

**UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

**FERRAMENTA PARA GERAÇÃO EM  
TEMPO REAL DE BORDAS DE MAPAS  
VIRTUAIS PSEUDO-INFINITOS PARA  
JOGOS 3D**

**DISSERTAÇÃO DE MESTRADO**

**Fernando Bevilacqua**

**Santa Maria, RS, Brasil**

**2009**

**FERRAMENTA PARA GERAÇÃO EM TEMPO REAL  
DE BORDAS DE MAPAS VIRTUAIS  
PSEUDO-INFINITOS PARA JOGOS 3D**

**por**

**Fernando Bevilacqua**

Dissertação apresentada ao Programa de Pós-Graduação em Informática  
da Universidade Federal de Santa Maria (UFSM, RS), como requisito  
parcial para a obtenção do grau de

**Mestre em Informática**

**Orientador: Prof. Dr Cesar Tadeu Pozzer (UFSM)**

**Dissertação de Mestrado Nº 6  
Santa Maria, RS, Brasil**

**2009**

**Universidade Federal de Santa Maria  
Centro de Tecnologia  
Programa de Pós-Graduação em Informática**

A Comissão Examinadora, abaixo assinada,  
aprova a Dissertação de Mestrado

**FERRAMENTA PARA GERAÇÃO EM TEMPO REAL DE  
BORDAS DE MAPAS VIRTUAIS PSEUDO-INFINITOS PARA  
JOGOS 3D**

elaborada por  
**Fernando Bevilacqua**

como requisito parcial para obtenção do grau de  
**Mestre em Informática**

**COMISSÃO EXAMINADORA:**

**Prof. Dr Cesar Tadeu Pozzer (UFSM)**  
(Presidente/Co-orientador)

**Prof. Dr Marcos Cordeiro d'Ornellas (UFSM)**

**Prof. Dr Esteban Walter Gonzalez Clua (UFF)**

## **DEDICATÓRIA**

A minha família e a Marília, meu amor.

## **AGRADECIMENTOS**

Primeiramente gostaria de agradecer a Deus por me amparar, me dar sabedoria e saúde para trilhar esse caminho.

Em seguida, gostaria de agradecer a minha família, por terem me apoiado e lutado para que eu tivesse a oportunidade de chegar aqui. Meu muito obrigado a vocês por tudo!

Meu obrigado especial a Marília, meu amor, que sempre esteve ao meu lado durante todos os momentos, bons e ruins. Sem suas palavras e apoio, os desafios que eu passei teriam me derrubado e impedido que eu concluísse esse trabalho. Obrigado pelo seu amor e carinho, por estar ao meu lado e por trazer felicidade e equilíbrio à minha vida!

Obrigado também ao meu orientador, Prof. Cesar Pozzer, por toda ajuda e horas de dedicação gastas comigo. Sempre entendi a sua preocupação para com a qualidade do trabalho, mesmo quando eu parecia não ter tanto tempo para finalizá-lo. Foi um prazer trabalhar com ele durante esses anos. Obrigado também ao Prof. Marcos d'Ornellas, pela ajuda e pelo seu bom senso, ambos foram muito úteis para mim. Obrigado ao pessoal do LaCA, que de alguma forma ou de outra me ajudaram também.

Por fim, mas não menos importante, agredeço aos meus colegas de trabalho da Decadum. O suporte e dedicação, tanto acadêmico quanto profissional, permitiram que eu concluísse esse trabalho. Obrigado por assumirem minhas responsabilidades quando eu estive ausente. Meu obrigado especial ao Vicentini, que me ajudou muito ao longo do mestrado.

*“Não é possível 9 mulheres gerarem um bebê em 1 mês.”* — FREDERICK  
BROOKS

# RESUMO

Dissertação de Mestrado  
Programa de Pós-Graduação em Informática  
Universidade Federal de Santa Maria

## FERRAMENTA PARA GERAÇÃO EM TEMPO REAL DE BORDAS DE MAPAS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D

Autor: Fernando Bevilacqua  
Orientador: Prof. Dr Cesar Tadeu Pozzer (UFSM)  
Local e data da defesa: Santa Maria, 29 de Julho de 2009.

O mercado de jogos para computador vem evoluindo consideravelmente ao longo dos anos. Dentro da categoria de jogos *multiplayer*, existem os jogos *massively multiplayer online* (MMO), que são jogos online no qual uma grande quantidade de jogadores interage uns com os outros dentro de um mundo virtual persistente. Em jogos MMO a existência de um mundo virtual persistente é um tópico importante para manter o jogo atrativo e divertido ao jogador. Quanto maior o mundo a ser explorado, tecnicamente mais tempo o usuário passará jogando para conseguir explorar o maior número possível de lugares. A solução proposta neste trabalho é uma ferramenta capaz de gerar mundos virtuais pseudo-infinitos com diversificação de formas e relevos ao longo de sua extensão. Utilizando uma combinação de algoritmos e métodos de gerenciamento de conteúdo, a ferramenta é capaz de criar praias, ilhas/arquipélagos, baías e costas que imitam as paisagens encontradas na natureza. Além disso, a possibilidade de parametrizar cada um desses elementos dá ao desenvolvedor um controle maior sobre o resultado que será obtido. Dentre as inovações apresentadas, estão a criação de um terreno virtual de vastas proporções com enfoque mais detalhado no que diz respeito à geração da costa. Fruto de um aprofundamento nas pesquisas já realizadas na área, o desenvolvimento da ferramenta foi focado em tratar de forma diferenciada a criação de conteúdo para os diversos elementos existentes (como continentes, relevo, etc). A ferramenta é capaz de gerar continentes contendo falésias, praias e rochedos em suas extremidades.

**Palavras-chave:** MMO, mundos virtuais, geração de terreno, jogos 3D, ruído, geração procedural, multifractal.

## **ABSTRACT**

Master's Dissertation  
Programa de Pós-Graduação em Informática  
Universidade Federal de Santa Maria

### **TOOL FOR REAL-TIME GENERATION OF COASTLINES FOR PSEUDO-INFINITE VIRTUAL WORLDS FOR 3D GAMES**

Author: Fernando Bevilacqua  
Advisor: Prof. Dr Cesar Tadeu Pozzer (UFSM)

Massively multiplayer online (MMO) games are multiplayer games featuring a huge amount of player living in a persistent virtual world. In MMO games the player's experience is mainly influenced by the size and details of the virtual world. Technically the bigger the world is, the bigger is the time the player takes to explore all the places. This work proposes a tool able to generate pseudo-infinite virtual worlds with different types of terrain. Using a combination of algorithms and content management methods, the tool is able to create beaches, islands, bays and cost lines that imitates the real world landscapes. One of the contributions of the tool is the ability to generate giant pieces of land focusing on the coast line generation. The development of the present work aimed to handle separately the generation of content for all elements in the world (continents, terrains, etc.). The main point in the work is the coast line generation, not the content inside the continents.

**Keywords:** MMO, virtual worlds, terrain generation, 3D games, noise, procedural generation, multifractal.

## LISTA DE FIGURAS

Figura 1.1 – Ilustração do mundo virtual de World of Warcraft [Blizzard Entertainment, 2007] .....	15
Figura 1.2 – Ilustração do mundo virtual gerado.....	17
Figura 2.1 – Quatro primeiras iterações do fractal de floco-de-neve de Koch [Dollins, 2002] .....	20
Figura 2.2 – À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [Elias, 2009] .....	20
Figura 2.3 – Desenho em 2D dos pontos gerados por uma função de ruído de Perlin [Elias, 2009] .....	21
Figura 2.4 – À esquerda, <i>heightmap</i> gerado a partir de divisões estocásticas; à direita, o mesmo <i>heightmap</i> renderizado em 3D [Dollins, 2002].....	23
Figura 2.5 – À esquerda, <i>heightmap</i> gerado a partir de falhas geológicas; à direita, o mesmo <i>heightmap</i> renderizado em 3D [Ribble, 2001].....	24
Figura 2.6 – <i>Heightmaps</i> obtidos com o algoritmo de deposição de sedimentos. [Ribble, 2001] .....	25
Figura 2.7 – Geração de relevo pela disposição do ponto médio (perspectiva 2D) [Dollins, 2002].....	25
Figura 2.8 – À direita, <i>heightmap</i> gerado a partir de ruído de Perlin com alta frequência. À esquerda, <i>heightmap</i> gerado com ruídos de menor frequência. [Häggström, 2006] .....	25
Figura 2.9 – À esquerda, <i>heightmap</i> gerado a partir da técnica de ruído de Perlin <i>ridged</i> ; à direita, o mesmo <i>heightmap</i> renderizado em 3D [Häggström, 2006] .....	26
Figura 2.10 –Mundo virtual dividido em células. Cada uma delas possui uma semente calculada com base em sua posição e com base uma semente global referente à cidade [Greuter et al., 2005] .....	27
Figura 2.11 –Geração de construções: cada andar é gerado pela mescla de polígonos escolhidos e centralizados aleatoriamente [Greuter et al., 2005]	27
Figura 2.12 –Campo de visão do usuário: apenas as células visíveis tem o seu conteúdo gerado [Greuter et al., 2005] .....	28
Figura 2.13 –Exemplo de cidade virtual gerada [Greuter et al., 2005].....	28
Figura 2.14 –Terreno texturizado com duas camadas: bloco escuro e um conjunto de texturas em formato de borda com transparência [Häggström, 2006]	30
Figura 2.15 –Diferentes relevos gerados pela utilização de fractais e suas combinações. [Linda, 2007] .....	31

Figura 2.16 – Planetas gerados com os diversos algoritmos. (a) Falhas aleatórias. (b) Disposição do ponto médio. (c) Disposição do ponto médio multifractal. (d) Ruído de Perlin. (e) Ruído de Perlin Multifractal. (f) Ruído de Perlin <i>Ridged</i> . (g) Ruído de Perlin <i>Ridged</i> Multifractal. [Linda, 2007] .....	33
Figura 2.17 –(a) Mapa de altura gerado a partir de um diagrama de Voronoi. (b) Aplicações de funções de ruído em (a). (c) Mapa resultante da aplicação de um filtro de perturbação em (b). [Olsen, 2004] .....	34
Figura 2.18 –À direita, renderização de um mapa de altura sem a aplicação de erosão; à esquerda, o mesmo mapa com a aplicação de erosão. [Olsen, 2004] .....	34
Figura 2.19 – <i>Quadtree</i> utilizada para a geração do terreno [Dollins, 2002] .....	36
Figura 2.20 –Tetraedro inicial utilizado no processo de projeção. [Mogensen, 2009]	38
Figura 2.21 –Mapas obtidos com a aplicação do algoritmo de subdivisão do tetraedro. [Mogensen, 2009] .....	38
Figura 2.22 –Zoom realizado sobre diferentes mapas. [Mogensen, 2009] .....	39
Figura 3.1 – Estrutura básica para geração de conteúdo .....	43
Figura 3.2 – Mapeamento da MM para o mundo virtual: cada vértice da MM é mapeado para diversos vértices no mundo virtual.....	45
Figura 4.1 – Problema da geração de conteúdo sob demanda. ....	47
Figura 4.2 – Organização do sistema de coordenadas do mundo virtual .....	48
Figura 4.3 – Campo de visão do usuário: a área visualizada corresponde a uma fatia do mundo virtual existente. ....	49
Figura 4.4 – Mapeamento das coordenadas do mundo virtual para as coordenadas de desenho da tela.....	51
Figura 4.5 – Conjunto de imagens utilizadas para texturização do terreno .....	52
Figura 4.6 – Geração de conteúdo com base apenas nas informações do campo de visão .....	54
Figura 4.7 – Função de geração de relevo parametrizável e o resultado gerado .....	55
Figura 4.8 – Planetas aleatórios gerados com diferentes sementes de entrada. (a) e (b) mapas com informações de relevo [Mogensen, 2009]; (c) mapa mostrando apenas informações sobre terra e mar. .....	59
Figura 4.9 – Resultado gráfico obtido quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher as discrepâncias de mapeamento da MM.....	60
Figura 4.10 –Funcionamento do algoritmo de quebra de linearidade da costa .....	62
Figura 4.11 –Representação gráfica do espectro de ruído utilizado para criação da costa, com diferentes níveis de granularidade. (a) Granularidade média. (b) Granularidade alta. (c) Granularidade muito alta. (d) Granularidade muito baixa. ....	64
Figura 4.12 –Funcionamento do algoritmo de geração de praias .....	67
Figura 4.13 –Praia gerada ao longo da costa .....	67
Figura 4.14 –Resultado obtido com a aplicação do diferenciador de praias .....	69
Figura 4.15 –Ilha obtida com o gerador de ilhas.....	70
Figura 5.1 – Representação 2D dos continentes gerados pela ferramenta.....	73
Figura 5.2 – Continentes e oceanos gerados pela ferramenta .....	74

