arquivos permitem um total de 65.000 altitudes possíveis a intervalos de 30 metros entre cada uma. Existem, juntamente com o programa, utilitários de conversão dos arquivos DEM para imagens GIF.

É importante frisar que as imagens obtidas a partir dos arquivos DEM e/ou USGS perdem precisão no que diz respeito à definição das regiões no momento da conversão. Isto se deve a que o formato de imagens escolhido para a realização deste trabalho (GIF) permite apenas um total de 256 altitudes possíveis. Contudo, o fator acima mencionado não afeta significativamente nem o objetivo deste estudo nem os resultados almejados. A figura 4.2 mostra uma imagem obtida de um arquivo DEM.

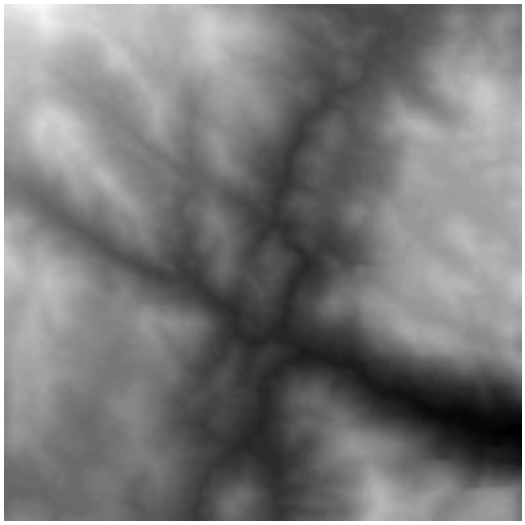


Figura 4.2 - Campo de altitude obtido de um arquivo DEM.

4.2.2.2 Campos de Altitude Virtuais

Para a obtenção de cenários virtuais, ou seja, cenários não existentes na natureza os campos de altitude foram produzidos de três formas.

A mais simples consistiu na utilização de imagens quaisquer já disponíveis nos mais diversos lugares. Desta forma seria possível se utilizar a imagem da bandeira do Brasil, por exemplo, como um campo de altitude. Obviamente que imagens como a anteriormente mencionada não apresentariam um resultado condizente com a simulação de fenômenos naturais. Contudo, imagens como a pele de um mosquito ampliada centenas de vezes, podem produzir um resultado interessante. Como fora mencionado anteriormente, a seleção das imagens depende do resultado desejado. É importante salientar que imagens monocromáticas (tonalidades de cinza) tendem a produzir melhores resultados.

O segundo método para a obtenção de campo de altitude virtuais foi a utilização de sistemas de pintura onde foram desenhadas imagens bidimensionais bem específicas visando-se um resultado em particular. Desta forma, no caso de se ter em mente uma cratera, bastaria criar diversos círculos concêntricos partindo-se de uma cor cujo índice na *palette* seja pequeno e ir aumentando este índice a cada novo círculo. Este tipo de imagem pode produzir resultados muito interessantes, dependendo da criatividade do desenhista como também dos recursos gráficos que o sistema de pintura ou processamento de imagens permita realizar. A figura 4.3 mostra uma imagem produzida num sistema de processamento de imagens e que pode ser utilizada como campo de altitude.

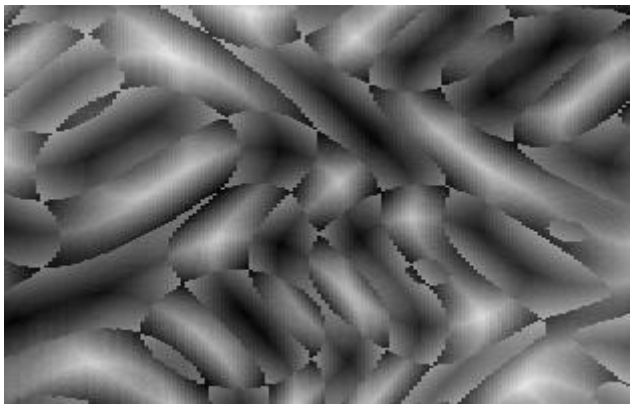


Figura 4.3 - Imagem produzida no sistema de processamento de imagens *Adobe Photoshop* e que pode ser utilizada como campo de altitude.

Finalmente, é possível se obter campos de altitude bem específicos através da implementação de determinados algoritmos, que produzirão como resultado uma imagem segundo um padrão determinado. Este tipo de recurso oferece uma infinidade de possibilidades, pois permite a geração de padrões gráficos dificilmente obtidos por mãos de artistas, como também a criação de seqüências de imagens com pequenas variações permitindo assim a criação de animações. Ao longo deste trabalho diversos algoritmos, visando fins específicos foram implementados, os quais serão vistos mais adiante.

4.3 Traçado de Raios de Campos de Altitude

A ideia de se visualizar fenômenos naturais através do uso de campos de altitude, ao invés dos modelos procedurais, foi gerada de uma necessidade. Definições procedurais de superfícies apoiam-se num conhecimento prévio sobre os polígonos utilizados na construção da forma da área a ser visualizada, para então serem determinadas as intersecções entre os raios e os polígonos. Pesquisas realizadas por Mandelbrot (PEITGEN 1988) têm proposto métodos de subdivisão poligonal para gerar modelos de superfícies montanhosas. Estes métodos embora iniciem com um simples polígono convexo tal como um triângulo ou hexágono evoluem até produzir objetos com bordas fractais similares às montanhas. Isto acontece porque os procedimentos de subdivisão não se "aninham" corretamente (PEITGEN 1988), contrariamente ao que acontece na subdivisão de um triângulo equilátero em quatro triângulos também equiláteros, pela união dos pontos médios dos seus lados. Como este não-aninhamento acontece sucessivamente, a área utilizada pelo polígono após a sua subdivisão é difícil

de ser determinada. A forma gerada dependerá do método de subdivisão utilizado e do número de vezes que esta subdivisão recursiva foi realizada. Desta forma, uma definição procedural de uma superfície torna-se impraticável. Eis porque é proposto este algoritmo mais genérico e prático.

4.3.1 Funcionamento do Algoritmo

Num sistema genérico de traçado de raios, o tempo necessário para gerar uma imagem é determinado principalmente pelo número de primitivas na cena; sendo assim, a maior parte do tempo de processamento, da unidade central de processamento, é gasto no cálculo das intersecções raio-objeto (RUBIN 1980) (WHITTED 1980). Diversas abordagens para reduzir o número de objetos a serem interseccionados têm sido propostas (ARVO 1987) (FUJIMOTO 1986) (GLASSNER 1984) (KAY 1986). É exatamente numa destas propostas que o método implementado neste trabalho para a geração de imagens de fenômenos naturais tem suas origens. O algoritmo de 3DDDA proposto por Fujimoto (FUJIMOTO 1986) é semelhante ao implementado neste trabalho, embora se utilize nesta pesquisa uma abordagem mais eficiente por estar limitada à síntese de superfícies bidimensionais descritas por campos de altitude.

Desta forma, os campos de altitude serão considerados como sendo um novo tipo de primitiva básica de um sistema de traçado de raios, e podem ser facilmente implementado em sistemas genéricos.

Um campo de altitude é considerado essencialmente como sendo um paralelepípedo de uma unidade de altura, uma unidade de comprimento e uma unidade de largura, contendo no seu interior um conjunto de triângulos definindo uma superfície. A altura de cada triângulo contido neste paralelepípedo é obtida através do índice na *palette* de uma imagem.

A resolução da imagem utilizada não influencia o tamanho do campo de altitude. Um campo de altitude terá sempre as dimensões 1x1x1 para fins de cálculo. Contudo, ele poderá ser depois escalado, transladado e/ou rotacionado como qualquer outra primitiva geométrica. Quanto maior a resolução da imagem maior será o número de triângulos dentro do paralelepípedo, produzindo assim uma superfície mais suave.

Assim, uma imagem GIF de resolução 1024x1024 produzirá um campo de altitude mais suave que outra imagem de resolução 128x128.

4.3.1.1 Intersecção dos Raios com o Campo de Altitude

Para determinar a intersecção de um raio com um campo de altitude, o sistema primeiro verifica se houve intersecção do raio com o paralelepípedo que envolve o conjunto de triângulos. Se o envelope é interseccionado, então o sistema determina em que região o raio entra e sai do paralelepípedo verificando então cada ponto da imagem que é cruzado pelo raio. A figura 4.4 ilustra o procedimento.

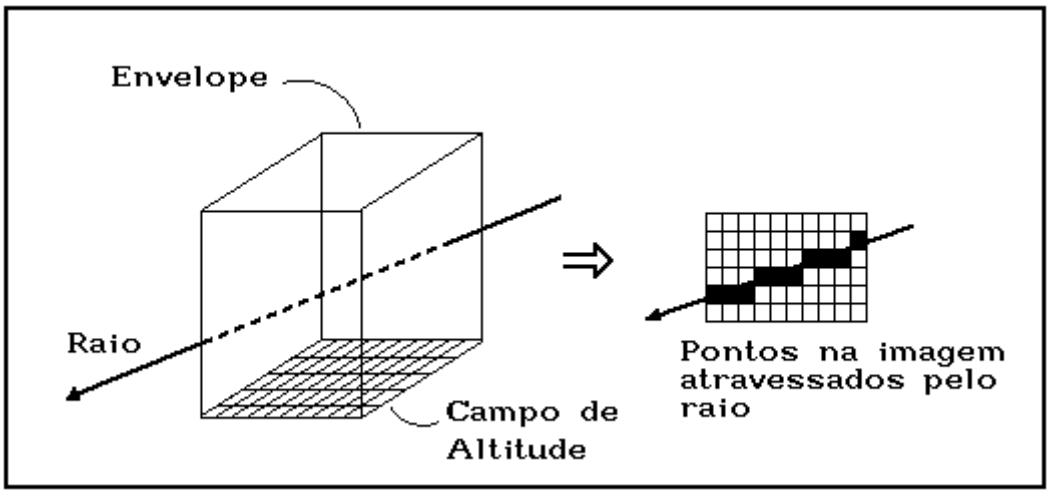


Figura 4.4 - Raio atravessando um campo de altitude

Para determinar que pontos da imagem são atravessados pelo raio é utilizado um algoritmo implementado na geração de linhas retas em dispositivos matriciais (vídeos) conhecido como Analisador Diferencial Digital - *DDA* (ROGERS 1985) (FUJIMOTO 1986). Este é um dos algoritmos mais conhecidos dentro da Computação Gráfica e tem sido implementado inclusive em nível de *hardware* em diversas máquinas.

4.3.1.1.1 O Algoritmo DDA

O princípio básico do funcionamento do algoritmo de DDA é descrito a seguir. No processo de determinação das coordenadas dos *pixels* que descrevem uma

linha reta, calcula-se a inclinação da reta, a fim de selecionar o eixo (*X* ou *Y*) em que o deslocamento (Δx ou Δy) é maior. O eixo que apresentar maior deslocamento, denominado de eixo de direção, será incrementado incondicionalmente de um *pixel* (menor unidade de deslocamento). Simultaneamente um termo de controle (correção) é atualizado subtraindo-se o valor da inclinação da reta e verificando se este resultado é menor que o tamanho de meio *pixel*. Quando o resultado deste teste for negativo, a coordenada do eixo perpendicular ao de direção é acrescida (ou subtraída) de uma unidade (*pixel*); caso contrário não. O termo de controle é corrigido pela adição do valor correspondente a menor unidade de deslocamento toda vez que o seu valor torna-se menor que o tamanho de um pixel. O leitor pode ainda referenciar outras fontes de consulta para maiores detalhes (ROGERS 1985) (FUJIMOTO 1986) (FOLEY 1990) (THALMANN 1987).

4.3.1.1.2 Alterações no Algoritmo DDA

Existem duas modificações fundamentais a serem feitas no algoritmo DDA de traçado de linhas para a travessia de um raio por uma imagem.

Primeiramente o DDA identifica uma célula de cada vez à medida que se desloca em uma direção, e no nosso caso é necessário que o algoritmo DDA determine não apenas uma célula de cada vez, mas sim todas as células atravessadas pelo raio. Este processo deve ser realizado com cuidado a fim de identificar às células na ordem exata em que são atravessadas pelo raio desde o ponto A até *B*.

A segunda modificação importante que o algoritmo DDA genérico deve sofrer é a nível numérico. Como se está lidando com valores que vão de 0 a 1 conforme estabelecido anteriormente (envelope de dimensões 1x1x1) o algoritmo proposto por Bresenham (ROGERS 1985) que retorna valores inteiros torna-se inapropriado para nosso objetivo. O algoritmo deve ser então modificado de forma a produzir intersecções em ponto flutuante entre as bordas de cada célula, ao invés de simplesmente fornecer as coordenadas *x, y* inteiras de cada célula da imagem.

Outras pequenas alterações que não interferem no funcionamento do algoritmo têm sido realizadas. Estas modificações são de caráter temporal visando à aceleração do mesmo.

4.3.1.2 Determinação da Superfície

Uma vez determinadas todas as células por onde o raio passa, é verificada se existe realmente a intersecção do raio com a superfície do campo de altitude.

À medida que o raio passa pelos pontos da imagem determinados pelo DDA, a altitude do raio é comparada com as quatro altitudes associadas a cada célula. O cálculo de intersecção do raio com a superfície será necessário apenas se a altitude do raio estiver contida entre o intervalo definido pelo menor e maior valor de altitude associado com a célula.

Para um raio atravessando sobre a superfície do campo de altitude a condição acima descrita pode ser definida por

$$Min(raio_{Z_{prox}}, raio_{Z_{dist}}) \leq Max(h_{i,j}, h_{i,j+1}, h_{i+1,j}, h_{i+1,j+1})$$

onde $raio_{Z_{prox}}$ e $raio_{Z_{dist}}$ representam as altitudes dos pontos origem e destino do segmento de reta contido dentro da célula atravessada e $h_{m,n}$ as altitudes do campo de altitude nos quatro vértices da célula *i,j*.

Cada *pixel* é então dividido em dois triângulos, onde a coordenada *y* de cada vértice é determinada pelos índices da *palette* de cores. A figura 4.5 mostra como é criado este conjunto de triângulos.

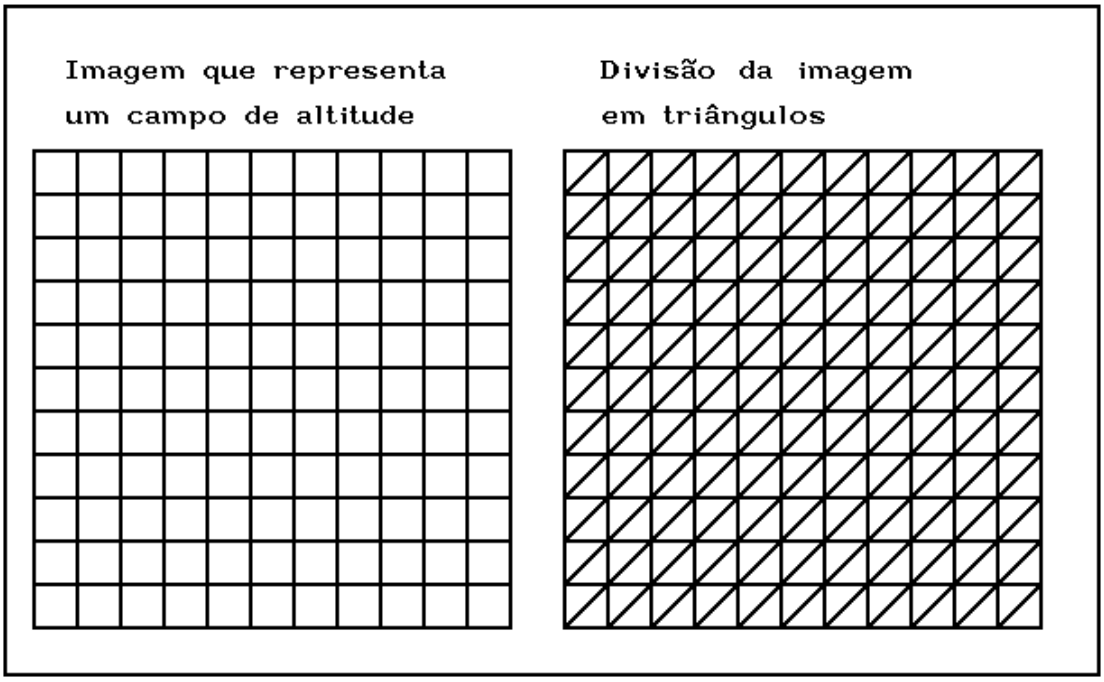


Figura 4.5 - Divisão de uma imagem num conjunto de triângulos.

Desta forma o teste de intersecção raio-superfície consiste em dois testes de intersecção raio-triângulo. A figura 4.6 mostra uma superfície definida por um conjunto de triângulos obtido de uma imagem bidimensional.