Figura 5.3 – Relevo gerado pela ferramenta .....	75
Figura 5.4 – Área com relevo de baixa frequência .....	76
Figura 5.5 – Diversos relevos gerados a partir de diferentes espectros de ruídos. (a) Pouco ruído estendido para o mundo inteiro sem replicação. (b) Pouco ruído replicado 200 vezes. (c) Bastante ruído com replicação de 200 vezes. (d) Bastante ruído sem replicação.....	77
Figura 5.6 – Local de encontro de duas linhas da costa sem a aplicação de qualquer algorítimo de geração de conteúdo extra .....	78
Figura 5.7 – Pequena baía com rochas .....	79
Figura 5.8 – Baía de médio porte gerada a partir da criação de um braço de terra originada no continente.....	81
Figura 5.9 – Extremidade de um continente .....	81
Figura 5.10 –Costa de um continente com praias de tamanho variável .....	82
Figura 5.11 –Baía criada a partir da geração de dois braços de terra originados no continente .....	82
Figura 6.1 – Alteração na estrutura atual para permitir uma maior flexibilidade na geração de conteúdos para diferentes tipos de terreno.....	85

# SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	14
<b>1.1 Contexto e Motivação .....</b>	14
<b>1.2 Objetivos e Contribuição .....</b>	16
<b>1.3 Organização do Texto .....</b>	17
<b>2 REVISÃO DA LITERATURA .....</b>	19
<b>2.1 Funções de ruído e fractais .....</b>	19
<b>2.1.1 Fractais .....</b>	19
<b>2.1.2 Ruído de Perlin .....</b>	20
<b>2.2 Geração de relevo .....</b>	22
<b>2.2.1 Divisões estocásticas .....</b>	22
<b>2.2.2 Falhas geológicas .....</b>	23
<b>2.2.3 Deposição de sedimentos .....</b>	24
<b>2.2.4 Disposição do ponto médio .....</b>	24
<b>2.2.5 Ruído de Perlin .....</b>	25
<b>2.3 Mundos pseudo-infinitos .....</b>	26
<b>2.3.1 Cidade virtual .....</b>	26
<b>2.3.2 Geração e texturização por camadas .....</b>	28
<b>2.3.3 Planetas procedurais .....</b>	30
<b>2.3.4 Fractais refinados por erosão .....</b>	32
<b>2.3.5 Divisões estocásticas de uma <i>quadtree</i> .....</b>	35
<b>2.3.6 Gerador de mapas para planetas .....</b>	37
<b>3 ESTRUTURA DA FERRAMENTA .....</b>	40
<b>3.1 Visão geral da ferramenta .....</b>	40
<b>3.2 Análise inicial .....</b>	41
<b>3.3 Estrutura básica .....</b>	42
<b>3.3.1 Gerador de mapas .....</b>	43
<b>3.3.2 Gerenciador de fatias .....</b>	43
<b>3.3.3 Gerador de relevo .....</b>	44
<b>3.3.4 Gerador de costa .....</b>	44
<b>3.3.5 Renderizador .....</b>	45
<b>4 IMPLEMENTAÇÃO .....</b>	46
<b>4.1 Terreno infinito .....</b>	47
<b>4.1.1 Renderização de conteúdo .....</b>	51
<b>4.2 Relevo .....</b>	53

4.2.1	Problemas na geração de relevo sob-demanda .....	53
4.2.2	Solução para geração de relevo sob-demanda .....	54
4.2.3	Otimização através de conservação de dados .....	56
<b>4.3</b>	<b>Continentes .....</b>	<b>57</b>
4.3.1	Oceano como um plano azul .....	57
4.3.2	Pré-processamento de faixas de terra .....	58
4.3.3	Visão micro e macro do mundo .....	59
4.3.4	Quebra de linearidade da costa .....	61
4.3.5	Praias .....	65
4.3.6	Arquipélagos e praias diferenciadas .....	68
<b>5</b>	<b>RESULTADOS .....</b>	<b>71</b>
<b>5.1</b>	<b>Aspectos gerais da ferramenta .....</b>	<b>71</b>
<b>5.2</b>	<b>Avaliação de técnicas e resultados obtidos .....</b>	<b>72</b>
5.2.1	Avaliação dos continentes .....	72
5.2.2	Avaliação do relevo .....	74
5.2.3	Avaliação da costa .....	78
<b>5.3</b>	<b>Análise de desempenho .....</b>	<b>80</b>
<b>6</b>	<b>CONCLUSÃO E TRABALHOS FUTUROS .....</b>	<b>83</b>
<b>REFERÊNCIAS .....</b>		<b>87</b>

# 1 INTRODUÇÃO

## 1.1 Contexto e Motivação

O mercado de jogos para computador vem evoluindo consideravelmente ao longo dos anos. Desde o lançamento dos primeiros consoles, o poder de processamento de hardware aumentou e novas tecnologias gráficas foram desenvolvidas, o que colaborou no desenvolvimento de jogos com os mais variados temas e estilos. Agrupando-se esses jogos pela forma como o usuário interage, obtém-se dois grandes grupos de jogos: os *singleplayer* e os *multiplayer*. No primeiro grupo, os jogadores interagem com o jogo de uma forma solitária, ou seja, sem interação com outros seres humanos; embora existam diálogos e interações com personagens do jogo (chamados *NPCs*), a interação é realizada através de ações pré-determinadas ou técnicas de inteligência artificial. Já no segundo grupo, os jogadores interagem com NPCs e, também, com outros seres humanos representados por personagens virtuais. A popularidade desse gênero é grande e a interação social entre os jogadores é assunto de pesquisas [Griffiths et al., 2003, Ducheneaut et al., 2006].

Dentro da categoria de jogos *multiplayer*, existem os jogos *massively multiplayer online* (MMO), que são jogos online no qual uma grande quantidade de jogadores interage uns com os outros dentro de um mundo virtual persistente. A interação dos jogadores é dada de várias formas, como batalhas, execução de missões pré-definidas (*quests*), busca de recursos, enfim, uma vasta gama de possibilidades sobre um mundo com economia e política próprios. A quantidade de jogadores vinculados a um MMO pode chegar a ordem dos milhões, como é o caso do jogo World of Warcraft [Blizzard Entertainment, 2007], com mais de 6 milhões de inscritos [Clark, 2006].

Em jogos MMO a existência de um mundo virtual persistente é um tópico importante para manter o jogo atrativo e divertido ao jogador. Quanto maior o mundo a ser explorado, tecnicamente mais tempo o usuário passará jogando para conseguir explorar



Figura 1.1: Ilustração do mundo virtual de World of Warcraft [Blizzard Entertainment, 2007]

o maior número possível de lugares. Utilizando o jogo World of Warcraft como exemplo, constata-se a existência de um mundo virtual complexo construído em torno de dois planetas diferentes: Azeroth e Outland, sendo o primeiro dividido em dois grandes continentes. Ao longo desses continentes, estão espalhadas várias cidades, dentre elas quatro capitais para cada uma das facções existentes no jogo (aliança e horda). A figura 1.1 ilustra o mundo virtual do jogo World of Warcraft. Um outro exemplo de MMO é o jogo EverQuest [Sony Entertainment, 2007]. Nesse jogo, o mundo virtual é dividido em diversas zonas, as quais representam elementos geográficos variados, como cidades, desertos e planícies.

Dada a magnitude de tais mundos virtuais, a criação (e consequente renovação) de cenários torna-se uma tarefa complexa e de proporções consideráveis. Tanto em World of Warcraft quanto em EverQuest, os mundos virtuais apresentam uma diversidade grande de elementos geográficos, como cadeias montanhosas, vales, florestas, campos, cavernas, etc., sendo a grande maioria deles ligados a história do jogo. A criação manual desses mundos virtuais exige uma equipe apta para modelar relevos, ornamentar paisagens, garantir usabilidade do mapa (não criar lugares inatingíveis, por exemplo), criar lugar interessantes aos jogadores, etc.

Em meio a esse contexto, a criação de uma ferramenta capaz de gerar mundos virtuais complexos e de grandes proporções pode ajudar e agilizar o desenvolvimento de jogos 3D ao estilo MMO.

## 1.2 Objetivos e Contribuição

A solução proposta neste trabalho tem por objetivo o desenvolvimento de uma ferramenta capaz de gerar mundos virtuais complexos e de grandes proporções, em tempo real. Através de técnicas de geração de relevo e criação de mapas por interpolação de funções de ruído, a ferramenta proposta é capaz de gerar um mundo virtual com continentes, oceanos, áreas planas, montanhas, baías, praias e falésias, com enfoque na geração da **borda dos continentes**. Todos esses elementos são processados sob demanda à medida que o jogador caminha pelo ambiente virtual.

**Partindo do fato de que o armazenamento em memória da malha de polígonos do mundo completo é proibitiva, a ferramenta gera todos os conteúdo do ambiente sob demanda. À medida que o usuário avança pelo terreno, os elementos que entram para o campo de visão são processados e colocados na memória e, à medida que eles saem do campo de visão, são removidos. Embora a geração dos elementos seja baseada em cálculos aleatórios, se o jogador passar pelo ponto A e, em seguida, andar por quilômetros, colocando uma gama completamente diferente de elementos na memória, e ele retornar ao ponto A, a mesma paisagem que foi vista na primeira vez será mostrada novamente. A geração de conteúdo, na abordagem sob demanda, permite que o mundo virtual seja tecnicamente infinito.**

Além da geração de relevo, a ferramenta também é capaz de criar oceanos e continentes, todos customizáveis. Diferentemente de outros trabalhos relacionados, nos quais a geração de continentes (e suas costas) é uma consequência da geração de relevo, o presente trabalho mostra como sua principal contribuição uma nova abordagem para a geração de conteúdos para mundos virtuais. Nessa abordagem, a geração de continentes, relevo e costas é tratada de forma separada, porém culminando num resultado final único.

A forma dos continentes é definida, em um nível macro, através de um algoritmo de criação de mapas que trabalha projetando pixels em uma esfera, o que resulta num mapa contínuo e sem distorções. A criação da costa, por sua vez, trabalha em

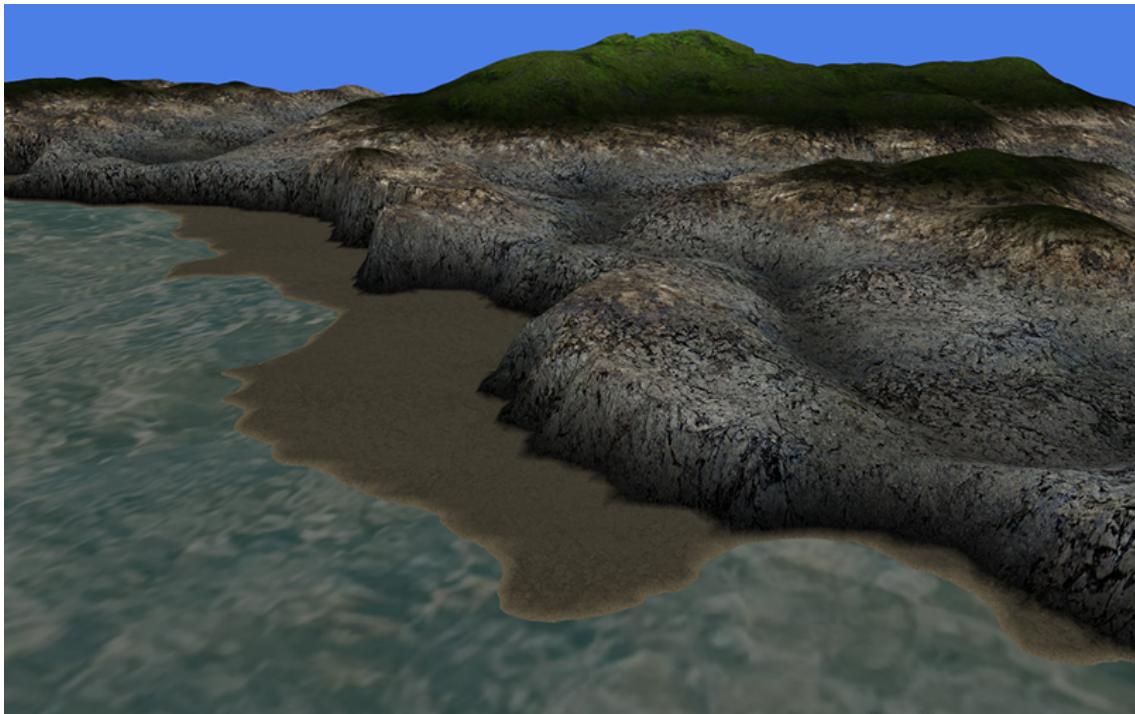


Figura 1.2: Ilustração do mundo virtual gerado

**um nível micro que utiliza as informações de mais alto nível para criar conteúdos mais refinados e ricos em detalhes. O resultado final apresenta costas com falésias, praias de tamanhos variados, baías e arquipélagos. A figura 1.2 ilustra o mundo virtual obtido com a ferramenta.**

### 1.3 Organização do Texto

**Este trabalho está organizado da seguinte forma: no capítulo 2 apresenta-se uma revisão da literatura relacionada à criação de mundos virtuais e geração de relevo através de técnicas diversas. Na primeira metade do capítulo são apresentadas técnicas utilizadas para a representação de terrenos virtuais, bem como algorítimos e métodos para a geração de paisagens que são semelhantes com a natureza do mundo real (como criação de montanhas através da deposição de sedimentos); na segunda metade do capítulo, ferramentas e contribuições semelhantes ao do presente trabalho são analisadas, ressaltando-se tópicos intimamente ligados com a proposta dessa dissertação e que foram utilizados como base para o desenvolvimento dos trabalhos de implementação.**

**O capítulo 3 apresenta uma descrição de como a ferramenta está organizada. No capítulo 4 busca-se destacar os problemas encontrados para a geração de um mundo**

**virtual pseudo-infinito com processamento de conteúdo sob-demanda, bem como as soluções encontradas e implementadas para cada um desses problemas. O capítulo 5 apresenta os resultados obtidos com a ferramenta desenvolvida, com análises de desempenho e ilustração das funcionalidades.**

**Por fim o capítulo 6 apresenta as considerações finais do trabalho. Além disso, apresenta-se sugestões de melhoria da ferramenta e ideias para a continuidade do trabalho.**

## 2 REVISÃO DA LITERATURA

### 2.1 Funções de ruído e fractais

As formas encontradas na natureza são geralmente complexas e ricas em detalhes. Uma cadeia montanhosa, por exemplo, apresenta variações de altitude que às vezes não pode ser representada por uma função matemática, visto que ela não possui um padrão definido (ela é completamente aleatória). Em contra-partida, certos elementos naturais apresentam um padrão bem definido, como as cores na concha de um molusco.

Para simular elementos naturais e suas peculiaridades em computação gráfica, pode-se utilizar fractais e funções de ruído. Fractais simulam elementos que possuem um padrão definido, enquanto funções de ruído adicionam a aleatoriedade existente no mundo real. Essa seção explana sobre conceitos e técnicas relacionadas a fractais e funções de ruído.

#### 2.1.1 Fractais

Fractais são equações matemáticas geradas recursivamente, sendo que a cada nível de recursão mais e mais detalhes da mesma forma são gerados [Kelly and McCabe, 2004]. As formas geradas são auto-similares e geralmente a cada nível de recursão a equação original é vista novamente, com algumas alterações. Fractais possuem um nível de detalhes infinito, então quando mais nos aproximamos, mais detalhes ele apresenta. A figura 2.1 ilustra o funcionamento de um fractal. É possível que fractais sejam utilizados para a geração de diversos elementos, tanto terrenos como plantas [Barnsley, 1993]. A utilização de fractais para a geração de relevo é uma técnica comum em computação gráfica, como pode ser visto na seção 2.2.

Conforme já citado, fractais apresentam um padrão de auto-similaridade, o que garante uma certa homogeneidade. As formas encontradas na natureza, porém, são em sua grande maioria heterogêneas, como a forma de um montanha: quando mais próximo do cume,

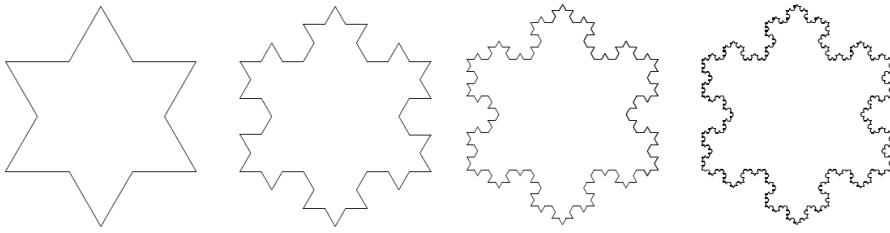


Figura 2.1: Quatro primeiras iterações do fractal de floco-de-neve de Koch [Dollins, 2002]

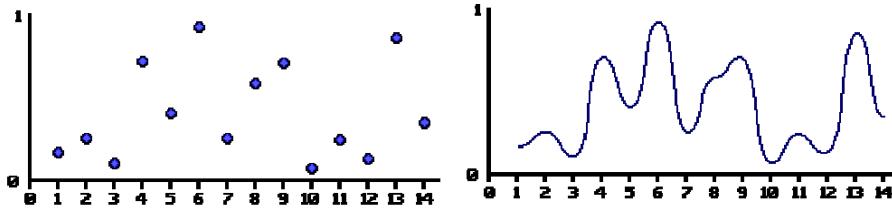


Figura 2.2: À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [Elias, 2009]

mais pontiagudas as formas se tornam, ao passo que na base da montanha as formas são mais suaves e planas. Aplicando o conceito de heterogeneidade à fractais obtém-se o que é denominado *multifractal*. Em um conceito simples, multifractais são fractais com dimensões e detalhes diferenciados ao longo de sua forma e eles são obtidos fazendo com que o seu detalhamento seja calculado em função de algum outro atributo. Considerando o exemplo da montanha já citada, o nível de suavização da forma pode ser dado em função da altitude da montanha, ou seja, quanto mais próximo do cume, menos suave as formas devem ser.

### 2.1.2 Ruído de Perlin

O ruído de Perlin foi desenvolvido por Ken Perlin [Perlin, 1985], cujo objetivo inicial era criar texturas com aparência mais natural. A função de ruído de Perlin gera um intervalo de valores que podem ser utilizados para diversos fins, dentre eles uma forma de adicionar aleatoriedade aos relevos gerados proceduralmente. Os resultados da função de ruído de Perlin são baseados no somatório dos valores de diversas funções redimensionadas, todas elas originadas de uma única função base. Essa função base é obtida a partir da interpolação de valores aleatórios gerados de forma discreta. A figura 2.2 ilustra os pontos gerados e a sua interpolação.

Partindo de um conjunto de pontos discretamente coletados, um espectro de valores

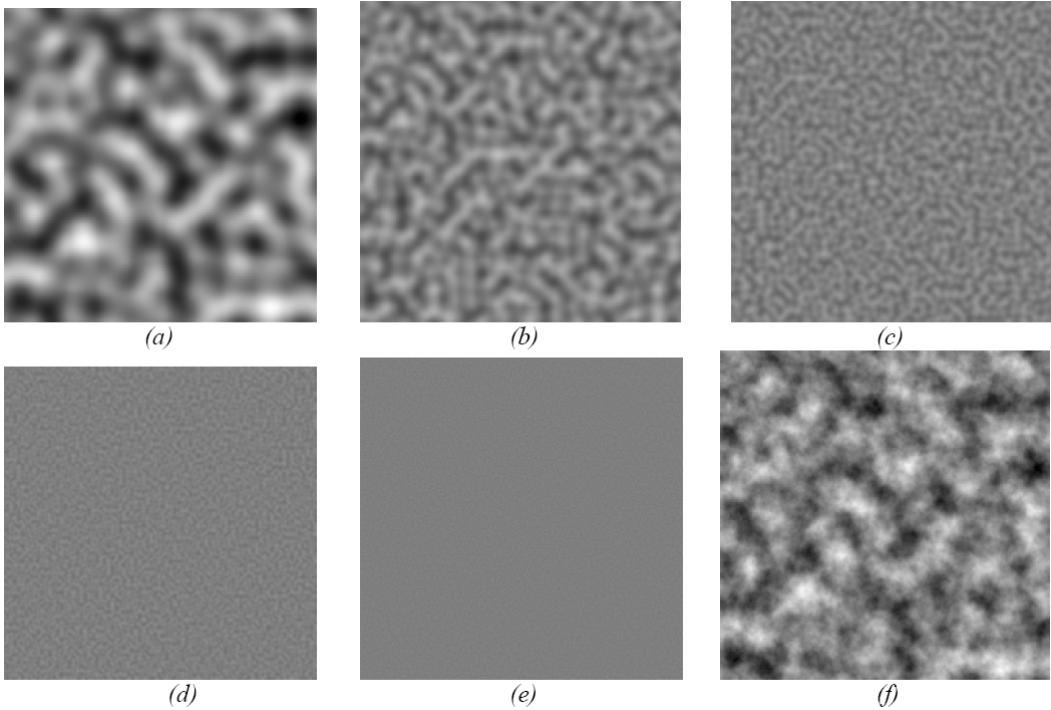


Figura 2.3: Desenho em 2D dos pontos gerados por uma função de ruído de Perlin [Elias, 2009]

é criado. Em seguida, todos os pontos são interpolados entre si, criando um domínio de valores infinito. Para cada parâmetro de entrada que a função de ruído receber, um valor correspondente será encontrado na curva que foi gerada como resultado da interpolação. Para facilitar os cálculos de geração de terreno, por exemplo, pode-se condicionar a função de ruído para que os valores retornados estejam sempre contidos dentro do intervalo  $[-1, 1]$ .