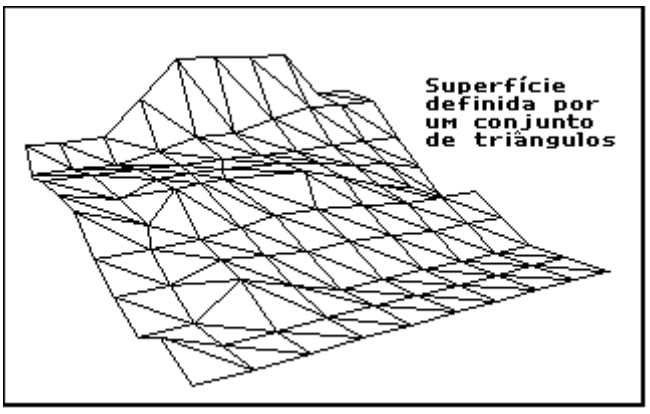


Figura 4.6 - Superfície definida por um campo de altitude.

O cálculo das intersecções em cada célula é realizado da seguinte forma:

1) Determinar a intersecção do raio com os planos definidos por cada triângulo contido na célula,
2) Verificar se o ponto de intersecção se encontra dentro de um dos triângulos.

O primeiro passo é comum a todo sistema de traçado de raios; e o segundo pode ser implementado de forma bastante eficiente e rápida.

4.3.1.3 Nível do Mar.

Neste trabalho foi introduzido um parâmetro opcional denominado de *Water_Level* (nível do mar). Este parâmetro tem como função indicar ao sistema de traçado de raios um valor mínimo de altitude, abaixo do qual não serão consideradas outras altitudes. Os valores permitidos estão contidos no intervalo 0 a 1.

Por exemplo, se *water_level=0,5* então somente a metade superior do *height field* será considerada, e a metade inferior será desprezada. Um campo de altitude pode ser utilizado para sintetizar uma ilha num oceano, onde uma grande parte do campo de altitude estará oculta por outra primitiva qualquer que represente a água. Este parâmetro permitirá que as partes ocultas sejam desconsideradas, minimizando assim o número de cálculos de intersecção raio-triângulos, como também economizando tempo.

O nível do mar permitirá também que sejam eliminadas partes não desejadas de alguns campos de altitude. Assim, no caso de se possuir uma imagem de uma superfície fractal sobre um fundo vermelho, onde a cor do fundo referenciar o índice 0 da *palette*, será possível eliminar o fundo não desejado especificando *water_level=0,01*, por exemplo.

Esta operação é realizada verificando se a altura do raio que passa pelas células determinadas pelo DDA é superior à altura determinada como nível do mar. Este teste obviamente será realizado antes de verificar se a altitude do raio está contida entre a maior e menor altitude dos quatro cantos da célula. O benefício advindo deste recurso, redução do número de testes de intersecção realizados, é extraordinário diante do custo de implementação, apenas um teste a mais.

4.3.1.4 Implementação Modular

Este algoritmo de traçado de raios de campos de altitude foi implementado de forma modular e gradual, de forma a permitir a sua futura inclusão em outros sistemas de traçado de raios já existentes. A seguir serão descritas as etapas seguidas na implementação deste algoritmo:

4.3.1.4.1 Intersecção de Raio com Envelope do Campo de Altitude

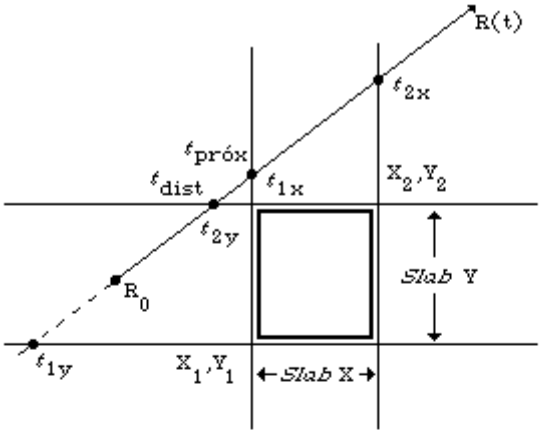
Nesta etapa são calculadas as coordenadas *x,y,z* do ponto de intersecção de um raio com o envelope (paralelepípedo) do campo de altitude.

O princípio básico do sistema de traçado de raios consiste na emissão de um raio (reta) com origem no centro de projeção e que passa por um ponto (*pixel*) da imagem a ser sintetizada. Visto que um raio pode atingir qualquer um dos seis lados de um paralelepípedo, deverão ser testadas as seis possíveis intersecções.

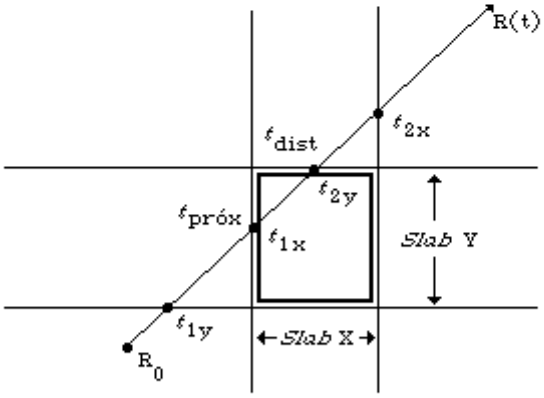
Cada teste de intersecção é realizado em duas etapas. A primeira etapa consiste na determinação do ponto de intersecção do raio com o plano definido pela face do cubo. Depois é verificado se o ponto de intersecção com o plano está contido dentro dos limites definidos pela face do cubo.

Kay e Kajiya (KAY 1986) apresentaram um método para o cálculo de tais intersecções baseados em *slabs*. Um *slab* é simplesmente o espaço definido por dois planos paralelos. A intersecção de um conjunto de *slabs* define um paralelepípedo. O método tem como base a intersecção de dois *slabs* pelo raio, armazenando o ponto de intersecção mais próximo e o mais distante para cada *slab*. Se o maior valor dos pontos de intersecção mais próximos for maior que o menor valor dos pontos de intersecção mais distantes, então o raio não intersecciona o paralelepípedo; caso contrário sim.

A figura 4.7 mostra dois casos de testes de intersecção, um interseccionando o paralelepípedo e o outro não; considerando apenas os planos *X* e *Y*.



$t_{próx} > t_{dist}$ então não intersecciona



$t_{próx} < t_{dist}$ e $t_{próx} > 0$ então $t_{próx}$ e o ponto de intersecção

Figura 4.7 - Testes de intersecção de um raio com um paralelepípedo (GLASSNER 1991).

O algoritmo a seguir considera cada um dos planos paralelos com as normais na mesma direção que os planos *X*, *Y* e *Z*, visando tornar mais eficiente o cálculo das intersecções. Este algoritmo devolve a variável booleana: TRUE se o raio intersecciona o paralelepípedo, caso contrario retorna: FALSE.

Definindo-se o paralelepípedo através de duas coordenadas correspondentes ao ponto superior esquerdo e inferior direito

$$P_1 = (X_1, Y_1, Z_1)$$
$$P_2 = (X_2, Y_2, Z_2)$$

e o raio através do seu ponto de origem e o vetor de direção

$$R_o = (X_o, Y_o, Z_o)$$
$$R_d = (X_d, Y_d, Z_d)$$

então raio será definido pela função (GLASSNER 1991)

$$R(t) = R_o + R_d\ t$$

sendo $t > 0$. O algoritmo é descrito a seguir, descrevendo as operações apenas para o plano *X*:

$t_{próx} = -\infty$ e $t_{dist} = \infty$ (ou valores muito grandes)

Para cada par de planos associado com X;

Se direção $X_d = 0$ então raio paralelo ao plano

Se origem $X_o < X_1$ ou $X_o > X_2$ (não entre os planos) então FALSE

Senão

{

Calcular as interseções nos planos

$$t_1 = (X_1 - X_o) / X_d$$
$$t_2 = (X_2 - X_o) / X_d$$

Se $t_1 > t_2$ então trocar t_1 por t_2

Se $t_1 > t_{próx}$ então $t_{próx} = t_1$

Se $t_2 < t_{dist}$ então $t_{dist} = t_2$

Se $t_{próx} > t_{dist}$ então (não intersecciona) FALSE

Se $t_{dist} < 0$ então (paralelepípedo atrás do raio) FALSE

}

Fim_para

TRUE (Satisfez todos os testes)

Se o raio interseccionou o paralelepípedo então o ponto de intersecção mais próximo é fornecido por $t_{próx}$ e o ponto de saída do paralelepípedo por t_{dist}.

4.3.1.4.2 Implementação do DDA

Implementação de um algoritmo DDA padrão para atravessar bidimensionalmente o campo de altitude iniciando pelo *pixel* cujas coordenadas coincidem com as coordenadas *X* e *Y* do ponto de intersecção $t_{próx}$ até o *pixel* correspondente ao ponto t_{dist}.

Definindo-se o paralelepípedo através de duas coordenadas correspondentes ao ponto superior esquerdo e inferior direito

$$P_1 = (X_1, Y_1, Z_1)$$
$$P_2 = (X_2, Y_2, Z_2)$$

os pontos de intersecção (entrada e saída) do raio com o envelope (paralelepípedo) do campo de altitude

$$P_e = (X_e, Y_e, Z_e)$$
$$P_s = (X_s, Y_s, Z_s)$$

e a resolução R_x x R_x da imagem utilizada como campo de altitude as coordenadas I, J das células de entrada e saída do raio no campo de altitude serão determinadas por

$$I_e = INT((X_e - X_1) / ((X_2 - X_1) / R_x))$$
$$J_e = INT((Y_e - Y_1) / ((Y_2 - Y_1) / R_y))$$

A figura 4.8 mostra graficamente este procedimento.

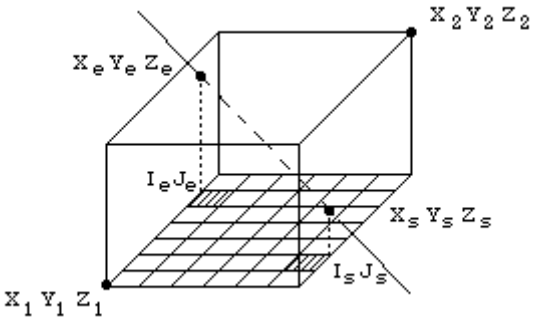


Figura 4.8 - Identificação das células de entrada e saída de um raio num campo de altitude.

Uma vez determinadas as células de entrada e saída do raio no campo de altitude é realizada a determinação das células que desenhariam uma linha reta através do algoritmo de DDA conforme descrito por Foley (FOLEY 1990). A figura 4.9 mostra os pontos células determinados pelo algoritmo para uma dada linha reta.

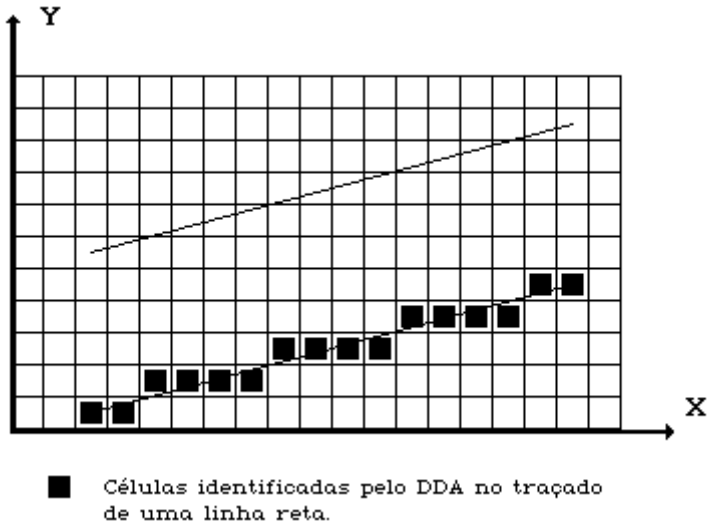


Figura 4.9 - Células marcadas pelo algoritmo DDA padrão.

4.3.1.4.3 Determinação de Todo os Pontos Atravessados pelo Raio

Este item mostra as alterações realizadas no DDA para identificar não apenas as células que desenhariam a melhor linha reta definida pelo raio, mas sim todas as células atravessadas por este. A figura 4.10 mostra a determinação de todas as células atravessadas pelo raio.

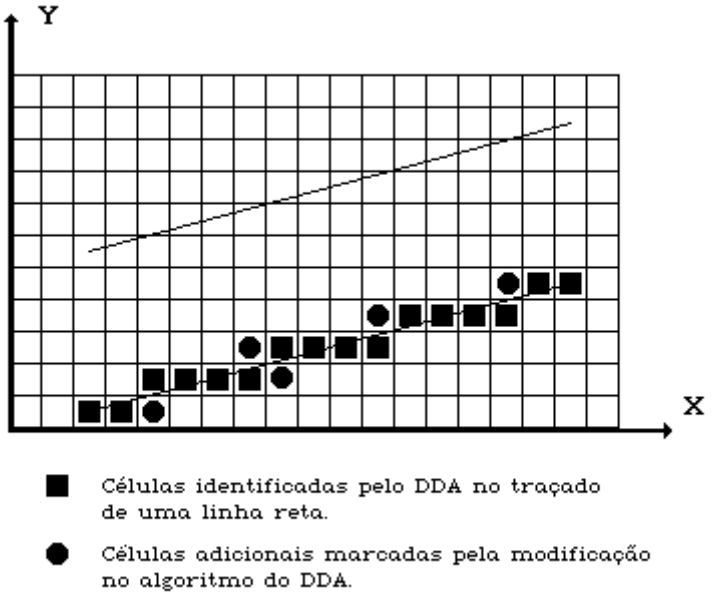


Figura 4.10 - Células determinadas pelo algoritmo DDA modificado.

Para isto são realizadas duas modificações simples, sugeridas por Fujimoto (FUJIMOTO 1986), ao algoritmo de DDA descrito anteriormente na seção 4.3.1.1.1. Primeiramente, o movimento perpendicular ao eixo de direção, que no DDA padrão está acoplado ao movimento do eixo de direção, é separado deste a fim de garantir que a linha passe de um pixel a outro obrigatoriamente adjacente. Em segundo lugar, o valor limite com o qual é testado o termo de controle deve ser zero e não mais meio *pixel* .

Esta alteração é necessária visto ser possível a interseção do raio com qualquer célula atravessada, por mais tangencial que possa ser a travessia, ao contrário do algoritmo DDA padrão cuja preocupação é a determinação das células que

visualmente representam a linha reta mais perfeita, desconsiderando as intersecções mais tangenciais.

4.3.1.4.4 Consideração da Altitude do Raio

Modificação do DDA para levar em consideração a coordenada *z* (altitude) do raio em cada célula desde o ponto de entrada no envelope até o momento de sua saída. O valor da coordenada *z*, que até então não era considerado será calculado a cada determinação de um novo pixel, pois os valores de *z* de entrada e saída do raio no paralelepípedo são conhecidos.

Uma vez realizadas todas as alterações anteriormente mencionadas e levando em consideração o valor da altitude *z* o algoritmo DDA para atravessar uma matriz bidimensional de altitudes fica conforme descrito a seguir. A descrição do algoritmo é feita considerando-se uma metalinguagem similar à linguagem **C**, utilizada na implementação do protótipo.

```
/* Início do algoritmo */

        /* Determina posição inicial */
        Z_próx = Coordenada Z do ponto de entrada;
        Z_dist = Coordenada Z no ponto de saída;
        Cel_x = Índice X da célula correspondente ao ponto de
             entrada;
        Cel_y = Índice Y da célula correspondente ao ponto de
             entrada;

        /* Determina variáveis do DDA */
        contr = termo de correção do DDA;
        delta = delta para a correção;
        delta_x = delta para X;
        delta_y = delta para Y;
        delta_z = delta para Z;

        /* a função SINAL() retorna +1 se o argumento for positivo,
             caso contrário retorna -1 */
        sinal_x = SINAL(raio->dir.x);
```

```
sinal_y = SINAL(raio->dir.y);


/* Laço principal, realizado enquanto não for atingida a
     borda de saída do campo de altitude. O eixo X é o eixo
     de direção */
faça
  {
       Z_menor = MIN(Z_próx, Z_dist);
       se (Z_menor <= Maior_altitude[Cel_x][Cel_y])
            se (Intersecção(Cel_x, Cel_y, raio))
                 retorna;
            se (contr > MUITO_PEQUENO)
              {
                /* Verifica as células que um DDA padrão
                     desconsideraria */
                Cel_y += sinal_y;
                se ((Cel_y < 0) || (Cel_y == Comp_lado))
                     /* o raio atingiu o limite do campo
                          de altitude */
                     break;
                senão
                     se    (Z_menor    <=    Maior_altitude
                          [Cel_x][Cel_y])
                          se   (Intersecção(Cel_x,   Cel_y,
                               raio))
                               retorna;
                contr--;
              }
            senão se (contr > -MUITO_PEQUENO)
              {
                /*  raio atravessa exatamente o canto da
                     célula */
                Cel_y += sinal_y;
              }
            Cel_x += sinal_x;
            contr += delta;
            Z_próx = Z_dist;
            Z_dist += delta_z;
  }
```

```
        enquanto ( (Z_próx >= Limite_inferior_envelope) &&
                   (Z_próx <= Limite_superrio_envelope) &&
                   (Cel_x >= 0) && (Cel_x < Comp_lado) &&
                   (Cel_y >= 0) && (Cel_y < Comp_lado) );

/* Término do algoritmo */
```

4.3.1.4.5 Comparação da Altitude do Raio com as Quatro Altitudes da Célula

Comparação dos valores de *z* de entrada e saída do raio em cada célula com as quatro altitudes associadas aos cantos da célula. Se o resultado da comparação

$$Min(raio_{Z_{prox}}, raio_{Z_{dist}}) \leq Max(h_{i,j}, h_{i,j+1}, h_{i+1,j}, h_{i+1,j+1})$$

for positivo, ou seja, se *z* estiver dentro do intervalo definido pelas altitudes máxima e mínima da célula, prosseguir com o passo 6, caso contrário, avançar para a próxima célula e repetir a operação.