Alterando-se a amplitude e a frequência da função base, obtém-se um conjunto de novas funções em outra escala. A maneira mais comum de obter novas funções em outra escala é dobrando a frequência e reduzindo a amplitude pela metade; cada nova função gerada é chamada de *octava*. O resultado da adição dessas diversas funções (octavas), com diferentes amplitude e frequência, é a função de **ruído de Perlin**. A figura 2.3 ilustra graficamente as funções base utilizadas para a criação da função de ruído; as ilustrações de (a) até (e) são oitavas, sendo (a) a oitava de menor frequência e (e) a oitava de maior frequência. A ilustração (f) mostra o resultado da soma de todas as oitavas, que é a função de ruído de Perlin propriamente dita.

## 2.2 Geração de relevo

O elemento mais básico que compõe um mundo virtual é o terreno. O terreno é a superfície que irá guiar e servir como base para todos os demais elementos que compõem o mundo, como árvores, estradas, cidades, etc. Em computação gráfica, uma das formas possíveis de se representar um terreno é através de um *heightmap*. Nessa abordagem, o terreno é representado por uma matriz de pontos, sendo que cada ponto  $(i, j)$  representa a altura de um ponto do plano. Iterando-se sobre os pontos dessa matriz, conectando cada um deles através de arestas até eles formarem triângulos, é possível modelar uma malha que pode ser utilizada como um terreno virtual.

Essa seção apresenta diversas técnicas que podem ser utilizadas para a geração de um *heightmap* que, quando renderizado, é capaz de produzir relevos muito próximos aos que são encontrados no mundo real.

### 2.2.1 Divisões estocásticas

Essa técnica consiste na subdivisão recursiva do *heightmap* através da utilização de números pseudo-aleatórios. Partindo de uma matriz com todos os pontos tendo a altura zero, o funcionamento do algoritmo é o seguinte:

- Uma altura escolhida aleatoriamente é atribuída a cada um dos cantos do retângulo. Essa altura é proporcional ao tamanho do retângulo.
- O retângulo é dividido em quatro retângulos menores, com as alturas dos cantos de cada retângulos sendo calculada com base na interpolação das alturas dos cantos dos retângulos vizinhos ao retângulo original.
- O algoritmo repete os dois primeiros passos para cada retângulo gerado. A recursão é quebrada quando nível de detalhes desejado é atingido, de forma semelhante ao que ocorre na construção de um terreno utilizando-se uma quadtree.

O resultado obtido com esse algoritmo é um plano que simula as elevações montanhosas do mundo real. Embora essa abordagem seja eficiente, ela não produz resultado muito realistas, visto que a variação de altura ao longo dos pontos é linear, o que não é comum na natureza. Além disso, é possível encontrar picos ou bordas muito pontiagudos ao longo dos cantos dos quadrados gerados, o que gera um relevo irreal [Lewis, 1987]. A figura 2.4 ilustra um relevo gerado a partir de divisões estocásticas.

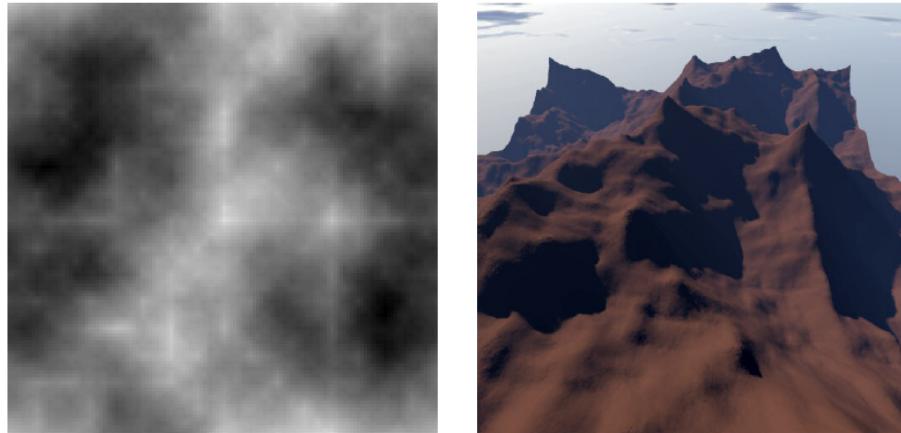


Figura 2.4: À esquerda, *heightmap* gerado a partir de divisões estocásticas; à direita, o mesmo *heightmap* renderizado em 3D [Dollins, 2002]

Existem variações desse algoritmo que visam contornar as imperfeições descritas anteriormente. Uma dessas variações é a subdivisão em forma de diamante, que consiste em rotacionar os novos quadrados gerados em  $45^\circ$  em relação ao quadrado original. Outra variação é afastar os quadrados novos dos cantos altos do quadrado original, recalculando as alturas dos cantos de cada um dos novos quadrados com base em pesos, o que produz um relevo mais suave e sem picos pontiagudos.

### 2.2.2 Falhas geológicas

A geração por falhas geológicas simula a criação de relevo por movimentação de placas tectônicas. Partindo de um terreno plano composto por uma malha regular, o algoritmo escolhe aleatoriamente uma linha que divide o terreno em duas partes. Para um dos lados, a altura dos pontos é incrementada em uma certa quantidade, enquanto a altura dos pontos do outro lado é decrementada dessa mesma quantidade. Depois disso, o valor usado para aumentar/diminuir as alturas é reduzido em uma certa quantidade e o algoritmo é repetido. No momento que esse valor chegar a zero o algoritmo termina.

O relevo gerado com esse método não apresenta elevações pontiagudas como no algoritmo de subdivisões estocásticas, porém ele é muito lento para ser utilizado numa abordagem de tempo real. Depois que a linha divisória é escolhida, todos os pontos da matriz precisam ser atualizados e isso é feito a cada iteração do algoritmo. A figura 2.5 ilustra um relevo gerado a partir de falhas geológicas.

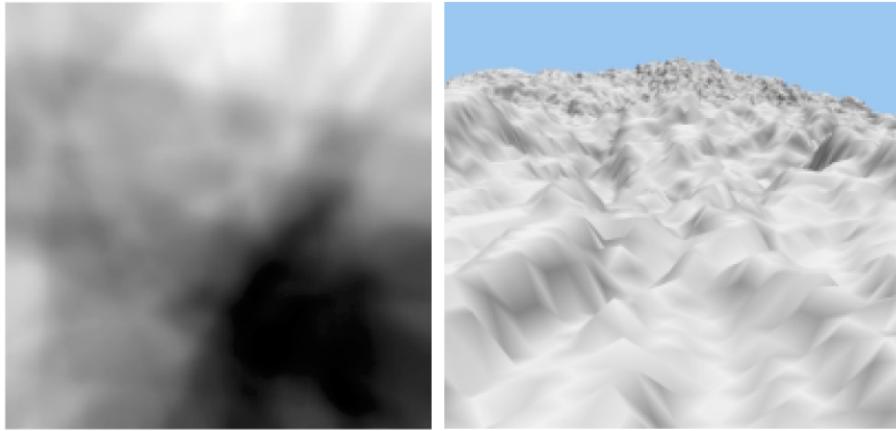


Figura 2.5: À esquerda, *heightmap* gerado a partir de falhas geológicas; à direita, o mesmo *heightmap* renderizado em 3D [Ribble, 2001]

### 2.2.3 Deposição de sedimentos

A geração por deposição de sedimentos simula a criação de relevo por fluxos de lava. Nessa abordagem, pontos de liberação de sedimentos são escolhidos e, em cada um desses pontos, um determinado número de sedimentos é depositado. Cada partícula depositada irá tentar chegar até o ponto mais baixo da região onde foi depositada, escorregando por cima das demais partículas. Se a partícula cair em um local no qual todas as posições à sua volta tem a mesma altura, então ela termina o movimento.

Variando-se os pontos de deposição de partículas e a quantidade de elementos depositados em cada ponto, é possível gerar terrenos visualmente aceitáveis. Novamente o tempo de execução do algoritmo é um problema; para cada partícula depositada faz-se necessário o cálculo de movimentação desse elemento, o que, dependendo do número de iterações, pode ter um custo computacional grande. Além disso, o algoritmo apresenta uma complexidade de implementação maior que os demais já citados. A figura 2.6 ilustra alguns *heightmaps* obtidos com o algoritmo de deposição de sedimentos.

### 2.2.4 Disposição do ponto médio

Essa técnica consiste em perturbar o ponto médio de um determinado segmento. Em uma abordagem 2D, por exemplo, dado um determinado segmento, o algoritmo acha o ponto médio desse segmento e, utilizando um número pseudo-aleatório, aumenta ou diminui a altura desse ponto em uma determinada quantidade. Em seguida, essa quantidade utilizada para perturbação é reduzida e o algoritmo é aplicado novamente aos dois novos segmentos gerados a partir da perturbação do ponto médio. A figura 2.7 ilustra o

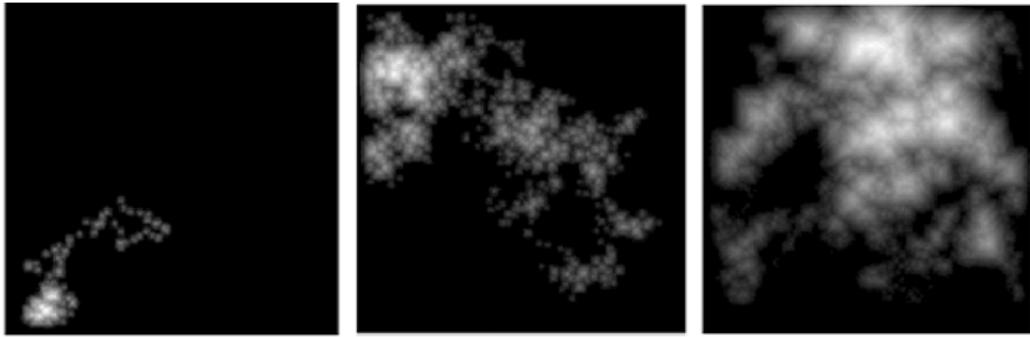


Figura 2.6: *Heightmaps* obtidos com o algoritmo de deposição de sedimentos. [Ribble, 2001]

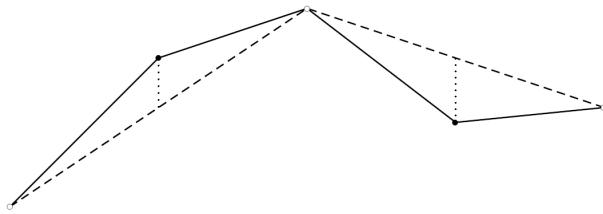


Figura 2.7: Geração de relevo pela disposição do ponto médio (perspectiva 2D) [Dollins, 2002]

funcionamento do algoritmo.

### 2.2.5 Ruído de Perlin

Essa técnica consiste em utilizar funções de ruído de Perlin para causar perturbações no *heightmap*. A utilização de uma função de ruído de Perlin num *heightmap* não produz, por si só, um relevo visualmente parecido com aqueles encontrados na natureza; a combinação de diversas camadas de ruido, porém, é capaz de produzir um efeito mais natural e convincente. Cada uma dessas recursões, chamadas de octavas, possui uma amplitude e uma frequência, e a combinação delas é comumente chamada de turbulência de Perlin. A figura 2.8 ilustra os resultados obtidos com essa técnica.