4.3.1.4.6 Divisão da Célula em Triângulos

São primeiramente determinados os pontos tridimensionais respectivos aos vértices dos dois triângulos, com base nos quatro vértices da célula interseccionada. Depois serão determinadas e armazenadas duas primitivas (triângulos) que dividem a célula a partir dos quatro valores de altitude associados a ela.

Sejam as quatro altitudes associadas a uma célula

$h_{i,j}$,
$h_{i,j+1}$,
$h_{i+1,j}$ e
$h_{i+1,j+1}$

os dois possíveis triângulos que a dividem serão definidos por

$$
\begin{aligned}
T_1 &= (Xh_{i,j}, Yh_{i,j}, Zh_{i,j}) \\
&\quad (Xh_{i,j+1}, Yh_{i,j+1}, Zh_{i,j+1}) \\
&\quad (Xh_{i+1,j+1}, Yh_{i+1,j+1}, Zh_{i+1,j+1}) \\
T_2 &= (Xh_{i,j}, Yh_{i,j}, Zh_{i,j}) \\
&\quad (Xh_{i+1,j+1}, Yh_{i+1,j+1}, Zh_{i+1,j+1}) \\
&\quad (Xh_{i+1,j}, Yh_{i+1,j}, Zh_{i+1,j})
\end{aligned}
\quad .
$$

É importante salientar que é apenas neste momento que um objeto tridimensional é determinado (com exceção do envelope) e armazenado pelo sistema de traçado de raios.

4.3.1.4.7 Intersecção do Raio com os Triângulos da Célula

Esta etapa determina os pontos de intersecção do raio com os dois triângulos que definem a superfície da célula. Estes testes de intersecção são realizados em duas etapas:

- Determinar o ponto de intersecção do raio com o plano definido pelo triângulo e
- Verificar se o ponto de intersecção está contido dentro do espaço definido pelo triângulo.

Visto um plano ser considerado como uma primitiva básica do sistema de traçado de raios os procedimentos necessários para a determinação da intersecção de um raio com este é similar ao cálculo de intersecções descrito no anexo A-1. Este mesmo procedimento já foi utilizado anteriormente pela rotina de cálculo da intersecção do raio com o envelope do campo de altitude (secção 4.3.1.4.1).

A verificação da inclusão do ponto de intersecção determinado dentro do triângulo pode ser realizada de forma muito rápida, e pode ser expressa da seguinte forma

$$
\begin{aligned}
&\textbf{se} \quad (\text{intersecção}_x >= \text{célula}_{x_{min}}) \;\&\& \\
&\qquad (\text{intersecção}_y >= \text{célula}_{y_{min}}) \;\&\&
\end{aligned}
$$

$$(\text{intersecção}_x + \text{intersecção}_x <= \text{célula}_{x_{max}} + \text{célula}_{y_{max}})$$

então intersecção=TRUE;

onde $\text{intersecção}_{[xy]}$ é a coordenada *x* ou *y* do ponto de intersecção do raio com o plano definido pelo triângulo e $\text{célula}_{[xy]_{[min\ max]}}$ é o menor ou maior valor das coordenadas *x,y* dos quatro cantos da célula. Não são levadas em consideração as altitudes, pois estas já foram testadas anteriormente, ver seção 4.3.1.4.5.

Os passos aqui descritos representam a implementação básica do algoritmo. Outras otimizações foram realizadas no algoritmo, mas são em nível de implementação na linguagem C, como o uso de variáveis *double* ao invés de *long*, por exemplo.

4.3.2 Colorização do Campo de Altitude

O campo de altitude, por ser uma primitiva geométrica do sistema, terá a sua cor definida pelo algoritmo de traçado de raios conforme foi descrito no capítulo 2. Para este fim, será atribuído ao mesmo uma única cor que será a cor de todos os triângulos que estão contidos dentro do envelope. Assim, toda a superfície terá a mesma cor.

Esta abordagem permite a obtenção de resultados muito interessantes tais como dunas de areia. Contudo, para a síntese de determinados fenômenos naturais, tais como montanhas, faz-se necessário que as regiões mais baixas do campo de altitude possuam uma coloração verde (existência de vegetação) e uma coloração branca nas regiões mais altas (existência de neve). É importante então que seja elaborado um sistema que permita a existência de diversas cores numa única superfície definida por um campo de altitude.

Para solucionar este problema, foi utilizada a seguinte abordagem.

Na existência do parâmetro *color_map*, serão consideradas os índices da *palette* como sendo as altitudes de cada um dos cantos de uma célula e também a cor referenciada na tabela de *palette* por tais índices. Sendo assim, os índices da *palette*

indicam a elevação dos vértices do triângulo e a cor é determinada pela média das cores (R, G, B) referenciadas na *palette* pelos três pontos que determinam o triângulo. A figura 4.11 mostra parte de uma imagem colorida e parte do conjunto de triângulos colorizados por este procedimento. Observar-se-á que através deste processo serão criadas cores intermediárias não existentes na imagem, o que contribuirá para gerar uma superfície com coloração mais suave e evitando assim o serrilhamento na imagem sintetizada. É importante salientar que embora as cores do campo de altitude não sejam idênticas as da imagem original existe uma clara e notória semelhança entre ambas.

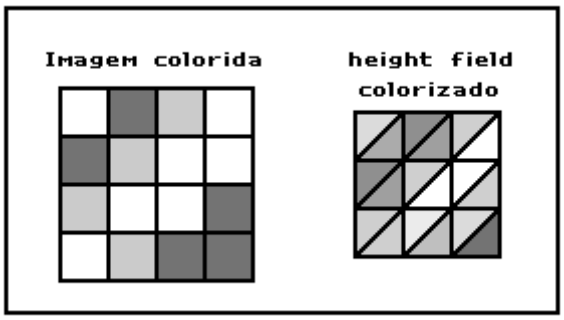


Figura 4.11 - Parte de uma imagem e a sua semelhança nas cores produzidas na superfície do campo de altitude.

Para que as montanhas possam ser representadas corretamente, é fundamental que exista uma correspondência entre o índice da *palette* e a cor referenciada por este índice. Assim, as cores devem ser ordenadas de acordo com a altitude que representam e colocadas nesta ordem na tabela de *palette*.

4.4 Conclusão

Neste capítulo foi apresentado um modelo para o traçado de raios de campos de altitude. Foram apresentados o tipo de dado utilizado com campos de altitude, a seleção deste conjunto de dados, o algoritmo utilizado para a síntese e coloração dos dados, bem como um esquema simples de implementação deste trabalho em sistemas já existentes.

5 PROTÓTIPO IMPLEMENTADO

"A exposição das tuas palavras dá luz,
dá entendimento aos simples."
(Salmo 119:130)

5.1 Introdução

Visando a validação do modelo aqui apresentado, foi desenvolvido um sistema de traçado de raios utilizando campos de altitude de acordo com as características descritas nos capítulos 3 e 4.

A implementação inicial do algoritmo foi uma extensão de um sistema de traçado de raios já desenvolvido, como um trabalho anterior de pesquisa (FIGUEROA 1993). Este sistema possui uma entrada de dados textual, contendo a descrição da cena a ser sintetizada, e uma saída de dados via arquivo, mais especificamente um arquivo no formato *Targa (TGA)*. O algoritmo foi desenvolvido na linguagem de programação C num microcomputador 80486 DLC 40 MHz com oito megabytes de memória. O sistema foi portado para estações Sun.

Este sistema é limitado, possuindo apenas as seguintes primitivas básicas: triângulo, esfera, planos, paralelepípedos, cones, cilindros e campos de altitude. No sistema não foram implementadas rotinas de mapeamento de texturas, embora isto seja considerado como o próximo passo após a conclusão deste trabalho.

5.2 Módulos do Protótipo

A arquitetura do protótipo implementado foi definida a partir das diferentes tarefas realizadas pelo sistema. A figura 5.1 apresenta a disposição gráfica desta arquitetura

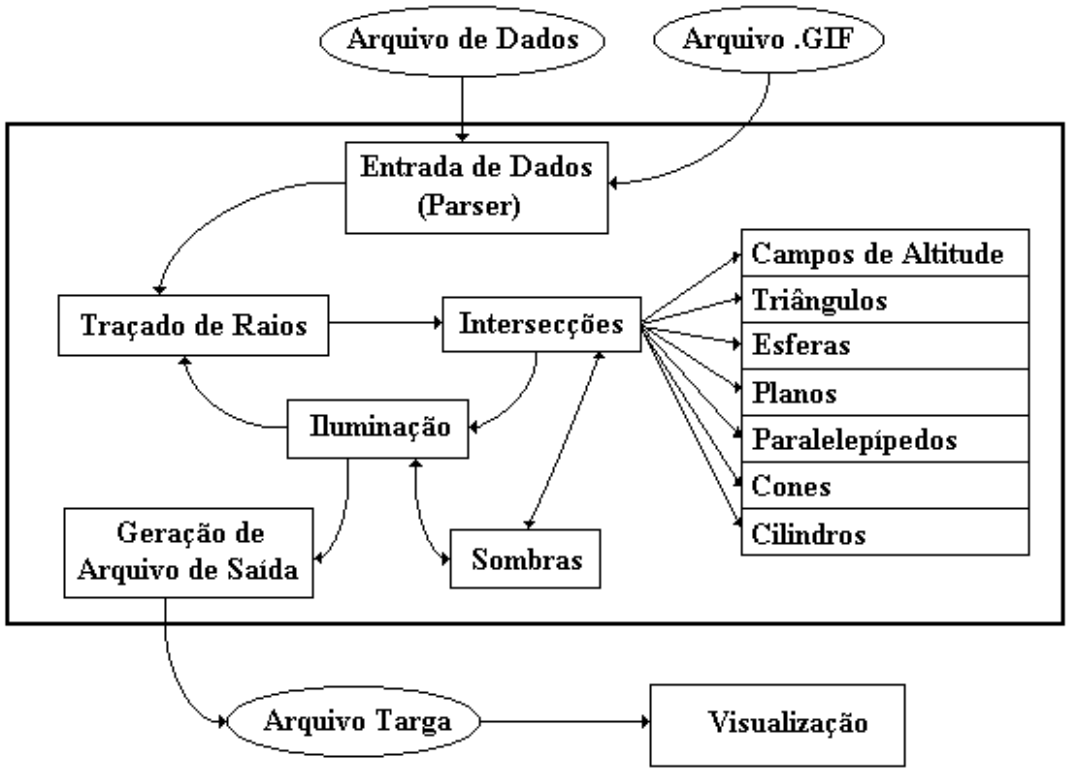


Figura 5.1 - Arquitetura do protótipo implementado.

O módulo de **Entrada de Dados (Parser)** faz a leitura da sintaxe do arquivo de entrada e monta as devidas estruturas de dados pertinentes à modelagem e especificação da cena a ser sintetizada. Neste processo a imagem que representa o campo de altitude é lida e armazenada pelo sistema. A linguagem utilizada pelo arquivo de entrada de dados será apresentada na 5.4

O módulo **Traçado de Raios** funciona como o programa principal do sistema, determinando as funções a serem realizadas. Este módulo realiza o traçado de raios para cada ponto da imagem a ser sintetizada a partir da especificação geométrica definida pelo módulo de Entrada de Dados.

O módulo **Intersecções** é o responsável pelo cálculo das possíveis intersecções dos raios com os objetos definidos na cena. Este módulo faz o cálculo de intersecções de um raio com cada uma das primitivas na cena, determinando os procedimentos matemáticos específicos necessários para cada primitiva. É composto por outros sete módulos, a saber:

- triângulos, implementa as equações de intersecção raio com um triângulo;
- esferas, implementa as equações de intersecção raio com uma esfera;
- planos, implementa as equações de intersecção raio com um plano;
- paralelepípedos, implementa as equações de intersecção raio com um paralelepípedo;
- cones, implementa as equações de intersecção raio com um cone;
- cilindros, implementa as equações de intersecção raio com um cilindro e
- campos de altitude, implementa todos os procedimentos descritos no capítulo 4, para a determinação da interseção de um raio com um campo de altitude, ou mais especificamente, com a superfície definida pelo campo de altitude.

Ver no anexo A-1 as listagens correspondentes a cada um dos submódulos acima descritos.

O módulo **Iluminação** é responsável pela determinação da colorização do ponto na superfície de uma primitiva. Este módulo determina as contribuições que a luz ambiente, a reflexão difusa e a reflexão especular fornecem ao objeto. São também determinadas as influências que as características de reflexão e transparência do objeto provocam na coloração do ponto de intersecção. Este módulo em algumas situações é chamado recursivamente.

O módulo **Sombras** implementa a determinação da existência de sombras (redução ou ausência de intensidade na cor do ponto de intersecção) na cena a ser sintetizada.

O módulo **Geração de Arquivo de Saída** faz a gravação dos dados calculados num arquivo formato *Targa*. Este módulo grava as componentes (vermelha, verde e azul) calculadas pelos módulos anteriores de um ponto da imagem e retorna ao módulo de Traçado de Raios. Este procedimento se repete até que todos os pontos da imagem tenham sido gravados.

5.3 Fluxo de Execução

Esta seção tem por objetivo apresentar o fluxo de execução principal do protótipo implementado, visualizando-se os módulos anteriormente apresentados.

A execução do sistema dá-se conforme o algoritmo abaixo:

```
Programa Principal
     {
       Ler arquivo de entrada                      (1)
       Para cada linha                             (2)
          Para cada ponto                          (3)
             Determine o raio                      (4)
             Determime intersecção mais próxima    (5)
             Determine coloração no ponto de
                  intersecção                      (6)
          Fim_para
          Grava cor do ponto no arquivo de saída   (7)
       Fim_para
     }
```

Descrição do Fluxo:

1) É lido o arquivo de entrada de dados. As informações são armazenadas nas estruturas de dados.

2) A partir da resolução vertical (número de linhas) é feito um comando de repetição para passar por todos os pontos da linhas.

3) A partir da resolução horizontal (número de pontos) é feito um comando de repetição para traçar um raio para cada ponto.

4) É determinado matematicamente o raio com origem no centro de projeção e que passa pelo ponto (*pixel*) determinado acima.

5) A partir da descrição da cena são calculadas todas as intersecções do raio com os objetos na cena. Das intersecções ocorridas será retornada a mais próxima da câmera sintética. É importante frisar que nesta parte que são realizados todos os procedimentos relacionados aos campos de altitude descritos no capítulo 4.

6) A seguir são realizados todos os cálculos referentes ao modelo de iluminação utilizado, que determinam a cor no ponto de intersecção. Estes cálculos foram amplamente descritos no capítulo 3.

7) Por fim, os dados calculados contendo a cor para um ponto da imagem são gravados no arquivo de saída.

5.4 Entrada de Dados

A entrada de dados é feita através de um arquivo texto contendo uma descrição da cena a ser sintetizada. Para a definição da linguagem de descrição da cena foram levados em consideração os seguintes aspectos:

- Compatibilidade com o maior número possível de trabalhos realizados pelo grupo de computação gráfica da UFRGS. Este foi basicamente o fator decisivo na escolha da linguagem de descrição dos dados, visto que é prática comum no Instituto de Informática da UFRGS a realização de animações computadorizadas integrando os diversos trabalhos realizados pelos mestrandos;
- Compatibilidade com os sistemas de uso comum pela comunidade internacional de computação gráfica, uma vez que o trabalho foi enriquecido por inúmeras discussões e sugestões virtuais através da rede Internet com vários pesquisadores de outros países;
- Simplicidade no uso e flexibilidade na inclusão de novos recursos e/ou aprimoramentos.

Tendo em mente estes aspectos, foi implementada a mesma linguagem de descrição de cenas utilizada pelo *Pov-Team* (TESTA 1994), uma entidade de pesquisa em computação gráfica que fornece, em caráter de domínio público, um *parser* para uso em sistemas de traçado de raios. Visto que o sistema de uso geral pelo grupo de computação gráfica da UFRGS é compatível com o mesmo, assim como também diversos trabalhos no Instituto de Informática possuem entrada e saída de dados neste padrão achou-se conveniente o seu uso.