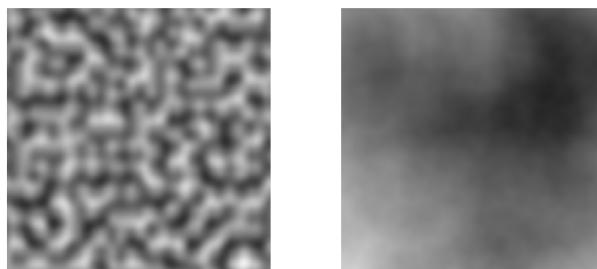


Figura 2.8: À direita, *heightmap* gerado a partir de ruído de Perlin com alta frequência. À esquerda, *heightmap* gerado com ruídos de menor frequência. [Häggström, 2006]

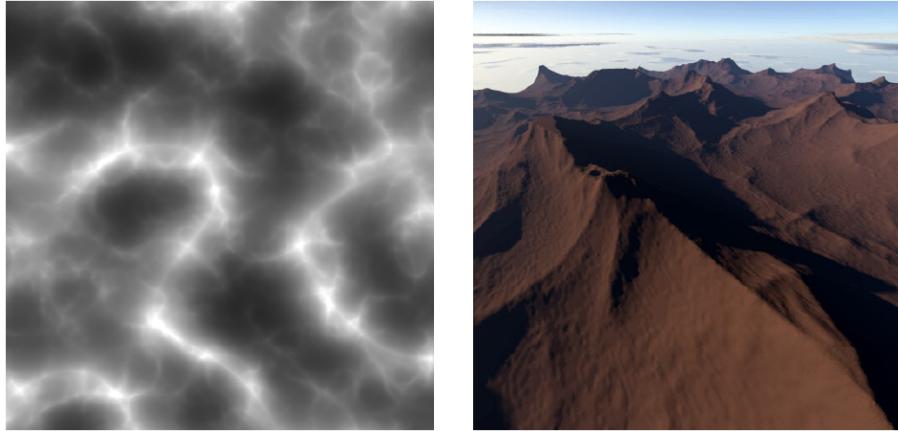


Figura 2.9: À esquerda, *heightmap* gerado a partir da técnica de ruído de Perlin *ridged*; à direita, o mesmo *heightmap* renderizado em 3D [Häggström, 2006]

As características do relevo podem ser ajustadas através da modificação do número de octavas e da frequência de cada uma delas. Quanto maior for a frequência da octava adicionada, maior será a quantidade de detalhes do relevo. Uma variação para a geração de relevo através de ruído de Perlin é a técnica conhecida como ruído de Perlin *ridged*, que consiste na utilização de funções com valores absolutos em conjunto com a função de ruído original para produzir um relevo com mais "cristas". Na abordagem original, no intervalo  $[-1, 1]$ , que é o domínio da função de ruído,  $-1$  indicaria um local muito baixo no mapa, ao passo que  $1$  indicaria o cume da montanha mais alta; na versão *ridged*, o algoritmo original é modificado para que os valores  $-1$  e  $1$  gerem relevos de baixa altitude, ao passo que os valores gerados ao longo do intervalo  $[-1, 1]$  são de grande altitude. A figura 2.9 ilustra os resultados obtidos com essa abordagem.

## 2.3 Mundos pseudo-infinitos

### 2.3.1 Cidade virtual

A geração de conteúdos procedimentais é um assunto antigo no campo da computação gráfica. A aplicação desse tipo de técnica na geração de um mundo virtual completo foi utilizada por Greuter et al. [2005], cujo objetivo era gerar uma cidade virtual que fosse visualmente interessante e composta por construções complexas, porém cada uma delas sendo criada a partir de elementos mais simples.

Na abordagem utilizada, dividiu-se o mundo virtual numa grade composta por diversos quadrados, chamados células. As coordenadas de localização de cada célula, em conjunto com uma semente global, são utilizadas como entrada para uma função de hash

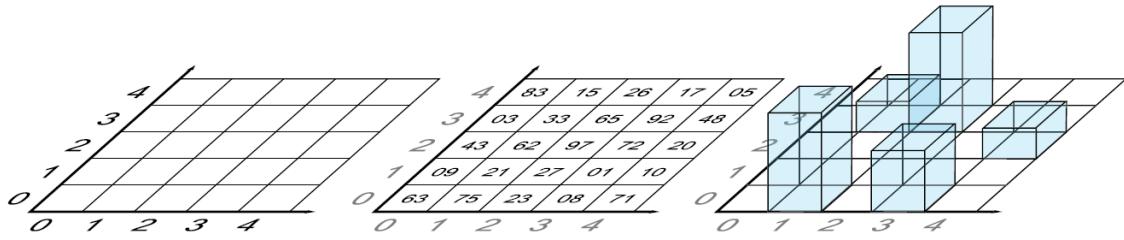


Figura 2.10: Mundo virtual dividido em células. Cada uma delas possui uma semente calculada com base em sua posição e com base uma semente global referente à cidade [Greuter et al., 2005]

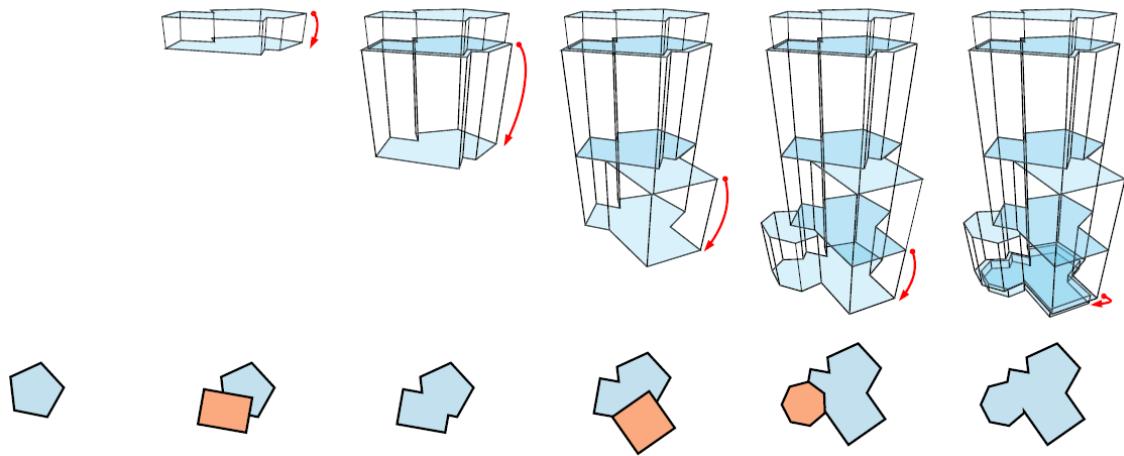


Figura 2.11: Geração de construções: cada andar é gerado pela mescla de polígonos escolhidos e centralizados aleatoriamente [Greuter et al., 2005]

[Wang, 2000]. O resultado dessa função é utilizado como semente para um pseudo gerador de números aleatórios e irá definir todas as características das construções que estão dentro da célula. Dessa forma, o conteúdo de uma célula é sempre o mesmo, independente de quanto o usuário caminhe pelo mundo virtual e faça a célula em questão entrar ou sair do seu campo de visão. A figura 2.10 ilustra a divisão do mundo virtual em células.

Cada uma das construções existentes é gerada pela mescla de polígonos simples escolhidos aleatoriamente. Utilizando um processo iterativo, partindo do topo até a base, em cada iteração o polígono escolhido é mesclado com o polígono anterior e, então, mais um nível (andar) é criado; esse processo garante que a construção, ao longo das iterações, cresça em altura e em largura de uma forma realista. Depois que a geometria da construção está pronta, ela é texturizada com janelas, sendo que o tipo da janela é escolhido aleatoriamente. A figura 2.11 ilustra a geração de construções.

Para garantir o uso racional de recursos computacionais, como memória e processamento, utilizou-se o conceito nomeado por eles de preenchimento por *view frustum*, que

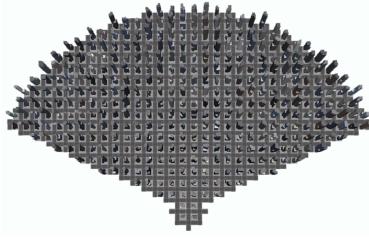


Figura 2.12: Campo de visão do usuário: apenas as células visíveis tem o seu conteúdo gerado [Greuter et al., 2005]

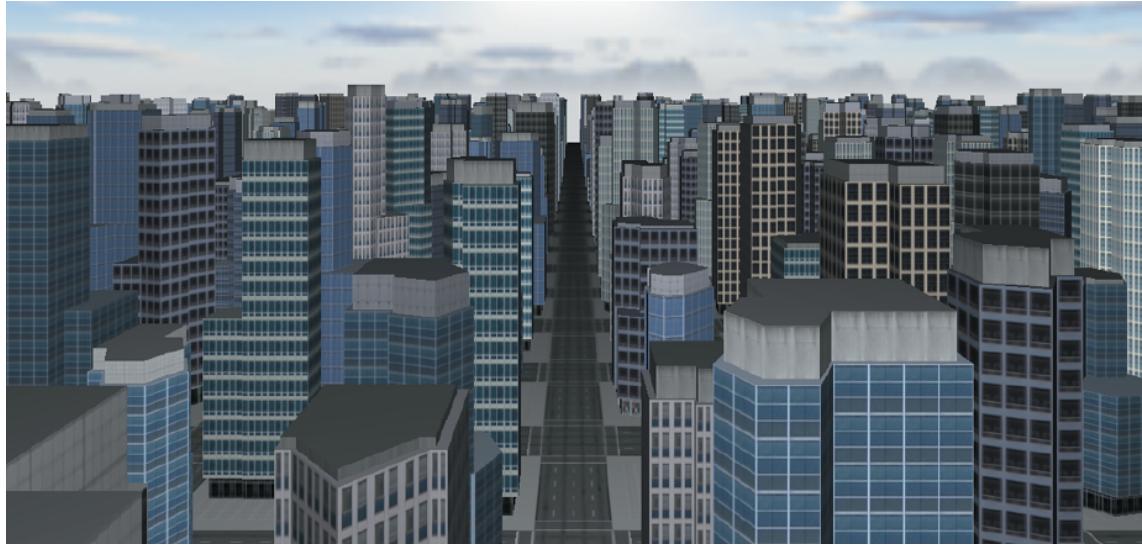


Figura 2.13: Exemplo de cidade virtual gerada [Greuter et al., 2005]

consiste em restringir à geração de conteúdo somente para as células que estão dentro do campo de visão do usuário. A medida que este anda pela cidade virtual, novas células vão sendo adicionadas ao campo de visão, sendo o seu conteúdo gerado de acordo com necessidade; quando a célula sai do campo de visão, ela é removida da memória e seus recursos são liberados. As células são posicionadas em loops quadrados ao redor do usuário e são consideradas pertencentes ao campo de visão se estão a uma certa distância do usuário e dentro de um ângulo de 120° de visão. A figura 2.12 ilustra o funcionamento do campo de visão descrito e a figura 2.13 mostra uma cidade virtual gerada.

### 2.3.2 Geração e texturização por camadas

Outro trabalho analisado foi o de Häggström [2006] para a construção da ferramenta SkyCastle [Häggström, 2009], uma *engine* para jogos online multijogador com suporte à geração procedimental de mundos virtuais. No referido trabalho, há um enfoque no problema de mundos virtuais cada vez maiores em jogos de computador e aplicações, o

que gera a necessidade de desenvolvimento de ferramentas capazes de ajudar na geração de conteúdos realistas para esses mundos. Para a geração de relevo, utiliza-se procedimentos parametrizados e sistemas baseados em fractais em uma abordagem de camadas: a aplicação adiciona um mapa de ruído (com amplitude reduzida) ao mapa de altura base em cada iteração. Os mapas de ruído são pré-computados e criados através de funções de ruído de Perlin, abordagem semelhante ao método de geração de relevo por deposição de sedimentos, que foi abordado em mais detalhes na seção 2.2.