A descrição da cena é composta por basicamente por três partes:

- especificação do posicionamento da câmera sintética,
- especificação da localização das fontes de luz e
- descrição e localização dos objetos na cena.

Visto que a interface de entrada de dados é uma linguagem que parametriza as informações a serem manipuladas pelo sistema de traçado de raios, serão apresentadas a seguir a sintaxe e semântica para cada uma das partes descritas.

Sintaxe da Câmera Sintética

```
CAMERA {                                          (1)
     LOCATION <-18, 7.5, -28>                     (2)
     LOOK_AT <0.5, 1.07, 0.5>                     (3)
}
```

Semântica da Câmera Sintética

(1) Palavra reservada que indica o início da descrição da câmera sintética. A descrição propriamente dita deverá vir a seguir contida dentro de um par de chaves.

(2) Indica a posição no espaço tridimensional em que se encontra a câmera sintética.

(3) Este parâmetro indica o ponto para o qual a câmera está olhando.

Sintaxe da Fonte de Luz

```
LIGHT_SOURCE {  <-10, 12, -30>                    (1)
     COLOR RGB <1.0, 0.8, 0.8>                    (2)
}
```

Semântica da Fonte de Luz

(1) Indica a posição no espaço tridimensional em que se encontra a fonte luminosa.

(2) Este parâmetro indica a cor da luz nas suas três componentes: vermelho, verde e azul.

Sintaxe do Campo de Altitude

```
HEIGHT_FIELD {                                    (1)
     GIF "CANYON.GIF"                             (2)
     SCALE <1.0, 2.0, 1.0>                        (3)
     ROTATE <-20, 25, 0>                          (4)
     TRANSLATE <0, 0, 0>                          (5)
     COLOR RGB <1.0, 0.8, 0.8>                    (6)
     IMAGE_MAP                                    (7)
     WATER_LEVEL = 0.01                           (8)
}
```

Semântica do Campo de Altitude

(1) Palavra reservada que indica o início da descrição do campo de altitude. A descrição propriamente dita deverá vir a seguir contida dentro de um par de chaves.

(2) Este parâmetro indica o formato de arquivo utilizado como campo de altitude. Foi colocado para permitir que futuramente possam ser especificados arquivos *Targa* e/ou *Tiff*. Esta palavra é seguida pelo nome do arquivo de imagem entre aspas.

(3) Especificação de alterações na escala do campo de altitude. Destaca-se que o campo de altitude é sempre de dimensões 1x1x1 inicialmente.

(4) Indica uma rotação a ser aplicada no campo de altitude.

(5) Este parâmetro realiza a translação do campo de altitude, visto que ele é sempre considerado como inicialmente no centro do universo (0,0,0).

(6) Indicação da cor que será aplicada no campo de altitude.

(7) Parâmetro que indica que as cores contidas na *palette* devem ser levadas em consideração.

(8) Determinação do nível do mar aplicado ao campo de altitude.

5.5 Saída de Dados

O arquivo de saída do sistema de traçado de raios contém a imagem sintetizada, e é armazenado diretamente em disco. Isto se deve ao fato que o sistema produz uma imagem multiespectral (16 milhões de cores), não sendo possível a sua visualização na maioria dos equipamentos. Gerando-se um arquivo de saída em disco o sistema não fica restrito a nenhuma plataforma de *hardware*. A imagem é armazenada no padrão Targa não compactado, visto ser um dos formatos de imagens mais padronizados a nível mundial.

5.6 Conclusão

Este capítulo apresentou detalhes de implementação do sistema de traçado de raios desenvolvido para validar o modelo proposto neste trabalho. Embora existam outras primitivas (cilindros, cones, etc.) no sistema considerou-se desnecessário a especificação da sintaxe e semântica dos mesmos visto que não estão diretamente relacionados com o modelo proposto.

6 APRESENTAÇÃO E ANÁLISE DE RESULTADOS

"Tão somente esforça-te e tem mui bom ânimo...
não se aparte da tua boca o livro desta lei,
antes medita nele dia e noite...
então farás prosperar o teu caminho."
(Josué 1:7,8)

6.1 Introdução

Este capítulo tem por objetivo apresentar alguns resultados obtidos na síntese de campos de altitude visando à simulação visual de fenômenos naturais, através do método anteriormente descrito. Além disto, será realizada uma análise do algoritmo em termos de custo temporal visto ser este o maior obstáculo a ser vencido em sistemas de traçado de raios. Este capítulo divide-se em três etapas: na primeira serão analisados os resultados visuais quanto ao realismo da síntese de diversos fenômenos naturais, a saber:

- Síntese de Montanhas;
- Síntese de Água;
- Síntese de Madeira e
- Síntese de Fogo.

A segunda etapa visa apresentar uma análise de complexidade do algoritmo no que diz respeito ao tempo gasto pelo mesmo. E a terceira etapa tem por objetivo levantar alguns pontos de relativa importância sobre o algoritmo proposto.

6.2 Síntese de Fenômenos da Natureza com Campos de Altitude

A seguir apresentam-se alguns exemplos de imagens sintetizadas através de traçado de raios de campos de altitude conforme o método implementado neste trabalho. Em algumas imagens foi acrescida posteriormente ao processo de síntese outra imagem de fundo (nuvens, por exemplo) para aumentar o realismo da cena.

Para evitar o serrilhamento em algumas imagens, estas foram sintetizadas em resoluções maiores que a de exibição e após foi aplicado um processo de filtragem. Este processo de filtragem consiste na determinação de um novo *pixel* cuja cor será a média das cores dos seus vizinhos (FOLEY 1990). Embora este procedimento reduza pela metade o tamanho da imagem (em ambas as direções) ele melhora a imagem.

É importante salientar que procedimento de filtragem é realizado após a síntese da imagem, após o módulo de Geração de Arquivo de Saída (ver seção 5.2). Embora este pós-processamento não influencie a eficiência do algoritmo proposto ele é de significativa importância na visualização da imagem sintetizada, visto minimizar o serrilamento da mesma.

6.2.1 Síntese de Montanhas

Nesta secção serão mostradas diversas imagens de cenários baseados em campos de altitude reais e virtuais, utilizando as diversas abordagens (uma cor ou várias cores) propostas anteriormente.

6.2.1.1 Campos de Altitude Reais

A figura 6.1 mostra uma imagem GIF obtida de um arquivo DEM representando uma vista aérea de um lago denominado "Crater Lake" na região do Oregon, Estados Unidos. Esta imagem mostra apenas o solo da região, inclusive a parte que se encontra submersa pela água. A imagem possui as seguintes características: resolução de 512 X 512 e das 256 cores da *palette* apenas 108 estão presentes na imagem.

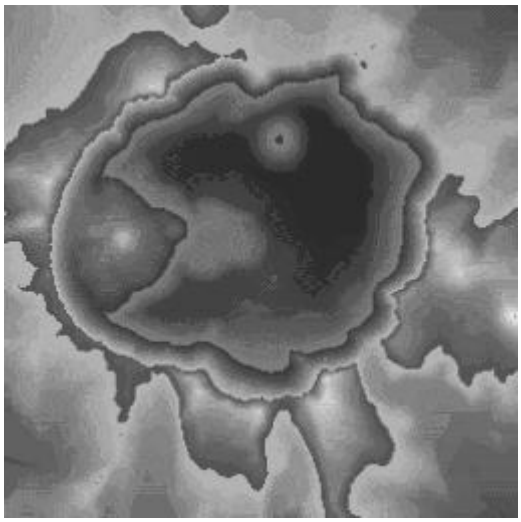


Figura 6.1 - "Crater Lake", Oregon, Estados Unidos.

Com base nesta imagem foi produzida a figura 6.2 que mostra tridimensionalmente o solo da região em questão. É importante salientar que o resultado não apresenta uma superfície suave como a que existe na realidade, isto se deve à pequena quantidade de altitudes (cores) diferentes existentes no campo de altitude, 108 ao todo, o que dá impressão da existência de degraus na superfície. Toda a região foi sintetizada numa resolução de 320 X 200 atribuindo-se uma única cor ao campo de altitude, e o tempo total de geração no equipamento indicado na secção 5.1 foi de 22 minutos e 26 segundos.

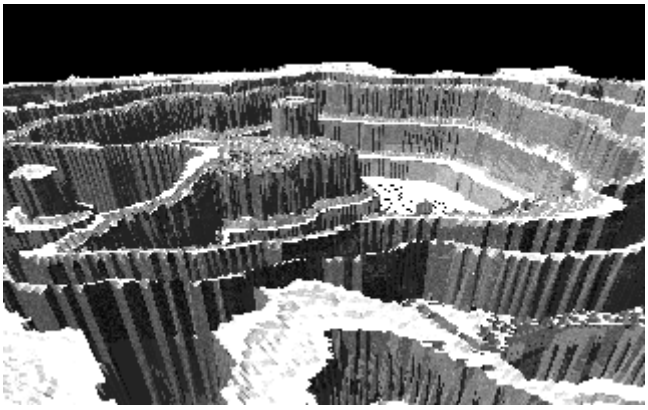


Figura 6.2 - Visão tridimensional do "Crater Lake"

A partir deste resultado podem ser levantadas algumas questões:

- O número total de triângulos virtuais na cena é de 524.288. São chamados *virtuais* porque eles não existem como uma entidade na cena, a não ser no momento em que são requeridos para o teste de intersecção raio-superfície. Não existe maneira fácil e compacta de armazenar esta quantidade de triângulos (com três coordenadas cada) num sistema geral de traçado de raios.
- O tempo gasto na geração da imagem foi de 22 minutos e 26 segundos. Este tempo é muito menor que o tempo necessário para sintetizar 524.288 triângulos, visto que foram gastos 10 minutos e 56 segundos na geração de outra imagem definida por apenas 12.168 triângulos. Ambas as imagens foram sintetizadas pelo mesmo sistema, equipamento e resolução.
- Os degraus surgidos na imagem reforçam a constatação anteriormente citada, que indicava que o resultado obtido depende em grande forma da imagem escolhida como campo de altitude. Neste caso seria interessante se realizar uma conversão de maior qualidade do arquivo DEM para a imagem GIF.

A figura 6.3 é outro campo de altitude de resolução 128 X 128 representando a região "*Pikes Peak*" no Colorado, Estados Unidos, e foi obtida a partir de um arquivo USGS. Nesta imagem foram utilizadas 150 cores da *palette*. É importante observar que as cores escolhidas foram dispostas na tabela de *palette* numa ordem bem específica, onde as mais escuras referenciam índices menores e as mais claras índices maiores; isto visa à simulação da coloração da região em questão.

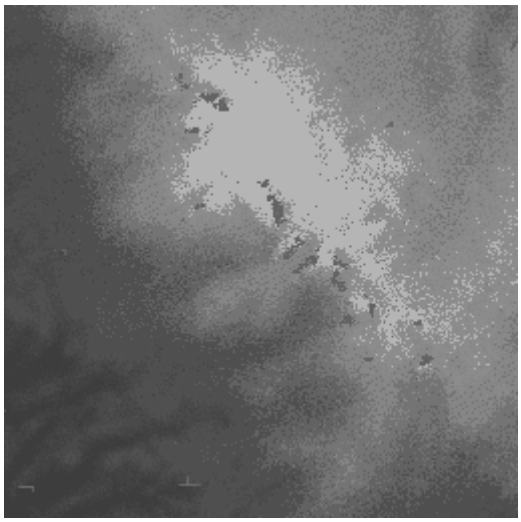


Figura 6.3 - Pikes Peak, Colorado.

As figuras 6.4 e 6.5 representam tridimensionalmente a região acima mencionada. A primeira delas foi sintetizada atribuindo-se apenas uma cor ao campo de altitude; já a segunda foi sintetizada levando-se em consideração as cores na tabela de *palette*.

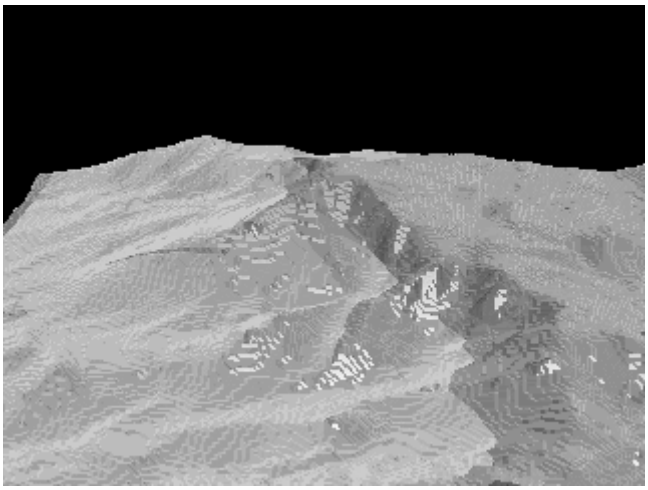


Figura 6.4 - "Pikes Peak" sintetizado sem levar em consideração as cores da "palette".

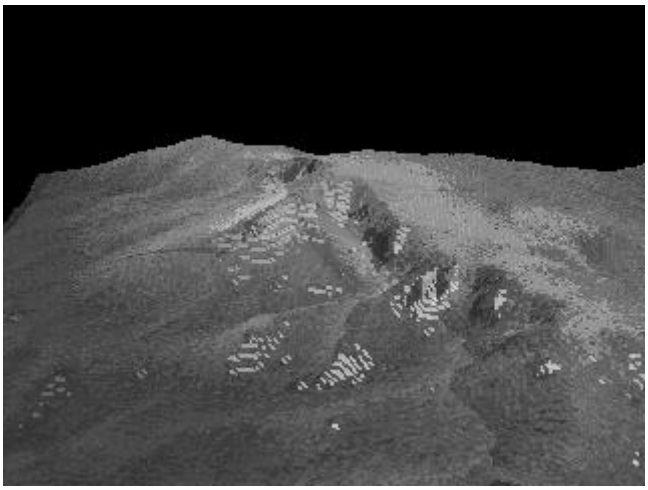


Figura 6.5 - "Pikes Peak" sintetizado considerando-se as cores na tabela de "palette".

Para se obter um resultado mais condizente com a realidade foi reduzida a altitude do campo de altitude, visando minimizar a altura dos possíveis degraus criados. Como pode ser observada, a imagem 6.5 possui um realismo muito maior que a imagem 6.4 devido à inclusão de diversas cores na superfície de acordo com a altitude da região.

Embora os resultados obtidos permitam a síntese de regiões com um grau de realismo aceitável, foi necessário um refinamento no processo de conversão dos dados volumétricos dos arquivos DEM e USGS para a imagem no formato GIF. Os programas de domínio público que realizavam tal conversão mostraram ser pouco eficientes, perdendo muitos valores de importância de um formato para outro. Por este motivo foi implementado um programa, na linguagem C, que realiza esta conversão com maior precisão, permitindo assim a síntese dos arquivos DEM e USGS com maior grau de realismo.

A figura 6.6 é outro exemplo de imagem sintetizada a partir de um campo de altitude real. Para a síntese desta imagem, foi realizada a conversão de um arquivo USGS para um arquivo GIF através do programa de conversão desenvolvido. Após a síntese da imagem (pós-processamento) foi acrescentada uma imagem de nuvens ao fundo, visando aumentar o realismo da mesma.

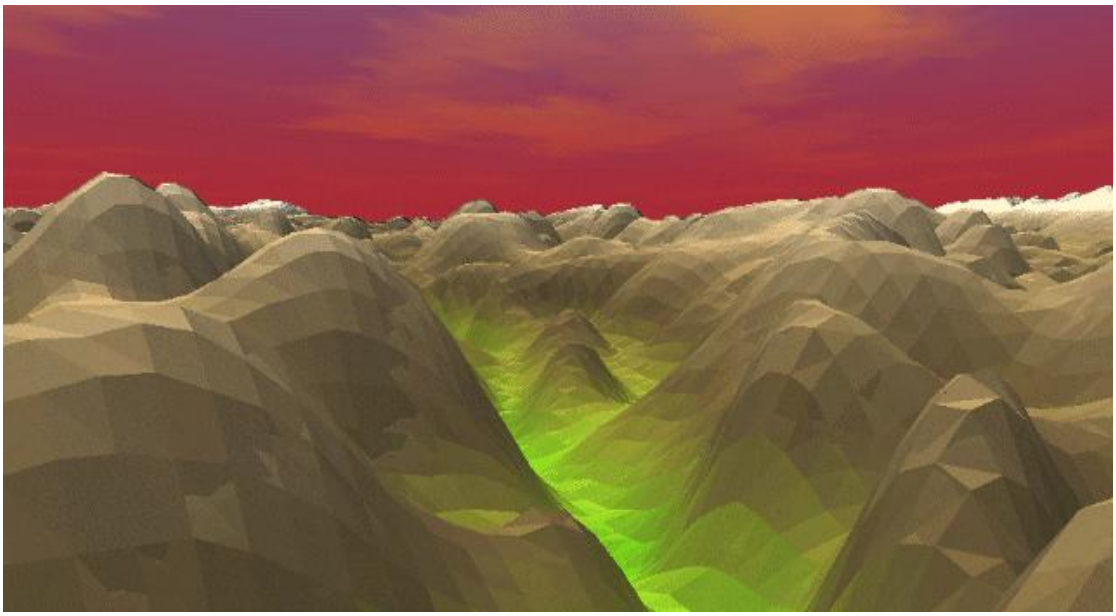


Figura 6.6 - Síntese da região leste do "Grand Canyon", Estados Unidos, usando campos de altitude.

6.2.1.2 Campos de Altitude Virtuais

Como pode ser observado na secção anterior, um dos principais problemas foi a falta de campos de altitude adequados para a síntese de montanhas. Por este motivo foram utilizados campos de altitude virtuais obtidos a partir de sistemas de processamento de imagens ou de algoritmos fractais.

A figura 6.7 mostra um campo de altitude produzido num sistema de processamento de imagens através de um procedimento fractal (conhecido como plasma). Esta imagem possui uma resolução de 512 X 512 e usa quase a totalidade das cores da *palette*. As cores na *palette* são diversas tonalidades de quatro básicas: azul, marrom, verde e branco; nesta ordem.

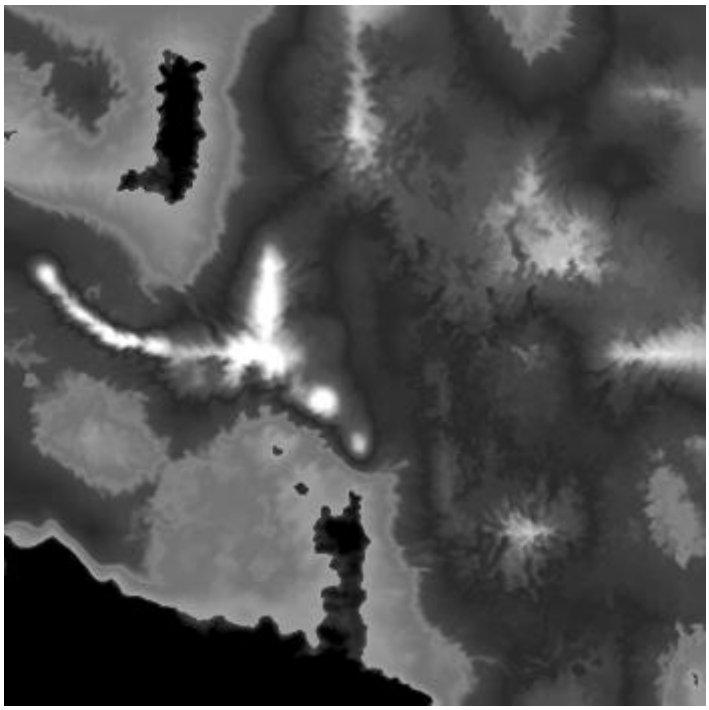


Figura 6.7 - Campo de altitude gerado por rotina fractal de plasma no sistema Adobe Photoshop.

Visando a síntese de uma região montanhosa próxima do mar, foi colocado um plano que intersecciona o campo de altitude no seu extremo inferior com um índice de reflexão alto, simulando água (e a reflexão da mesma). Para a síntese da imagem foram utilizados os parâmetros *Water_Level* para minimizar o número de intersecções e o parâmetro *Color_map* para colorizar o campo de altitude.

A figura 6.8 mostra o resultado do processo de síntese obtido após 1 hora, 34 minutos e 15 segundos. É importante salientar que o tempo de síntese sofreu um acréscimo significativo devido à inclusão de uma primitiva altamente reflexiva.

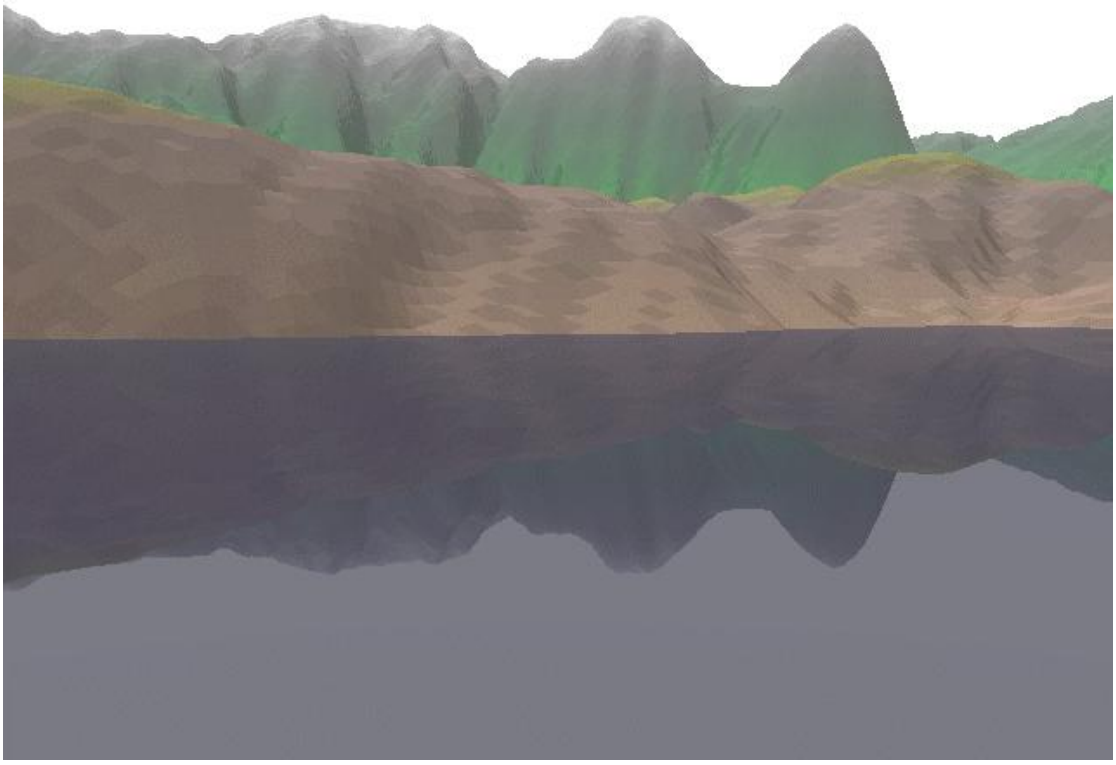


Figura 6.8 - Imagem sintetizada de uma montanha junto ao mar.

Como pode ser observado na figura 6.8 o método proposto por este trabalho permite a síntese de montanhas com um alto nível de realismo. Pode ser implementada uma série de outros recursos no sistema de traçado de raios para aumentar o grau de realismo das imagens. Por exemplo, é possível acrescentar uma perturbação do vetor normal ao plano que define a água, criando assim ondas sobre o mar, mas tais implementação não foram contempladas neste trabalho.

A figura 6.9 mostra outro campo de altitude de resolução 512 X 512 produzido num sistema de processamento de imagens. Os resultados obtidos são apresentados nas figuras 6.10 e 6.11. É importante salientar que a figura 6.11 é a figura 6.10 acrescida de uma imagem de nuvem no fundo. Os resultados obtidos são animadores.

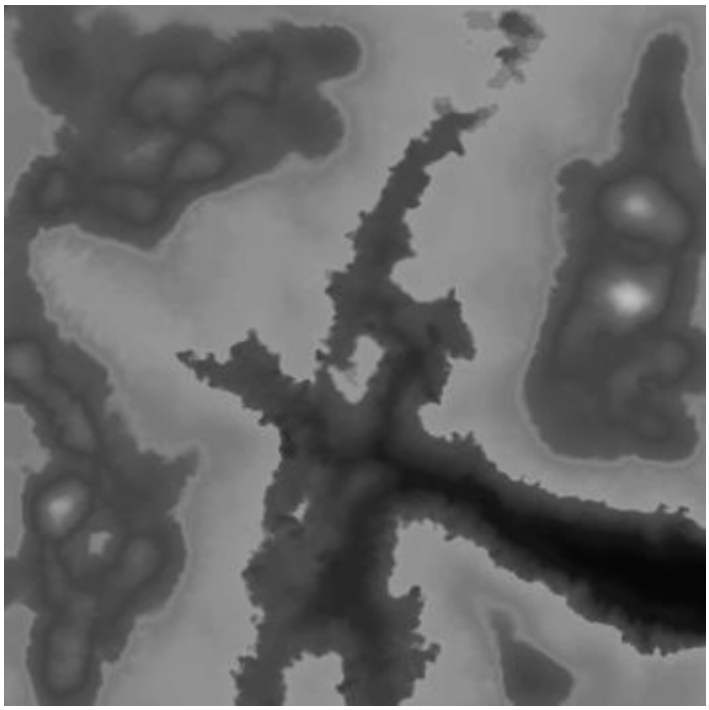


Figura 6.9 - Campo de altitude criado no sistema de processamento imagens Adobe Photoshop.

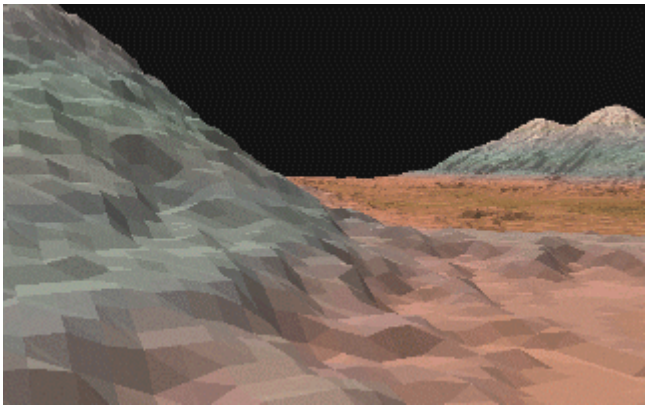


Figura 6.10 - Cenário de região montanhosa sintetizado utilizando campos de altitude virtuais.

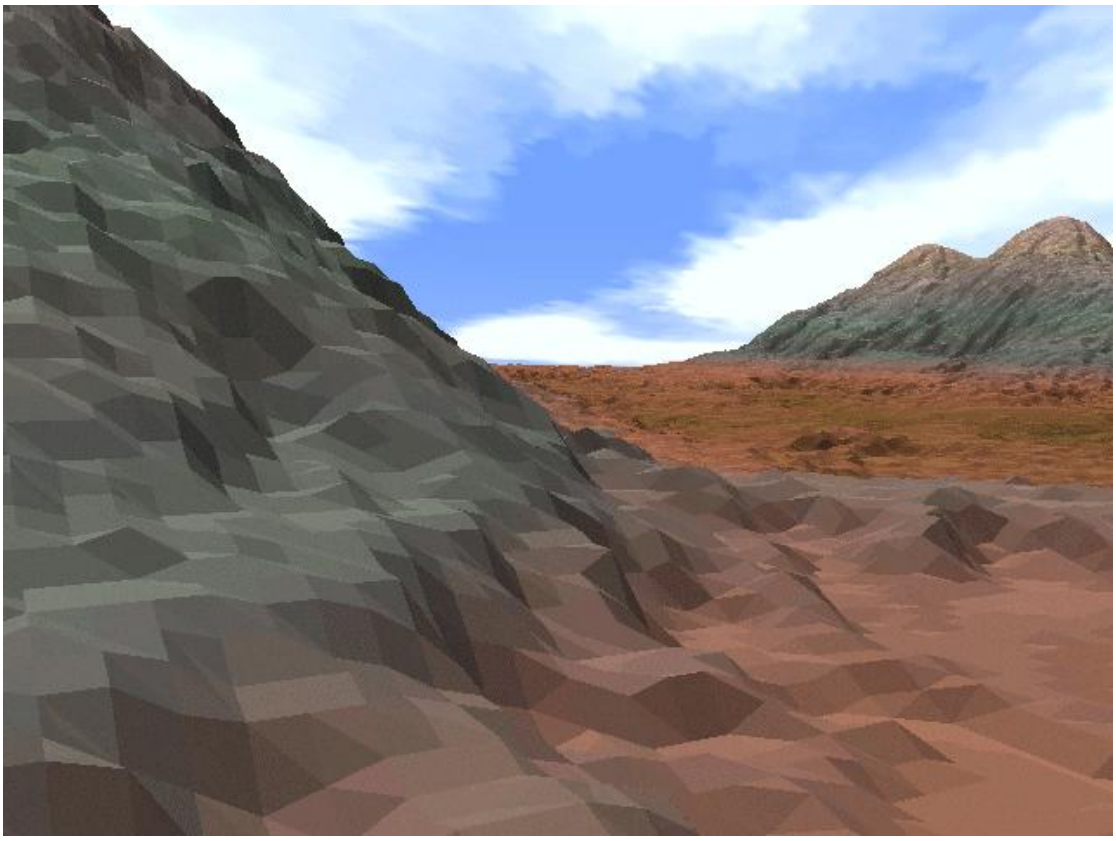


Figura 6.11 - Cenário sintetizado através do método proposto com imagem de fundo (nuvens) acrescentada.

O método proposto neste trabalho produz imagens que se assemelham em muito a cenários montanhosos encontrados na natureza.

A figura 6.12 mostra a síntese de uma ilha no oceano, com uma imagem de nuvem acrescentada no fundo. Para a síntese desta imagem foi utilizado um campo de altitude virtual.

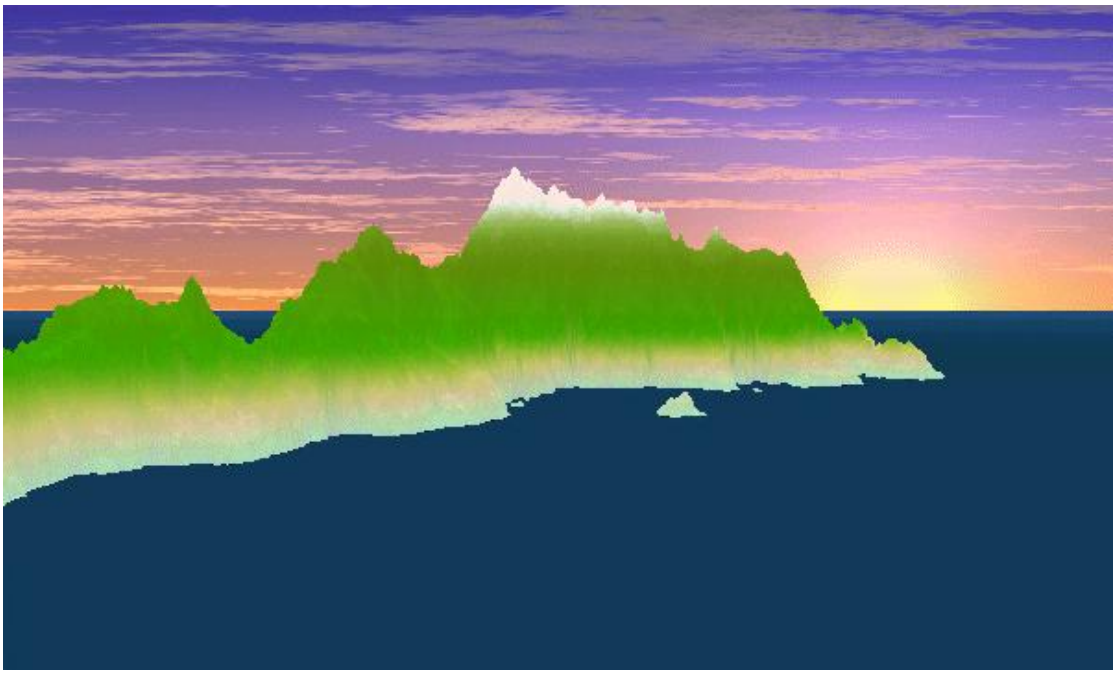


Figura 6.12 - Síntese de ilha no mar usando campos de altitude virtuais.

6.2.2 Síntese de Água

No sistema implementado é possível criar imagens que simulem água através do mesmo procedimento empregado para a síntese de montanhas.

As figuras 6.13 e 6.14 são duas imagem de uma animação produzida utilizando-se o protótipo implementado neste trabalho. A imagem representa um recipiente contendo água onde foram deixadas cair diversas gotas (ou pedras) produzindo o movimento circular da água.