Para a texturização da malha gerada, um dos métodos apresentados é a utilização de uma imagem com proporções muito grandes, que seria capaz de cobrir o mundo inteiro. Embora essa abordagem possa ser útil para cenários pequenos, ela não é viável para terrenos grandes ou mundos virtuais, uma vez que o tamanho da imagem poderia atingir proporções proibitivas. Para contornar esse problema, a abordagem de reticulados é sugerida [Cohen et al., 2003]; nessa abordagem, uma célula de textura é criada de tal forma que ao posicionar várias células, uma do lado da outra, o plano gerado apresenta uma texturização contínua. O resultado obtido com essa abordagem é aceitável, porém ele não é visualmente atraente para o usuário final, uma vez que o terreno apresenta uma continuidade que não existiria no mundo real. Para conseguir um resultado visual melhor, sugere-se a utilização de uma abordagem proposta por Lefebvre and Neyret [2003], que consiste na utilização de um conjunto de texturas pré-definidas juntamente com a texturização em reticulados já citada. O funcionamento do algoritmo resume-se em aplicar as texturas pré-definidas sobre um plano já uniformemente texturizado, porém utilizando transparência nas áreas não desenháveis das texturas pré-definidas. As áreas não desenháveis são calculadas com base em alguma característica do terreno, como a altura de cada ponto. Dessa forma, a sobreposição das texturas pré-definidas sobre o plano texturizado irá produzir uma paisagem menos homogênea, o que gera um resultado visual melhor. A figura 2.14 ilustra as texturas pré-definidas e a sua utilização na abordagem descrita.

Para ornamentar o mundo virtual, cita-se a utilização de plantas proceduralmente geradas através de três métodos principais: *L-System* [Przemyslaw and Lindenmayer, 1990], geração baseada em componentes [Lintermann and Deussen, 1998] e árvores parametrizadas [JasonWeber and Penn, 1995]. O algoritmo *L-System* consiste na geração de elementos a partir da interpretação de uma cadeia de caracteres, sendo que cada um desses caracteres representa uma estrutura geométrica ou operação (rotação, translação,

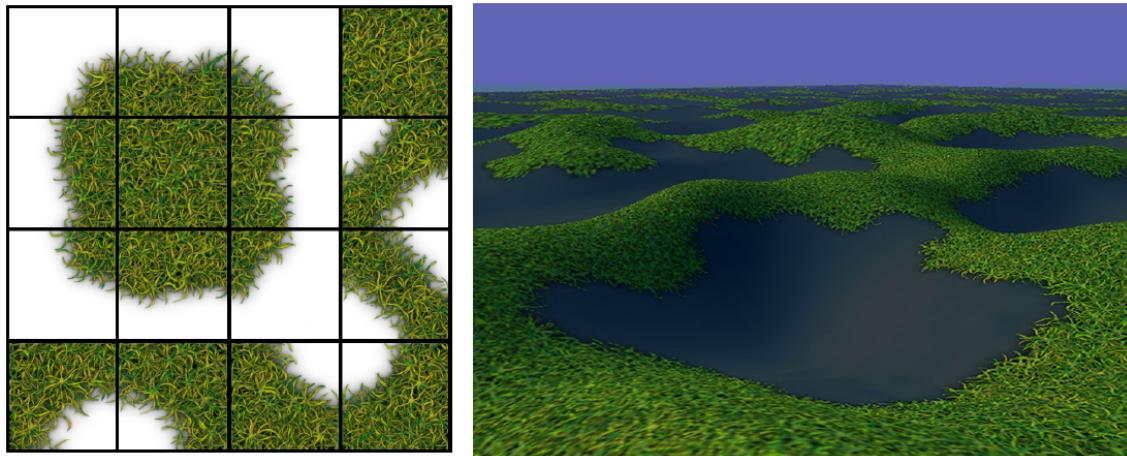


Figura 2.14: Terreno texturizado com duas camadas: bloco escuro e um conjunto de texturas em formato de borda com transparência [Häggström, 2006]

etc); a cadeia de caracteres resultante é obtida a partir da aplicação sucessiva de regras sobre uma cadeia base. O algoritmo de geração baseado em componentes consiste na interpretação de uma árvore de elementos, sendo cada um desses elementos modificável através de parâmetros; cada componente pode possuir filhos e, também, uma descrição de qual é o elemento que pode ser usado como folha da árvore. Por fim, a geração de árvores parametrizadas é conceitualmente semelhantes à geração por componentes, exceto que a geração é orientada a ramificações; cada ramificação da árvore corresponde à um nível de recursão, sendo o tronco da árvore o nível zero; quando um ramo sofre uma divisão, os filhos resultantes dessa divisão herdam algumas características do pai (como resolução), porém eles adquirem características próprias, como o ângulo de curvatura; ao final do processo, através de um estudo dos parâmetros corretos a serem utilizados (como a quantidade de ramos, o nível de recursão, etc), é possível que uma árvore completa seja gerada.

### 2.3.3 Planetas procedurais

Uma das abordagens analisadas trata da geração de planetas procedurais através de fractais combinados com outros métodos [Linda, 2007]. Nessa abordagem, o mundo gerado não é infinito, porém é esférico e simula a visualização do planeta Terra. Partindo da subdivisão recursiva de um octaedro, cria-se um mundo esférico que serve como base para a aplicação dos algoritmos de relevo. Utiliza-se o conceito de LOD (*level of detail* [Heckbert and Garland, 1994]) para garantir um bom desempenho da aplicação.

O relevo gerado é obtido através de diversas técnicas, cada uma com suas pecu-

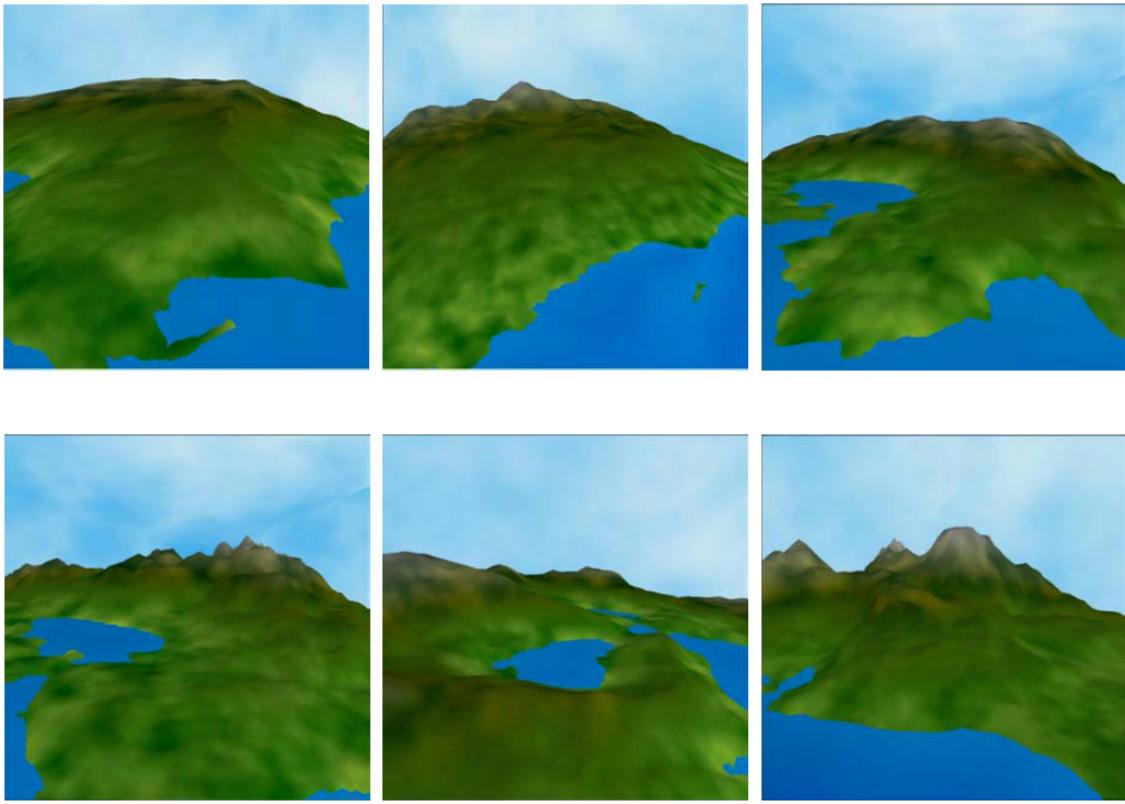


Figura 2.15: Diferentes relevos gerados pela utilização de fractais e suas combinações. [Linda, 2007]

liaridades e resultados. Dentre as técnicas estudadas, utilizou-se: geração por falhas aleatórias, disposição do ponto médio (incluindo uma variação multifractal) e ruido de Perlin (e suas diversas variações). A figura 2.15 ilustra os relevos obtidos com essas técnicas. Depois que a malha é gerada, utiliza-se uma combinação de técnicas para colorir os pontos da falha; as técnicas variam conforme o resultado desejado, porém todas são baseadas na interpolação de cores parametrizadas pela altura do ponto sendo analisado. Para garantir um aspecto mais real, utiliza-se um fator de aleatoriedade através de ruídos e turbulências.

Tendo em vista os diversos métodos para geração de relevo explorados, é importante salientar quais foram os resultados obtidos com a utilização de cada um deles frente ao problema de geração de continentes para um mundo esférico. Uma visão mostrando todos os planetas gerados é mostrada na figura 2.16. O algoritmo de geração por **fallas aleatórias** produz terrenos que parecem quebrados e diversas ilhas pequenas; de acordo com a análise feita, os resultados obtidos em sua grande maioria são muito aleatórios e na média o algoritmo produz um grande continente e um grande oceano. O algoritmo de **disposição do ponto médio** e sua variação multifractal geram continentes grandes e com-

pactos sem muitas ilhas pequenas, com as deformações geradas sendo uniformes e concentradas no centro dos continentes; embora a variação multifractal do algoritmo produza uma colorização aceitável (tanto em vista que há grandes variações de altitude), segundo a análise os resultados não são bons quanto os obtidos com ruído de Perlin. O algoritmo de **ruído de Perlin** e sua variação multifractal apresentam resultados visuais melhores em comparação com as técnicas citadas anteriormente; a comparação entre o planeta gerado apenas com ruído de Perlin e o planeta gerado pela sua variação multifractal mostra a considerável diferença existente entre essas duas abordagens: enquanto a primeira produz relevos com variações de altitude concentradas em determinados lugares, a segunda produz relevos mais realistas com uma distribuição menos concentrada em determinados pontos. Por último, as variações *ridged* da geração por **ruído de Perlin** produzem planetas com continentes estreitos e arredondados, algumas vezes gerando baías fechadas; da mesma forma que os resultados anteriores mostraram, uma abordagem multifractal gera um relevo com variações de altitude mais distribuídas ao longo dos continentes.

#### 2.3.4 Fractais refinados por erosão

Outro trabalho analisado foi a geração de terrenos procedimentais em tempo real com a utilização de fractais refinados por erosão [Olsen, 2004]. Conforme ressaltado, o aumento do poder de processamento de computadores domésticos possibilitou que jogos de computador façam uso de simulações de erosão em tempo quase real. Agregou-se à geração de relevo por fractais novas características decorrentes da aplicação de erosão, o que teve como resultado mapas mais reais e, ao mesmo tempo, customizáveis. Primeiramente, um mapa de altura foi criado a partir de um diagrama de Voronoi, sendo cada um dos vértices do diagrama chamados de *índices de funcionalidade*; o valor de cada célula no mapa de altura é obtido através de uma combinação linear das distâncias entre os pontos de funcionalidade mais próximos. Depois que o mapa de altura é gerado, ele é combinado com um algoritmo de geração de ruído para produzir montanhas menos pontiagudas. Para remover as linhas bem definidas criadas pelo diagrama de Voronoi, um filtro de perturbação é aplicado. A figura 2.17 ilustra o processo descrito.