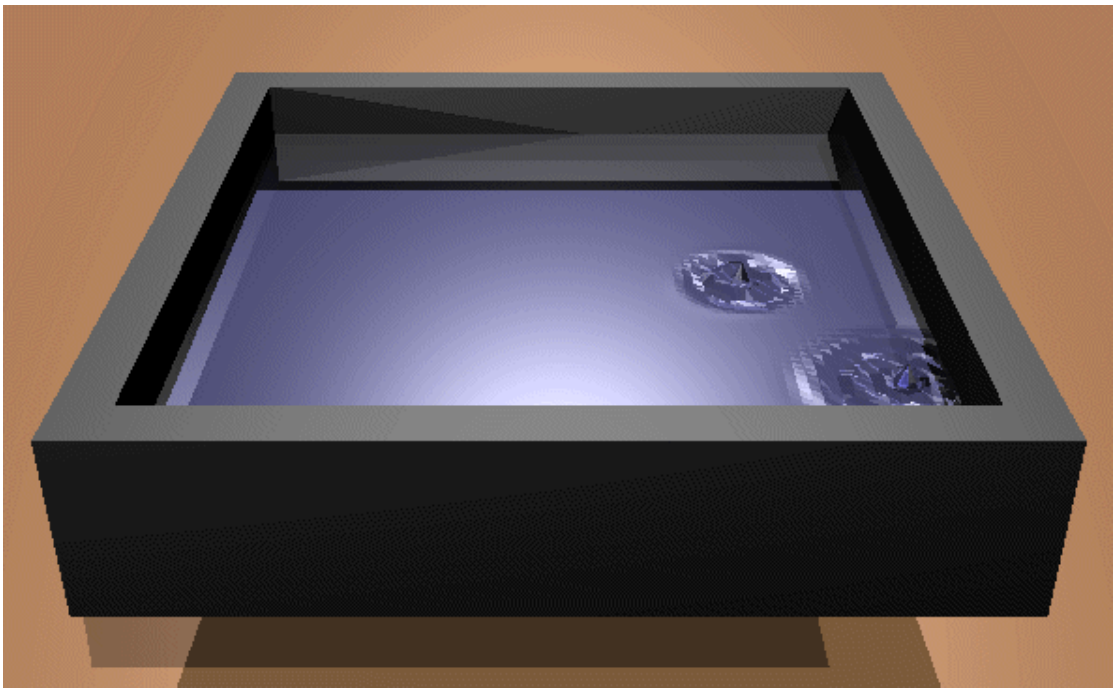


Figura 6.13 - Síntese de água (cena inicial de uma animação).

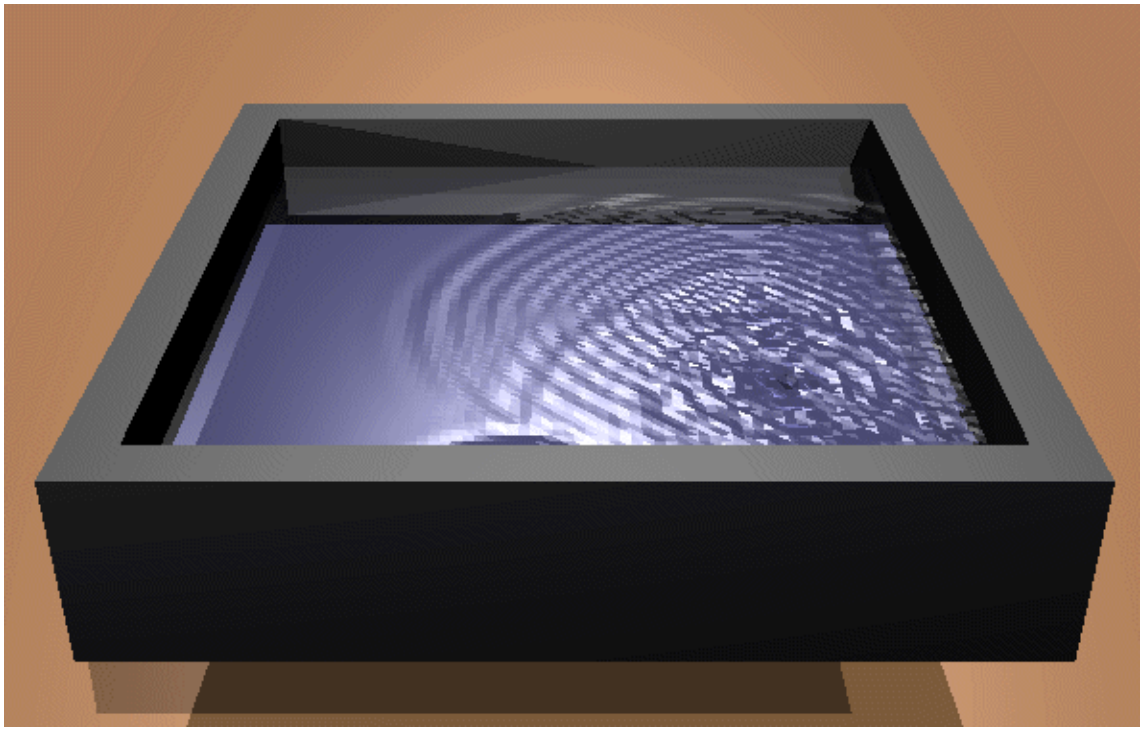


Figura 6.14 - Síntese de água.

Para a geração do(s) campo(s) de altitude(s) foi implementado um programa que desenha em posições aleatórias um conjunto de círculos concêntricos de várias cores (altitudes), gravando cada imagem da sequencia produzida para a animação. A figura 6.15 mostra diversas imagens (campos de altitude) em diferentes estágios de tempo gerados para a animação.

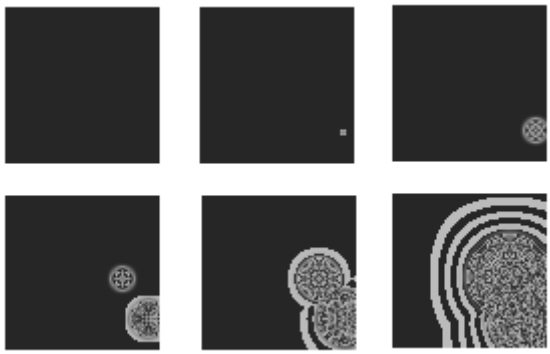


Figura 6.15 - Campos de altitudes para a animação de pingos caindo num recipiente com água.

É importante destacar a **versatilidade** do algoritmo proposto neste trabalho, pois através de uma única implementação podem ser sintetizadas tanto água como montanhas. Através da modificação do algoritmo de desenho dos campos de altitude é possível criar outros movimentos da água, como por exemplo, ondas do mar, etc.

6.2.3 Síntese de Madeira

A implementação de campo de altitude como primitiva num sistema de traçado de raios permite a síntese de objetos que são muito complexos se modelados através de outros métodos. Assim sendo, nas secções anteriores têm sido mostrado como com os campos de altitude é possível se modelar montanhas e água. Utilizando-se o mesmo procedimento será mostrado como é possível se sintetizar madeira.

A figura 6.16 mostra a síntese de um pedaço de madeira quebrada. Para a modelagem desta cena foram utilizadas apenas duas primitivas: um cilindro e um campo de altitude. O cilindro foi utilizado para a modelagem do pedaço da madeira e o *height*

field foi utilizado para modelar todas saliências e entrâncias que são produzidas ao se quebrar um pedaço de pau.

Para a criação do campo de altitude foi desenhado parte de uma imagem fractal dentro de um círculo e deixando o resto da imagem com a cor de índice zero. Utilizando-se o parâmetro *water_level* foi desprezada toda a região da imagem que estava fora do círculo.

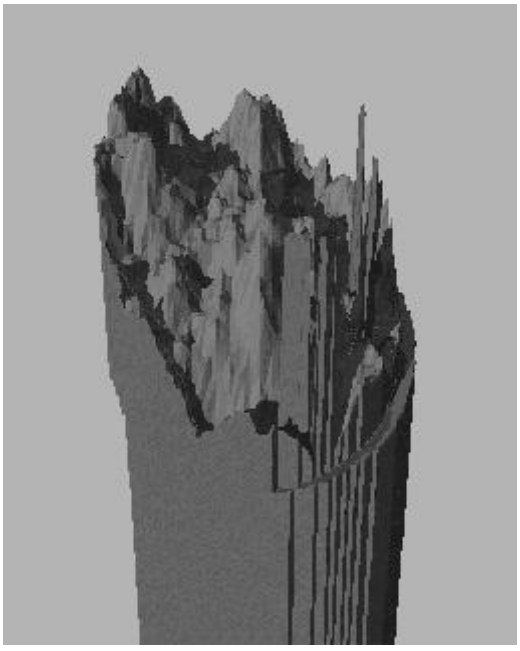


Figura 6.16 - Síntese de madeira quebrada.

6.2.4 Síntese de Fogo

A utilização de campos de altitude específicos pode permitir também a síntese de fogo.

Para este fim foi desenvolvido um programa que gera uma sequencia de imagens com a aparência de fogo. O algoritmo calcula, para cada ponto do vídeo, a média das cores dos pontos vizinhos, substituindo depois a cor do ponto em questão pela nova cor calculada. Este processo é realizado iniciando-se com os pontos da região inferior do vídeo até os pontos na região superior. É utilizada uma *palette* contendo

diversas tonalidades das seguintes cores: preto, azul, vermelho, amarelo e branco, nesta ordem. A figura 6.17 mostra uma das imagens geradas através deste procedimento.

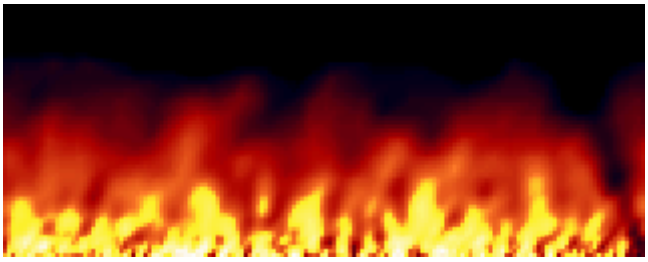


Figura 6.17 - Imagem bidimensional simulando fogo.

Para a síntese de fogo utilizando estas imagens como campos de altitude foram realizados os seguintes passos:

- Utilização do parâmetro *color_map* para considerar as cores na tabela de *palette*,
- redução da escala do eixo *Y* do campo de altitude, criando assim um objeto com forma de paralelepípedo ao invés de um cubo, e
- síntese da imagem com o observador localizado perpendicularmente à superfície.

A figura 6.18 mostra a imagem sintetizada através do processo descrito acima.

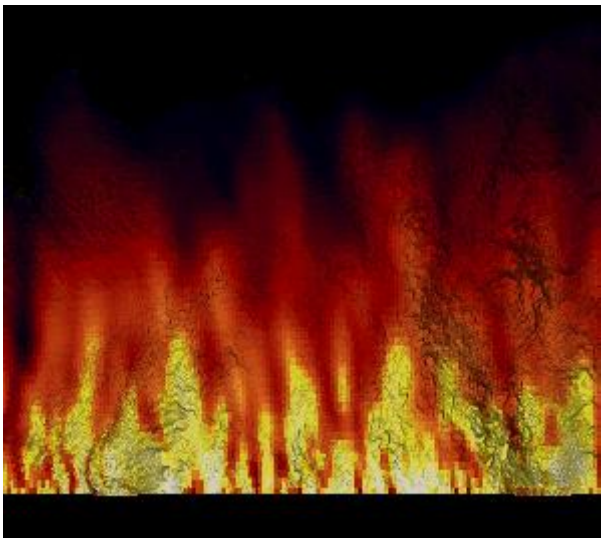


Figura 6.18 - Síntese de fogo utilizando-se campos de altitude.

O relevo e sombras ocasionados pela diferença das cores (índices da *palette*) embora não correspondam à forma física real do fogo, acrescenta realismo da imagem aparentando ser fumaça.

A figura 6.19 é outro exemplo dos resultados obtidos neste trabalho na síntese de fogo através de campos de altitude.

Figura 6.19 - Exemplo de síntese de fogo.

6.3 Complexidade do Algoritmo

Visto que este trabalho considera os campos de altitude como primitivas básicas do sistema de traçado de raios, o estudo da complexidade deter-se-á ao algoritmo de tratamento específico dos *height fields*.

Num campo de altitude de dimensões n_1 e n_2 o número total *C* de células é determinado por

$$C = n_1\, n_2$$

visto ser uma matriz bidimensional. Conforme foi dito anteriormente cada célula é dividida em duas primitivas básicas (triângulos). Desta forma o número total *T* de triângulos existentes no campo de altitude é

$$T = 2C = 2(n1\ n2) \qquad (1)$$

Considerando-se que o pior caso é aquele em que um raio atravessa o campo de altitude diagonalmente pode-se afirmar com base na geometria plana que o número máximo *M* de células interseccionadas será

$$M = \sqrt{n_1^{\,2} + n_2^{\,2}} \qquad (2)$$

e consequentemente que

$$S = 2\,M = 2\sqrt{n_1^{\,2} + n_2^{\,2}} \qquad (3)$$

onde S é o número de triângulos interseccionados por um raio que atravessa o campo de altitude diagonalmente.

Como foi explicado anteriormente somente são divididas em triângulos as células que são atravessadas pelo raio e consequentemente apenas serão calculadas as

intersecções dos triângulos referentes a tais células. Podemos determinar então que no pior caso o algoritmo proposto apresenta uma complexidade O $(\sqrt{N})$.

O algoritmo DDA implementado determina as células atravessadas pelo raio na ordem em que isto acontece. Assim é possível garantir que a primeira intersecção raio-triângulo determinada será a mais próxima e consequentemente pode-se evitar o cálculo das intersecções restantes. Isto minimiza ainda mais o número total de cálculos realizados, reduzindo para apenas **dois** cálculos de intersecção para um dado raio, no **melhor dos casos**. O melhor dos casos é aquele que o raio intersecciona um dos triângulos da primeira célula determinada pelo DDA.

Assim, por exemplo, numa cena de uma superfície montanhosa definida por um campo de altitude de resolução 512 X 512, perfazendo um total de 262.144 células, pode-se determinar o total de triângulos *T* através da fórmula (1)

$$\begin{aligned} T &= 2(n_1\ n_2) \\ &= 2(512\ \text{x}\ 512) \\ &= 2(262.144)\ = \mathbf{524.288} \end{aligned}$$

Conforme descrito acima, o total de triângulos interseccionados pelo raio, no **pior caso**, é determinado através da formula (3)

$$\begin{aligned} S &= 2.\sqrt{n_1^{\,2} + n_2^{\,2}} \\ &= 2.\sqrt{512^2 + 512^2} \\ &= 2.\sqrt{524.288} = \mathbf{1448} \end{aligned}$$

Desta forma, pode-se ver que embora o número total de triângulos seja em extremo elevado, apenas uma pequena parte dos mesmos serão testados para verificar se houve intersecção com um raio. No exemplo acima descrito, descrevendo o **pior caso**, o número de triângulos testados representa menos de 1% do total existente.

6.4 Considerações Importantes

É possível concluir, a partir das imagens sintetizadas pelo sistema, que em termos visuais o algoritmo implementado satisfaz além das expectativas os objetivos propostos, como também mostrou ser muito versátil, não se limitando à síntese de apenas um tipo de fenômeno, mas sim diversos (montanhas, ilhas, água, madeira, fogo, etc.).

O algoritmo aqui proposto, para a síntese de fenômenos naturais através do traçado de raios utilizando campos de altitude apresenta, pela sua própria natureza, diversas vantagens sobre a implementação convencional de traçado de raios. Tais vantagens são descritas a seguir.

6.4.1 Uso de Dados Bidimensionais

Diversos segmentos da sociedade, como medicina, topografia, informática, etc., utilizam conjuntos de dados expressos na forma de matrizes bidimensionais.

A medicina, por exemplo, utiliza amplamente imagens (matrizes bidimensionais) obtidas através de um procedimento radiológico, denominado tomografia, que consiste no cálculo da absorção de raios X de um corte do corpo humano com uma dada espessura.

Já a topografia moderna utiliza fotos digitalizadas obtidas por satélite ou aeronaves da superfície terrestre com os detalhes de seus elementos naturais ou artificiais.

Os dispositivos de *hardware* são baseados em tecnologias matriciais, e realizam a entrada e/ou saída de dados de forma bidimensional, dentre os quais podemos citar o vídeo, as impressoras, digitalizadores de vídeo, *scanners*, etc.

Livros, fotografias, revistas, imagens de satélites, e desenhos podem ser tomados como base para a obtenção de campos de altitudes através de processos de

digitalização. Assim, a síntese de campos de altitudes (conjuntos bidimensionais de dados), não apenas constitui um passo lógico para a Ciência da Computação como também amplia grandemente os horizontes e aplicações da Computação Gráfica.

6.4.2 Pré-processamento Implícito

O uso de campos de altitude traz diversas informações que embora não estejam explicitamente definidas, podem ser facilmente obtidas devido às características matriciais e bidimensionais dos dados.

A utilização de campos de altitude também permite a determinação das coordenadas (implícitas) de cada par de triângulos associados a uma célula, através das coordenadas do paralelepípedo que o envolve, minimizando assim o consumo de memória.

Já a ordenação matricial dos dados num campo de altitude consiste basicamente num pré-processamento do posicionamento espacial dos triângulos, associados a cada célula, considerando-se apenas os eixos X e Y. Em determinados casos este pré-processamento pode reduzir para dois o número de cálculos de intersecção do raio com o campo de altitude, vide seção 6.3.

O uso de valores de altitude associados a cada célula constitui também um pré-processamento que permite a verificação da intersecção ou não de um raio com os triângulos associados a uma célula sem ser necessário nenhum tipo de cálculo matemático, mas bastando apenas um conjunto de comparações, conforme descrito nas seções 4.3.1.2 e 4.3.1.4.5.