Para a simulação de erosão no mapa de altura gerado, utiliza-se um método híbrido de duas técnicas de erosão: a termal e a hidráulica. A erosão termal consiste na simulação de sedimentos se desprendendo de áreas mais altas e deslizando para baixo, na base

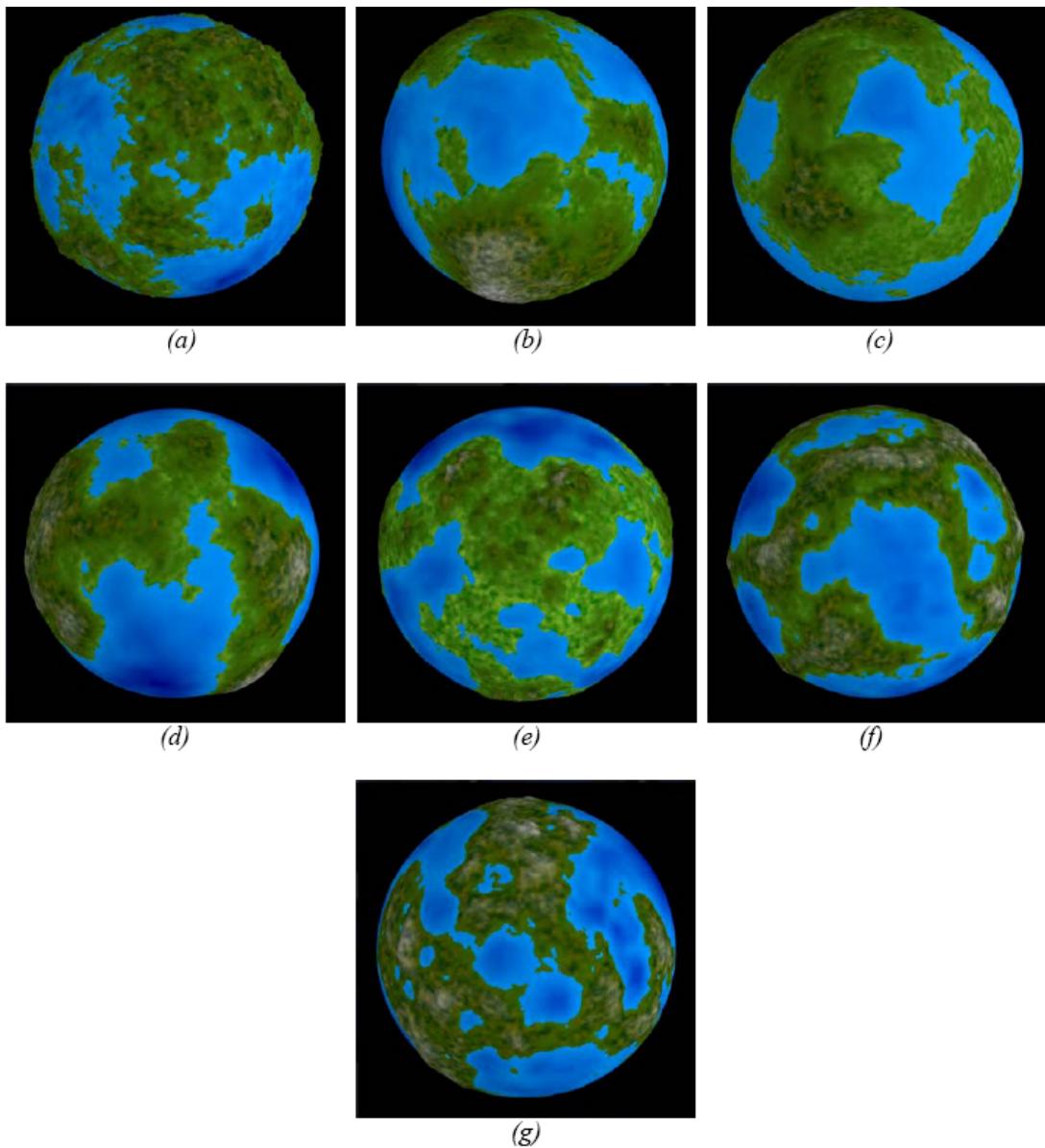


Figura 2.16: Planetas gerados com os diversos algoritmos. (a) Falhas aleatórias. (b) Disposição do ponto médio. (c) Disposição do ponto médio multifractal. (d) Ruído de Perlin. (e) Ruído de Perlin Multifractal. (f) Ruído de Perlin *Ridged*. (g) Ruído de Perlin *Ridged Multifractal*. [Linda, 2007]

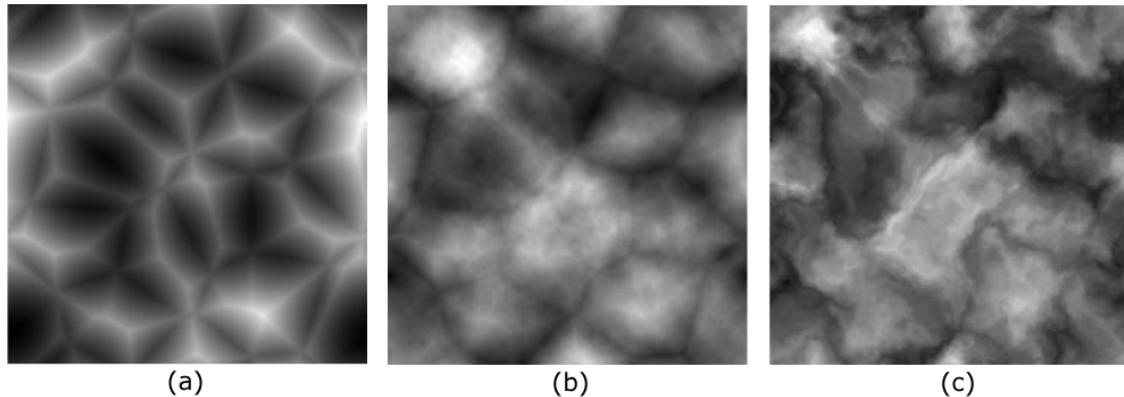


Figura 2.17: (a) Mapa de altura gerado a partir de um diagrama de Voronoi. (b) Aplicações de funções de ruído em (a). (c) Mapa resultante da aplicação de um filtro de perturbação em (b). [Olsen, 2004]

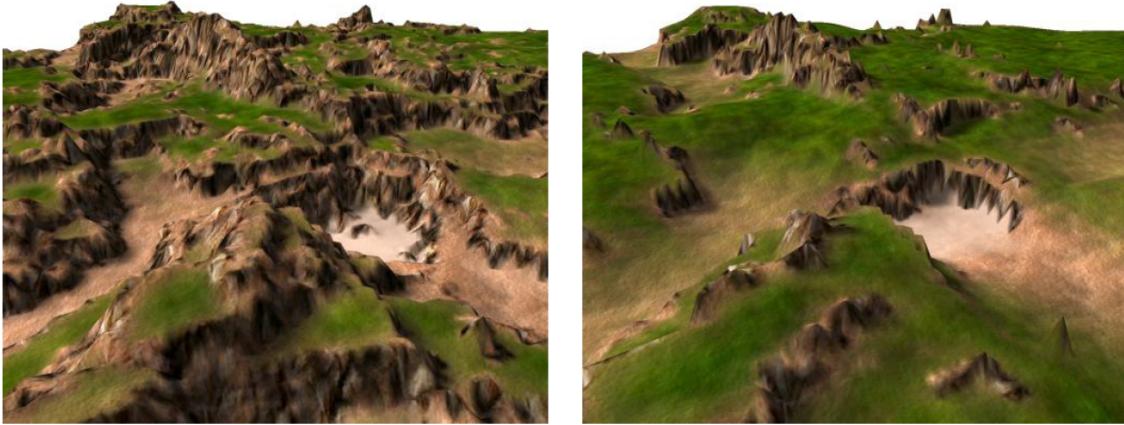


Figura 2.18: À direita, renderização de um mapa de altura sem a aplicação de erosão; à esquerda, o mesmo mapa com a aplicação de erosão. [Olsen, 2004]

do relevo. Aplicando uma série de otimizações sobre o algoritmo original proposto por Musgrave et al. [1989], obtiveram-se resultados satisfatórios que podem ser utilizados em uma abordagem em tempo real para jogos. A erosão hidráulica simula a deposição de sedimentos causada pelo transporte de materiais dissolvidos num fluxo de água corrente. Em uma comparação entre os dois métodos, a erosão hidráulica apresentou melhores resultados visuais, porém mesmo a sua implementação otimizada não foi mais rápida que a erosão termal otimizada. Criou-se, então, uma mescla entre as duas técnicas, alterando o método de erosão hidráulica para que o resultado ficasse mais próximo ao método de erosão termal. O resultado obtido é uma técnica capaz de gerar resultados visuais tão bons quanto o método da erosão hidráulica, porém com o desempenho do método de erosão termal, o que permite que a técnica seja utilizada em tempo real em jogos. A figura 2.18 ilustra o resultado final obtido.

### 2.3.5 Divisões estocásticas de uma *quadtree*

Outro trabalho analisado foi a geração procedural de mundos virtuais através de técnicas de subdivisão estocásticas [Dollins, 2002]. O objetivo foi criar um mundo virtual de grandes proporções, porém gerando o seu conteúdo sob-demanda de forma procedural e com multi-resolução. Para a geração do relevo do terreno, utilizou-se sub-divisões de uma *quadtree* em um processo recursivo, como ilustra a figura 2.19. A forma do terreno é definida pelo ponto médio de cada uma das células, que na figura são representados pelos círculos pequenos no centro de cada quadrado. Para cada nível de subdivisão, os novos pontos médios das células filho criadas são definidos em função das nove células pai que estão ao redor da célula sendo dividida. A altura dos demais pontos da malha, como os vértices que estão nos cantos das células, é definida pela interpolação dos pontos médios mais próximos. A cada camada de detalhamento, mais células são subdivididas, e a célula que sofreu a subdivisão é substituída pelas novas células filho geradas. Na figura 2.19, o quadrado localizado no centro representa uma célula que está sofrendo subdivisões e que será substituída pelas quatro células filho resultantes da subdivisão (as células filhos são os quadrados com círculos brancos no centro). Tendo em vista que as subdivisões são baseadas em cálculos estocásticos, necessitou-se de uma forma de geração de números aleatórios que pudesse prover uma sequência confiável de números ao longo de todo o processo de subdivisão, visto que são vários níveis de detalhes, cada um deles gerando várias células com diversos pontos. Para evitar repetições de números ao longo dos vários níveis de detalhamento, utilizou-se um gerador de números aleatórios alimentado por três elementos: as coordenadas X e Y da célula e o nível de detalhe sendo mostrado nesse momento; isso garante que um conjunto de números aleatórios seja gerado de forma satisfatória para cada um das células existentes no mundo virtual, independente do nível de detalhes exigido.