6.4.3 Eficiência na Síntese

A simplicidade do algoritmo de traçado de raios deve-se quase exclusivamente à sua dependência de duas únicas operações: o traçado de um raio (reta) para cada *pixel* do vídeo e o cálculo do ponto de intersecção do raio com as primitivas contidas na cena. Segundo Whitted (WHITTED 1980) o tempo gasto no cálculo das intersecções em cenas complexas chega a superar os 95% do tempo total do algoritmo.

A síntese de uma superfície definida por centenas ou milhares de triângulos em sistemas de traçado de raios representa a realização de centenas ou milhares de testes de interseção por raio. Um dos métodos sugeridos para minimizar o número de intersecções é a inclusão dos triângulos que definem a superfície dentro de outra primitiva que servirá de envelope (*Bounding Volume*) (GLASSNER 1991). Assim, somente serão testadas as intersecções com o conjunto de triângulos se o raio interseccionar primeiramente o envelope da superfície. Contudo, esta solução não minimiza o número de cálculos realizados uma vez que um raio intersecciona o envelope.

O uso de campos de altitude na síntese de superfícies definidas por triângulos apresenta uma abordagem simular a dos *bounding volumes*. Os cálculos de intersecção raio-triângulos somente serão testados se o raio primeiramente atingir o paralelepípedo que envolve o campo de altitude. Contudo, devido ao uso de um algoritmo de DDA para a determinação dos triângulos pelos quais o raio atravessa a quantidade de intersecções é amplamente minimizada, conforme demonstrado na seção 6.3.

Através do método proposto neste trabalho é possível a visualização tridimensional dos campos de altitudes a partir dos mais diversos pontos de observação (LOOK_AT), permitindo assim que detalhes específicos possam ser estudados e destacados.

Desta forma, o algoritmo proposto neste trabalho apresenta uma abordagem mais eficiente na determinação dos triângulos de uma superfície atravessados por um raio.

6.4.4 Baixo Consumo de Recursos

Nos sistemas de traçado de raios genéricos diversas primitivas são armazenadas através dos vértices que as definem. Assim, por exemplo, para cada triângulo é necessário armazenar os seus três vértices, sendo que cada um destes é definido por três pontos no espaço (X, Y, Z). Desta forma, para cada triângulo definido no espaço são necessários 27 dados. Se considerarmos que cada triângulo possui

também características próprias, tais como a sua cor, o número de dados associado a cada primitiva cresce rapidamente.

Assim, uma superfície definida por triângulos, como uma região montanhosa, facilmente pode requerer um consumo de dados enorme, tornando-a praticamente impossível de sintetizar. Quanto maior a superfície, ou quanto mais detalhada ela for, tanto mais aumentará o consumo de recursos computacionais.

A utilização de campos de altitude, na definição de superfícies definidas por triângulos, representa uma considerável economia de recursos computacionais, visto que são armazenados apenas um número de dados correspondente ao total de pontos definidos pela resolução do campo de altitude. Os triângulos, assim como os seus vértices são calculados quando necessários, evitando assim que seja necessário armazenar todas as informações referentes aos mesmos (coordenadas dos vértices).

A utilização de imagens GIF conforme descrito no item 4.3.2, traz também suas vantagens, visto ser um arquivo que contém apenas 256 índices que referenciam cores. Cada célula possui um índice que referência a sua altitude e simultaneamente a sua cor, armazenado em um único valor duas informações distintas, e desta forma economizando mais memória.

6.4 Conclusão

Este capítulo apresentou os resultados obtidos na implementação de um sistema de traçado de raios utilizando campos de altitude na síntese de fenômenos naturais. Observou-se que o algoritmo produziu não apenas imagens com alto grau de realismo, mas também mostrou ser muito versátil, podendo sintetizar diversos fenômenos.

Foi realizada uma análise da complexidade computacional do algoritmo onde é possível perceber que esta proposta possui diversas vantagens que a tornam muito atrativa na simulação de fenômenos naturais em sistemas de traçado de raios.

7 CONCLUSÕES

"Pois comerás do trabalho das tuas mãos:
feliz serás, e te irá bem"
(Salmo 128:2)

Este trabalho teve por objetivo principal o desenvolvimento de um modelo que permitisse a síntese de fenômenos naturais num sistema de traçado de raios com baixo custo computacional, no que se refere a modelagem, síntese e animação. Este modelo teve o comprometimento com a qualidade visual das imagens sintetizadas, sem contudo ter um compromisso rigoroso com todos os aspectos e detalhes que envolvem os fenômenos naturais.

A metodologia utilizada na definição e elaboração deste trabalho apresentou de forma cíclica e contínua as seguintes fases:

- levantamento do problema;
- avaliação das possíveis soluções e
- implementação das soluções no modelo.

A validação do trabalho se deu através da implementação computacional de um sistema de traçado de raios conforme as características do modelo proposto.

Com base na implementação realizada e nos resultados e considerações apresentadas no capítulo 6 é possível concluir que:

- É de muita importância a síntese de imagens baseada em campos de altitude, visto que grande parte dos dados armazenados pelo homem encontram-se dispostos na forma de matrizes bidimensionais, as imagens de satélite por exemplo (secção 6.4.1), e provavelmente continuarão a ser utilizados durante muito tempo;

- O modelo proposto permite a síntese de fenômenos naturais com um alto grau de realismo, conforme demonstram as diversas imagens sintetizadas pelo mesmo;
- O algoritmo apresentado é de fácil implementação, utilizando procedimentos matemáticos comuns a sistemas de traçado de raios (intersecções de raios com plano, triângulos e paralelepípedos), algoritmos simples e conhecidos (DDA), operações matriciais e comparações; conforme visto no capítulo 4;
- O uso de campos de altitude representa uma forma simples, rápida e eficiente para a síntese de superfícies definidas por triângulos, permitindo assim que fenômenos naturais possam ser sintetizados em equipamentos de baixo poder computacional, conforme demonstrado nas secções 6.4.3 e 6.4.4. Salienta-se neste ponto que a síntese de fenômenos naturais no equipamento utilizado para este trabalho só foi possível através da abordagem proposta, sendo inviável através de um sistema convencional de traçado de raios, devido às características de arquitetura e memória que o *hardware* usado apresenta;
- Os campos de altitude possuem uma organização espacial dos dados, implícita na sua estrutura (seção 6.4.2), que permitem:
    1. a criação de *height fields* virtuais, em sistemas de processamento de imagens, visando propósitos específicos, devido ao uso da *palette* da imagem como diferentes níveis de altitude, e
    2. um processo de síntese de imagens mais rápido através a utilização de diversas técnicas;
- O grau de realismo depende da imagem utilizada como campo de altitude (seção 6.2.1.1 e 6.2.1.2). Já a síntese de fenômenos e superfícies é restrita apenas pela criatividade do usuário.

Uma das sérias dificuldades encontradas neste trabalho foi a escassez de material bibliográfico sobre a síntese de fenômenos naturais, e quando existente, predominava a falta de clareza nos trabalhos apresentados. Salienta-se também que a totalidade do material bibliográfico sobre o assunto apresenta apenas os resultados e procedimentos considerados "positivos" e "bem sucedidos", não mencionando as abordagens pouco eficientes e o porquê da ineficiência das mesmas; forçando assim que

os novos pesquisadores incorram desnecessariamente em erros e caminhos pouco proveitosos, já trilhados por outros.

Este algoritmo apresenta como deficiência a falta de um gerenciamento eficiente do consumo de memória requerida, proporcional à resolução das imagens utilizadas como campos de altitude. Para resolver este problema seria interessante a implementação de uma estrutura de dados hierárquica para armazenar as imagens bidimensionais.

Outra dificuldade encontrada foi na análise dos resultados no que diz respeito ao realismo visual da imagem, tomando-se muitas vezes como ponto de referência a simples validação visual da mesma, o que não deixa de ser uma avaliação subjetiva.

7.1 Futuro

Existem contudo diversos problemas a serem resolvidos e possíveis refinamentos que o modelo pode sofrer. Considera-se que estes podem ser alvo de futuras pesquisas. As melhorias sugeridas são as seguintes:

- a utilização de outro formato de arquivo de definição dos campos de altitude, com *palette* superior a 256 cores. Sugere-se a utilização de arquivos *Targa* (TGA) de 2 bytes (65 mil cores). Isto permitirá uma definição maior na determinação da superfície;
- a implementação de mapeamento de texturas sobre a superfície definida pelos campos de altitude, permitindo assim que a cor da superfície possa possuir qualquer cor em qualquer posição. O modelo atual permite apenas a coloração da superfície de acordo com a altitude da mesma e das cores disponíveis na *palette*;
- extensão do algoritmo DDA de travessia do campo de altitude (matriz bidimensional) para atravessar uma matriz tridimensional através de um algoritmo de 3DDDA similar ao proposto por Fujimoto (FUJIMOTO 1986). A matriz pode ser armazenada numa *octree*. Isto permitirá a síntese de dados volumétricos tais como imagens de

tomógrafos e/ou sistemas de partículas tridimensionais. O sistema calculará, desta forma, somente as intersecções dos objetos (cubos, esferas ou qualquer outra primitiva) selecionados pelo algoritmo de 3DDDA, isto com um custo computacional reduzido.

Este trabalho tornou possível a síntese de campos de altitudes em sistemas de traçado de raios de forma rápida e com um baixo custo computacional. Isto permite que sejam visualizadas imagens de fenômenos naturais com o alto grau de realismo característico dos sistemas de *Ray Tracing*. Considerando os equipamentos de baixo poder computacional utilizados destaca-se que os resultados obtidos superaram em muito as expectativas.

Inicialmente, imaginou-se que o uso de campos de altitude permitiria apenas a síntese de montanhas, porém ao longo do trabalho pôde ser constatado que a variedade de objetos que podem ser modelados através deste método está limitada basicamente pela criatividade e interesse do usuário. Outro fator que merece destaque é a facilidade que o modelo traz implicitamente para a geração de animações, por permitir o uso de imagens bidimensionais como dados básicos para a síntese.

Apesar das diversas dificuldades encontradas, acredita-se que os objetivos de definição e construção de uma ferramenta de computação gráfica, que tratasse da síntese de fenômenos naturais através do traçado de raios utilizando campos de altitude, foram alcançados. Além disto, acredita-se que este é um dos caminhos rumo a uma maior integração da Computação Gráfica com outras ciências.

ANEXO A-1 LISTAGENS

Levando em consideração que os procedimentos do cálculo de intersecção de um raio com outra primitiva geométrica (plano, triângulo, esfera, cone e cilindro) têm sido amplamente descritos na bibliografia sobre o assunto, achou-se conveniente colocar apenas a listagens dos algoritmos utilizados para tais cálculos.

Listagem do algoritmo para o cálculo de intersecção de um raio com um plano.

```
/*************************************************************
*
*  Esta rotina calcula a intersecção de um raio com um plano.
*
*************************************************************/


int Inter_Plano (Raio, Plano, Depth)
RAY *Raio;
PLANE *Plano;
double *Depth;
 {
 double Normal_O, Normal_Dir;

 Teste_Plano++;
 if (Raio == CM_Raio)
  {
  VDot (Normal_Dir, Plano->Vetor_Normal, Raio->Direcao);
  if ((Normal_Dir < Delta_plano) &&
   (Normal_Dir > -Delta_plano))
   return (FALSE);

  if (!Plano->CM_ok)
   {
   VDot (Plano->CMNorm_O, Plano->Vetor_Normal, Raio->Initial);
   Plano->CMNorm_O += Plano->Dist;
   Plano->CMNorm_O *= -1.0;
   Plano->CM_ok = TRUE;
   }

  *Depth = Plano->CMNorm_O / Normal_Dir;
  if ((*Depth >= Delta_plano) && (*Depth <= Max_Dist))
   {
   Achou_Plano++;
```

```
   return (TRUE);
   }
  else
   return (FALSE);
  }
 else
  {
  VDot (Normal_Dir, Plano->Vetor_Normal, Raio->Direcao);
  if ((Normal_Dir < Delta_plano) &&
   (Normal_Dir > -Delta_plano))
   return (FALSE);

  VDot (Normal_O, Plano->Vetor_Normal, Raio->Initial);
  Normal_O += Plano->Dist;
  Normal_O *= -1.0;

  *Depth = Normal_O / Normal_Dir;
  if ((*Depth >= Delta_plano) && (*Depth <= Max_Dist))
   {
   Achou_Plano++;
   return (TRUE);
   }
  else
   return (FALSE);
  }
 }
```

Listagem do algoritmo para o cálculo de intersecção de um raio com um triângulo.

```
/*************************************************************
*
*  Esta rotina calcula a intersecção de um raio com um
*  triângulo
*
*************************************************************/


int Inter_Tri (Raio, Triagulo, Depth)
RAY *Raio;
TRIANGLE *Triangulo;
double *Depth;
 {
 double Normal_O, Normal_Dir;
 double s, t;

 Teste_Tri++;
 if(Triangulo->Flag_T)
  return(FALSE);

 if (Raio == CM_Raio)
  {
  if (!Triangulo->CM_ok)
   {
   VDot (Triangulo->CMNorm_O, Triangulo->Vetor_Normal, Raio->Inicial);
   Triangulo->CMNorm_O += Triangulo->Dist;
   Triangulo->CMNorm_O *= -1.0;
   Triangulo->CM_ok = TRUE;
   }

  VDot (Normal_Dir, Triangulo->Vetor_Normal, Raio->Direcao);
  if ((Normal_Dir < Min_Delta) &&
   (Normal_Dir > -Min_Delta))
   return (FALSE);

  *Depth = Triangulo->CMNorm_O / Normal_Dir;
  }
 else
  {
  VDot (Normal_O, Triangulo->Vetor_Normal, Raio->Inicial);
  Normal_O += Triangulo->Dist;
  Normal_O *= -1.0;

  VDot (Normal_Dir, Triangulo->Vetor_Normal, Raio->Direcao);
  if ((Normal_Dir < Min_Delta) &&
   (Normal_Dir > -Min_Delta))
   return (FALSE);

  *Depth = Normal_O / Normal_Dir;
```

```
  }

if ((*Depth < Min_Delta) || (*Depth > Max_Dist))
  return (FALSE);

switch (Triangulo->Eixo_Dom)
{
case X_Eixo:
  s = Raio->Inicial.y + *Depth * Raio->Direcao.y;
  t = Raio->Inicial.z + *Depth * Raio->Direcao.z;

  if ((Triangulo->P2.y - s)*(Triangulo->P2.z - Triangulo->P1.z) <
    (Triangulo->P2.z - t)*(Triangulo->P2.y - Triangulo->P1.y))
    return (FALSE);

  if ((Triangulo->P3.y - s)*(Triangulo->P3.z - Triangulo->P2.z) <
    (Triangulo->P3.z - t)*(Triangulo->P3.y - Triangulo->P2.y))
    return (FALSE);

  if ((Triangulo->P1.y - s)*(Triangulo->P1.z - Triangulo->P3.z) <
    (Triangulo->P1.z - t)*(Triangulo->P1.y - Triangulo->P3.y))
    return (FALSE);

  Achou_Tri++;
  return (TRUE);

case Y_Eixo:
  s = Raio->Inicial.x + *Depth * Raio->Direcao.x;
  t = Raio->Inicial.z + *Depth * Raio->Direcao.z;

  if ((Triangulo->P2.x - s)*(Triangulo->P2.z - Triangulo->P1.z) <
    (Triangulo->P2.z - t)*(Triangulo->P2.x - Triangulo->P1.x))
    return (FALSE);

  if ((Triangulo->P3.x - s)*(Triangulo->P3.z - Triangulo->P2.z) <
    (Triangulo->P3.z - t)*(Triangulo->P3.x - Triangulo->P2.x))
    return (FALSE);

  if ((Triangulo->P1.x - s)*(Triangulo->P1.z - Triangulo->P3.z) <
    (Triangulo->P1.z - t)*(Triangulo->P1.x - Triangulo->P3.x))
    return (FALSE);

  Achou_Tri++;
  return (TRUE);

case Z_Eixo:
  s = Raio->Inicial.x + *Depth * Raio->Direcao.x;
  t = Raio->Inicial.y + *Depth * Raio->Direcao.y;

  if ((Triangulo->P2.x - s)*(Triangulo->P2.y - Triangulo->P1.y) <
    (Triangulo->P2.y - t)*(Triangulo->P2.x - Triangulo->P1.x))
    return (FALSE);

  if ((Triangulo->P3.x - s)*(Triangulo->P3.y - Triangulo->P2.y) <
```

```
   (Triangulo->P3.y - t)*(Triangulo->P3.x - Triangulo->P2.x))
   return (FALSE);

  if ((Triangulo->P1.x - s)*(Triangulo->P1.y - Triangulo->P3.y) <
   (Triangulo->P1.y - t)*(Triangulo->P1.x - Triangulo->P3.x))
   return (FALSE);

  Achou_Tri++;
  return (TRUE);
}
return (FALSE);
}
```

Listagem do algoritmo para o cálculo de intersecção de um raio com um paralelepípedo.

```
/*************************************************************
*
*  Esta rotina calcula a intersecção de um raio com um
*  paralelepípedo.
*
*************************************************************/


int Inter_Box (Raio, box, Depth1, Depth2)
RAY *Raio;
BOX *box;
double *Depth1, *Depth2;
 {
 double t, tmin, tmax;
 VECTOR P, D;

 Teste_Box++;

 /* Transform the point into the boxes space */
 if (box->Trans != NULL)
  {
  MInvTransPonto(&P, &Raio->Inicial, box->Trans);
  MInvTransDir(&D, &Raio->Direcao, box->Trans);
  }
 else
  {
  P.x = Raio->Inicial.x;
  P.y = Raio->Inicial.y;
  P.z = Raio->Inicial.z;
  D.x = Raio->Direcao.x;
  D.y = Raio->Direcao.y;
  D.z = Raio->Direcao.z;
  }

  tmin = 0.0;
 tmax = MAXIMO;

 /* Verifica faces laterais */
 if (D.x < -EPSILON)
  {
  t = (box->bounds[0].x - P.x) / D.x;
  if (t < tmin)
   return 0;
  if (t <= tmax)
   tmax = t;
  t = (box->bounds[1].x - P.x) / D.x;
  if (t >= tmin)
   {
   if (t > tmax)
```

```
return 0;
     tmin = t;
     }
   }
  else if (D.x > EPSILON)
   {
   t = (box->bounds[1].x - P.x) / D.x;
   if (t < tmin)
    return 0;
   if (t <= tmax)
    tmax = t;
   t = (box->bounds[0].x - P.x) / D.x;
   if (t >= tmin)
    {
    if (t > tmax)
return 0;
     tmin = t;
     }
   }
  else if (P.x < box->bounds[0].x || P.x > box->bounds[1].x)
   return 0;

  /* Verifica face superior/inferior */
  if (D.y < -EPSILON)
   {
   t = (box->bounds[0].y - P.y) / D.y;
   if (t < tmin)
    return 0;
   if (t <= tmax)
    tmax = t;
   t = (box->bounds[1].y - P.y) / D.y;
   if (t >= tmin)
    {
    if (t > tmax)
return 0;
     tmin = t;
     }
   }
  else if (D.y > EPSILON)
   {
   t = (box->bounds[1].y - P.y) / D.y;
   if (t < tmin)
    return 0;
   if (t <= tmax)
    tmax = t;
   t = (box->bounds[0].y - P.y) / D.y;
   if (t >= tmin)
    {
    if (t > tmax)
return 0;
     tmin = t;
     }
   }
  else if (P.y < box->bounds[0].y || P.y > box->bounds[1].y)
```

```
  return 0;

 /* Verifica face anterior/posterior */
 if (D.z < -EPSILON)
  {
  t = (box->bounds[0].z - P.z) / D.z;
  if (t < tmin)
   return 0;
  if (t <= tmax)
   tmax = t;
  t = (box->bounds[1].z - P.z) / D.z;
  if (t >= tmin)
   {
   if (t > tmax)
return 0;
   tmin = t;
   }
  }
 else if (D.z > EPSILON)
  {
  t = (box->bounds[1].z - P.z) / D.z;
  if (t < tmin)
   return 0;
  if (t <= tmax)
   tmax = t;
  t = (box->bounds[0].z - P.z) / D.z;
  if (t >= tmin)
   {
   if (t > tmax)
return 0;
   tmin = t;
   }
  }
 else if (P.z < box->bounds[0].z || P.z > box->bounds[1].z)
  return 0;

 *Depth1 = tmin;
 *Depth2 = tmax;


 if ((*Depth1 < Min_Delta) || (*Depth1 > Max_Dist))
  if ((*Depth2 < Min_Delta) || (*Depth2 > Max_Dist))
   return (FALSE);
  else
   *Depth1 = *Depth2;
 else
  if ((*Depth2 < Min_Delta) || (*Depth2 > Max_Dist))
   *Depth2 = *Depth1;

 Achou_Box++;
 return (TRUE);
 }
```

Listagem do algoritmo para o cálculo de intersecção de um raio com uma esfera.

```
/*************************************************************
*
*  Esta rotina calcula a intersecção de um raio com uma
*  esfera.
*
*************************************************************/


int Inter_Esfera (Raio, Esfera, Depth1, Depth2)
RAY *Raio;
SPHERE *Esfera;
double *Depth1, *Depth2;
 {
 VECTOR Ori_Centro;
 double OCQuad, Prox_E, Corda, Corda_Quad;
 short interior;

 Teste_Esfera++;
 if (Raio == CM_Raio)
  {
  if (!Esfera->CM_ok)
   {
   VSub (Esfera->CMOtoC, Esfera->Centro, Raio->Inicial);
   VDot (Esfera->CMOCQuad, Esfera->CMOtoC, Esfera->CMOtoC);
   Esfera->CMinterior = (Esfera->CMOCQuad < Esfera->Raio_Quad);
   Esfera->CM_ok = TRUE;
   }
  VDot (Prox_E, Esfera->CMOtoC, Raio->Direcao);
  if (!Esfera->CMinterior && (Prox_E < Delta_Esfera))
   return (FALSE);
  Corda_Quad = Esfera->Raio_Quad - Esfera->CMOCQuad +
  (Prox_E * Prox_E);
  }
 else
  {
  VSub (Ori_Centro, Esfera->Centro, Raio->Inicial);
  VDot (OCQuad, Ori_Centro, Ori_Centro);
  interior = (OCQuad < Esfera->Raio_Quad);
  VDot (Prox_E, Ori_Centro, Raio->Direcao);
  if (!interior && (Prox_E < Delta_Esfera))
   return (FALSE);

  Corda_Quad = Esfera->Raio_Quad - OCQuad +
  (Prox_E * Prox_E);
  }

 if (Corda_Quad < Delta_Esfera)
  return (FALSE);
```

```
Corda = sqrt (Corda_Quad);
*Depth1 = Prox_E + Corda;
*Depth2 = Prox_E - Corda;

if ((*Depth1 < Delta_Esfera) || (*Depth1 > Max_Dist))
  if ((*Depth2 < Delta_Esfera) || (*Depth2 > Max_Dist))
    return (FALSE);
  else
    *Depth1 = *Depth2;
else
  if ((*Depth2 < Delta_Esfera) || (*Depth2 > Max_Dist))
    *Depth2 = *Depth1;

Achou_Esfera++;
return (TRUE);
}
```

Listagem do algoritmo para o cálculo de intersecção de um raio com cilindros e cones.

```
/*************************************************************
*
*  Esta rotina calcula a intersecção de um raio com um
*  cone ou cilindro.
*
*************************************************************/



static int Inter_Cone (Raio, Cone, Depths)
RAY *Raio;
CONE *Cone;
double *Depths;
 {
 double a, b, c, z, t1, t2, len;
 double d;
 VECTOR P, D;
 int i=0;

 Teste_Cone++;


 MInvTransPonto(&P, &Raio->Inicial, Cone->Trans);
 MInvTransDir(&D, &Raio->Direcao, Cone->Trans);

 VComp(len, D);
 VInvEscalEq(D, len);

 if (Cone->Flag_C)
  {
  /* Resolve interseccao com cilindro */
  a = D.x * D.x + D.y * D.y;
  if (a > EPSILON)
   {
   b = P.x * D.x + P.y * D.y;
   c = P.x * P.x + P.y * P.y - 1.0;
   d = b * b - a * c;
   if (d >= 0.0)
    {
    d = sqrt(d);
    t1 = (-b + d) / a;
    t2 = (-b - d) / a;
    z = P.z + t1 * D.z;
    if (t1 > Delta_Cone && t1 < Max_Dist && z >= 0.0 && z <= 1.0)
     Depths[i++] = t1/len;
    z = P.z + t2 * D.z;
    if (t2 > Delta_Cone && t1 < Max_Dist && z >= 0.0 && z <= 1.0)
     Depths[i++] = t2/len;
    }
```

```
   }
  }
 else
  {
  /* Resolve interseccoes com cone */
  a = D.x * D.x + D.y * D.y - D.z * D.z;
  b = D.x * P.x + D.y * P.y - D.z * P.z;
  c = P.x * P.x + P.y * P.y - P.z * P.z;

  if (fabs(a) < EPSILON)
   {
   if (fabs(b) > EPSILON)
    {
    /* Uma interseccao */
    t1 = -0.5 * c / b;
    z = P.z + t1 * D.z;
    if (t1 > Delta_Cone && t1 < Max_Dist && z >= Cone->dist && z <= 1.0)
     Depths[i++] = t1/len;
    }
   }
  else
   {
   /* Verifica os lados do cone */
   d = b * b - a * c;
   if (d >= 0.0)
    {
    d = sqrt(d);
    t1 = (-b - d) / a;
    t2 = (-b + d) / a;
    z = P.z + t1 * D.z;
    if (t1 > Delta_Cone && t1 < Max_Dist && z >= Cone->dist && z <= 1.0)
     Depths[i++] = t1/len;
    z = P.z + t2 * D.z;
    if (t2 > Delta_Cone && t1 < Max_Dist && z >= Cone->dist && z <= 1.0)
     Depths[i++] = t2/len;
    }
   }
  }

 if (Cone->Fechado)
  {
  d = (1.0 - P.z) / D.z;
  a = (P.x + d * D.x);
  b = (P.y + d * D.y);
  if ((a * a + b * b) <= 1.0 && d > Delta_Cone && d < Max_Dist)
   Depths[i++] = d/len;
  d = (Cone->dist - P.z) / D.z;
  a = (P.x + d * D.x);
  b = (P.y + d * D.y);
  if ((a * a + b * b) <= (Cone->Flag_C ? 1.0 : Cone->dist*Cone->dist)
   && d > Delta_Cone && d < Max_Dist)
   Depths[i++] = d/len;
  }
```

```
Achou_Cone +=i;

return(i);
}
```

BIBLIOGRAFIA

ARVO, J.; KIRK, D. Fast Ray Tracing by Ray Classification. **Computer Graphics**, v.21, n.4, p.55-64, Jul. 1987.

CLEARLY, J. G; WYVILL, B. M.; BIRTWISTLE, G. M.; VATTI, R. Multiprocessor Ray Tracing. **Computer Graphics Forum**, v.5, n.3, p.3-12. 1985.

COOK, R. L. Stochastic Sampling in Computer Graphics. **ACM Transaction on Graphics**, v.5, n.1, p. 51-72, Jan. 1986.

DIPPE, M.; SWENSEN, J. An Adaptive Subdivision Algorith and Parallel Architecture for Realistic Image Syntesis. **Computer Graphics**, v.18, n.3, p.149-158, Jul. 1984.

DIPPE, M.; WOLD, E. H. Antialiasing Trough Stochastic Sampling. **Computer Graphics**, v.19, n.3, p.69-78, Jul. 1985.

ERVIN, Stephen M. Landscape Visualization with Emaps. **IEEE Computer Graphics and Applications**, p.28-33, Mar. 1993.

FIGUEROA, Franz J.; *et al.* **A Realização do Filme CG. com Filtro,** In: SECOMP - Primeira Semana da Computação na Universidade Federal do Rio Grande do Sul. CPGCC-UFRGS, Porto Alegre: Out, 1992.

FIGUEROA, Franz J. **Métodos Computacionais para a Aceleração de Ray Tracing**. Porto Alegre: CPGCC-UFRGS, 1993. Trabalho Individual.

FOLEY, J.; DAM, A. van; FEINER, S. K.; HUGES, J. F. **Computer Graphics - Principles and Practice.** Second Edition, Addison - Wesley Publishing Company, 1990.

FOURNIER, A. A Simple Model of Ocean Waves. **Computer Graphics**, v.20, n.4, p.75-84. 1986.

FUJIMOTO, A.; TANAKA, T.; IWATA, K. ARTS: Accelerated Ray Tracing System. **IEEE Computer Graphics and Applications**, v.6, n.4, p.16-26, Abr. 1986.

GARDENER, G. Y. Simulation of Natural Scenes using Textured Quadric Surfaces. **Computer Graphics**, v.18, n.3, p.11-20. 1984.

GLASSNER, A. S. Space Subdivisions for Fast Ray Tracing. **IEEE Computer Graphics and Applications**, v.4, n.10, p.15-22, Out. 1984.

GLASSNER, A. S. **An Introduction to Ray Tracing**. Fourth Edition. Academic Press Limited. San Diego, 1991.

INAKAGE, M.. A Simple Model of Flames In: **Computer Graphics International '90**, p.73-81. 1990.

ISLER, V.; ÖZGÜÇ, Bülent. Fast Ray Tracing 3D models. **Computer and Graphics**, v.15, n.2, p.205-216. 1991.

JOY, K. I.; BHETANABHOTLA, M. N. Ray Tracing Parametric Surface Patches Utilizing Numerical Techniques and Ray Coherence. **Computer Graphics**, v.20, n.4, p.279-285, Ago. 1986.

KAJIYA, J. T. New Techniques for Ray Tracing Procedurrally Defined Objects. **Computer Graphics**, v.17, n.3, p.91-102, Jul. 1983.

KAY, T. L.; KAJIYA, J. T. Ray Tracing Complex Scenes. **Computer Graphics**, v.20, n.4, p.269-278, Ago. 1986.

LEE, M.; REDNER, R. A.; USELTON, S. P. Statistically Optimized Sampling for Distributed Ray Tracing. **Computer Graphics**, v.19, n.3, p.61-67, Jul. 1985.

MANDELBROT, B. B. **The Fractal Geometry of Nature.** W. H. Freeman Company, San Francisco, CA. 1982.

MILLER, G.; PEARCE, A. Globular Dynamics: A Connected Particle System for Animating Viscous Fluids, *in*: **Topics in Physically-Based Modeling**, ACM SIGGRAPH '87, v.30. 1989.

MULLER, H. Image Generation by Space Sweep. **Computer Graphics Forum**, v.5, p.189-196. 1986.

PEACHEY, D. R. Modelling Waves and Surf. **Computer Graphics**, v.20, n.4, p.65-74. 1986.

PEITGEN, H. O. **The Science of Fractal Images.** Springer, New York. 1988.

PERLIN, K. An Image Synthesizer. **Computer Graphics**, v.19, n.3, p.287-296. 1985.

PHONG, B. T. Illumination for Computer Generated Pictures. **Communications of the ACM**, v.18, n.6, p.311-317, Jun. 1975.

PURGATHOFER, W. A Statistical Method for Adaptive Stochastic Sampling. **Proceedings Eurographics'86**, p.145-152. 1986.

REEVES, W. T. A Technique for Modeling a Class of Fuzzy Objects. **Computer Graphics**, v.17, n.3, p.359-76. 1983.

REEVES, W. T., BLAU, R. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. **Computer Graphics**, v.19, n.3, p.313-22. 1985.

ROGERS, D. F. **Procedural Elements for Computer Graphics**, p.34-38, McGraw Hill, New York. 1985.

RUBIN, S.; WHITTED, T. A Three Dimensional Representation for Fast Rendering Complex Scenes. **Computer Graphics**, v.14, n.3, p.110-116, Jul. 1980.

SHINYA, M.; TAKAHASHI, T.; NAITO, S. Principles and applications of Pencil Tracing. **Computer Graphics**, v.21, n.4, p.45-54, Jul. 1987.

SIMS, K. Particle Animation and Rendering using Data Parallel Computation. **Computer Graphics**, v.24, n.4, p.405-13. 1990.

SWEENWY, M. A. J.; BARTELS, R. H. Ray Tracing Free-Form B-spline Surfaces. **IEEE Computer Graphics and Applications**, v.6 n.2, p.41-49, Fev. 1986.

TESTA, Bridget M.. Graphical Treasures on the Internet. **Computer Graphics World**, v.17 n.11 p.34-41, Nov. 1994.

THALMANN, N. M.; THALMANN, D. **Image Synthesis**, Springer-Verlag. 1987.

TOTTH, D. L. On Ray Tracing Parametric Surfaces. **Computer Graphics**, v.19 n.3, p.177-179, Jul. 1985.

VAN WIJK, J. J. Ray Tracing Objects Defined by Sweeping Planar Cubic Splines. **Computer Graphics**, v.6 n.2, p.41-49, Fev. 1986.

WATT, A. **Fundamentals of Three-Dimensional Computer Graphics**, Addison-Wesley, Wokingham, UK. 1989.

WATT, Alan., WATT, Mark. **Advanced Animation and Rendering Techniques**, Addison-Wesley, Wokingham, UK. 1992.

WHITTED, T. An Improved Illumination Model for Shaded Display. **Communications of the ACM**, v.23 n.6, p.343-349, Jun. 1980.

"O amor jamais acaba;
mas havendo profecias, serão aniquiladas;
havendo línguas, cessarão;
havendo ciência, passará;
Agora, pois, permanecem
a fé, a esperança e o amor,
estes três, porém o maior destes
é o amor"
(I Coríntios 13:8,13)