Para controlar a instanciação de cada célula, conforme a câmera se movimenta, novas células são instanciadas à frente da câmera e as células atrás da câmera, que estão muito distantes, são removidas da memória. Os nós pai sempre instanciam todos os seus filhos e os nós que estão dentro de um determinada camada (nível da árvore) só serão atualizados quando a câmera atingir o ponto médio da célula pai. À medida que a câmera se move para a direita, por exemplo, ultrapassando o ponto médio das células pai, novos nós vão sendo instanciados à direita (à medida que o nó pai entra no campo de visão), ao passo que

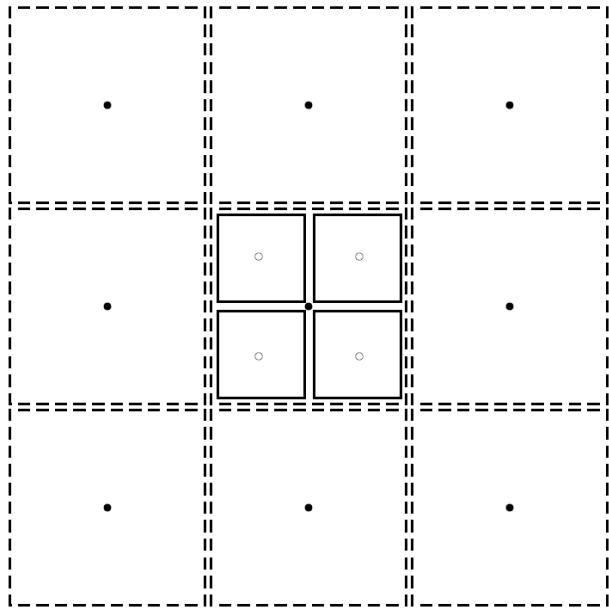


Figura 2.19: *Quadtree* utilizada para a geração do terreno [Dollins, 2002]

todos os nós pai que saem do campo de visão são removidos (junto com os seus filhos). A malha final é renderizada a partir das informações conditas nas folhas da árvore, sendo que a altura de cada um desses pontos na malha é definido pela interpolação de seu ponto médio com a vizinhança, como descrito anteriormente.

Para garantir o desempenho no cálculo dos pontos médios da vizinhança, buscou-se uma forma de pesquisar, com um tempo constante, os nós vizinhos duma determinada célula. Para encontrar os vizinhos imediatos a uma célula, basta que a relação de pai e filho da árvore seja utilizada, o que irá resultar em uma busca de apenas dois ponteiros (um do nó atual até o pai dele e outro do pai até o nó irmão). O problema está no cálculo para encontrar as células vizinhas mais distantes, que podem estar a uma distância arbitrária do nó atual, ou seja, vários ponteiros deverão ser consultados até que os vizinhos desejados sejam encontrados. Uma das soluções encontradas para essa busca é a utilização de uma tabela hash para indexar todos os nós de uma determinada camada (nível da árvore). Utilizando essa abordagem, o tempo de busca de qualquer nó de uma camada se tornaria constante. Outra solução encontrada é utilizar um vetor bi-dimensional para indexar os nós de uma determinada camada. Conforme apontado, quando uma camada deve ser atualizada (porque o ponto médio de uma célula pai foi atingido), a melhor abordagem é a utilização de um vetor; mesmo que o custo para atualizar todos os nós de um vetor seja mais alto do que apenas inserir uma nova entrada na tabela hash, a busca em vetor é

mais rápida do que o processo de calcular a chave de hash de cada célula a partir de suas coordenadas.

### **2.3.6 Gerador de mapas para planetas**

Outro trabalho analisado foi um gerador de mapas para planetas, que consiste na geração de mundos esféricos através da subdivisão recursiva de um tetraedro [Mogensen, 2009]. Embora o trabalho não seja apresentado como uma ferramenta gráfica na qual o usuário pode explorar um ambiente 3D, todas as informações geradas são parte de um mundo virtual completo, com continentes e oceanos altamente customizáveis. A ideia apresenta a criação de um mapa como resultado da projeção de pixels numa esfera. Cada pixel do mapa a ser gerado é projetado para uma esfera, sendo que o algoritmo que faz a projeção atribui uma altura ao pixel encontrado, o que gera um mapa com relevo.

Para renderizar um mapa, inicialmente encontram-se os pontos dele que são visíveis na superfície da esfera. Em seguida, para calcular a altura de cada um desses pontos, aplica-se o seguinte algoritmo:

- Inserir a esfera dentro do tetraedro.
- Cortar o tetraedro em dois tetraedros menores.
- Escolher em qual dos dois tetraedros está localizado o ponto sendo analisado.
- Repetir os passos anteriores até que o tetraedro seja pequeno o suficiente
- Usar a altura média dos vértices do tetraedro para calcular a altura do ponto sendo procurado.

No algoritmo de projeção analisado faz-se uso de um tetraedro para encontrar o ponto correspondente na superfície da esfera. A figura 2.20 ilustra o tetraedro inicial utilizado no processo.

Inicialmente a esfera é colocada dentro de um tetraedro irregular e informações sobre altura e uma semente para cálculos aleatórios é colocada em cada um dos vértices dele. Em seguida, ele é dividido em dois pelo plano formado entre o ponto médio da aresta mais longa e os dois pontos finais da aresta oposta ao ponto médio. Em suma, cada um dos dois tetraedros resultantes terão três vértices a partir do tetraedro original mais um novo vértice posicionado no ponto médio da maior aresta. A altura e a semente do novo vértice

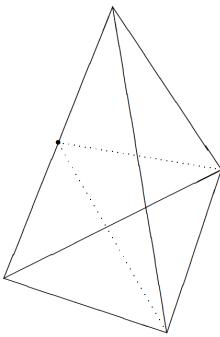


Figura 2.20: Tetraedro inicial utilizado no processo de projeção. [Mogensen, 2009]

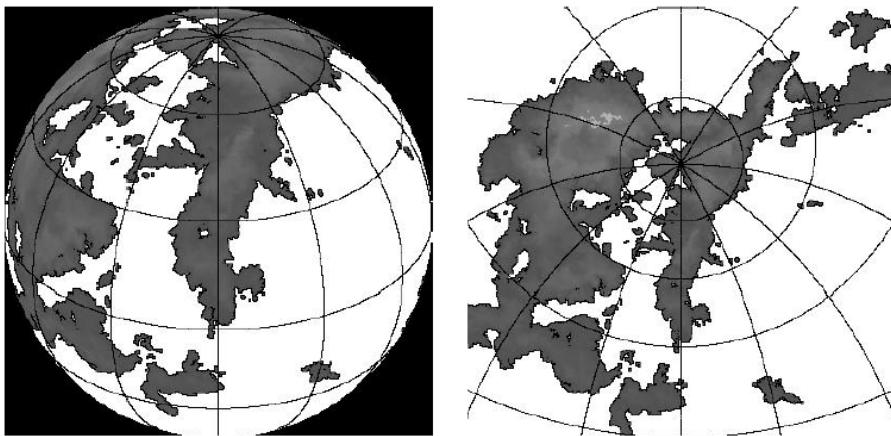


Figura 2.21: Mapas obtidos com a aplicação do algoritmo de subdivisão do tetraedro. [Mogensen, 2009]

são calculados a partir da altitude e semente dos pontos finais da linha que é dividida (maior aresta). O processo é repetido recursivamente para cada um dos tetraedros até que ele fique pequeno o suficiente para conter apenas o ponto desejado. A figura 2.21 ilustra alguns mapas obtidos com diferentes tipos de projeção.

Pode-se utilizar diferentes tipos de projeção para coleta dos pontos da esfera, como a de Mercator [ARRUMAR, 2007], que é uma das opções que o trabalho analisado disponibiliza. Além disso, é possível aplicar zoom ao mapa sendo gerado, bastando para isso aumentar o número de passos necessários para o cálculo de cada pixel. A figura 2.22 mostra dois mapas com a aplicação de zoom.

O mapa gerado apresenta continuidade ao longo dos continentes, uma vez que foi originado de uma esfera. Isso quer dizer que se o usuário seguir o trajeto de algum meridiano, ele irá retornar para o exato ponto de partida sem qualquer tipo de quebra de conteúdo ao longo do processo.

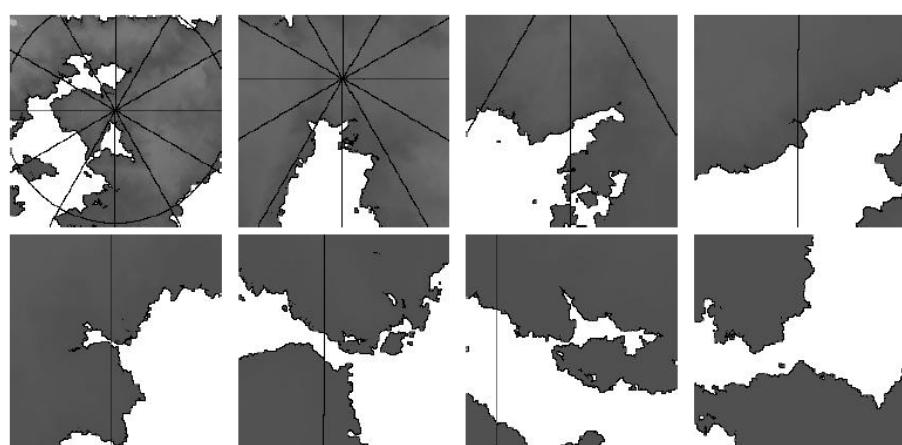


Figura 2.22: Zoom realizado sobre diferentes mapas. [Mogensen, 2009]

## 3 ESTRUTURA DA FERRAMENTA

### 3.1 Visão geral da ferramenta

O objetivo do presente trabalho é o desenvolvimento de uma ferramenta capaz de gerar mundos pseudo infinitos em tempo real. O emprego do termo "pseudo" faz-se necessário devido à limitação física dos computadores atuais, que apresentam um valor máximo possível para números inteiros (sejam eles de 32 ou 64 bits). Se não houvesse limitação física alguma, a ferramenta seria capaz de gerar, de fato, um mundo infinito. A ideia da ferramenta é permitir que programadores utilizem suas funcionalidades para a criação de terrenos em jogos 3D, principalmente MMOs, com a mínima intervenção humana no processo de geração. O tamanho dos terrenos a serem criados deve ser customizável, porém os mapas gerados devem ter como principal característica a magnitude de suas dimensões, que devem chegar à casa dos bilhões de pixels.

Depois que for inicializada, a ferramenta deve permitir que o usuário caminhe livremente pelo terreno criado. À medida que o usuário se move, a ferramenta adiciona novas paisagens dentro do campo de visão através de um processo de geração de conteúdo sob demanda. Se o usuário caminhar em direção ao norte por vários dias e, em seguida, quiser retornar para à sua primeira posição, ele deve ser capaz de ver, na viagem de volta, exatamente a mesma configuração de montanhas e paisagens que ele viu durante a viagem de ida. Além disso, a ferramenta deve ser capaz de gerar 1) continentes com configurações variadas, 2) diferentes tipos de relevo (montanhas, falésias, planícies, etc) e 3) costas que sejam convincentes ao usuário (como praias ao longo da borda dos continentes).

As seções seguintes explicam como a ferramenta, doravante denominada *Charack*, foi estruturada para permitir que os objetivos citados fossem atingidos. O texto está organizado de forma a apresentar uma abordagem *top-down*: parte-se da explicação sobre a geração de continentes (visão mais ampla) e termina-se com a explicação sobre a geração

de relevo para cada um dos vértices a serem desenhados na tela (visão mais específica e pontual).

### **3.2 Análise inicial**

Conforme apresentado na seção 2.3, existem diversos trabalhos na área de geração de mundos virtuais, sejam eles finitos ou infinitos. O trabalho desenvolvido por Greuter et al. [2005] cria uma cidade infinita que é apresentada ao usuário sob demanda à medida que esse passeia sobre o terreno. Esse foi o trabalho inicial de onde nasceu Charack, porém a ideia original foi alterada a fim de que mais conteúdos fossem criados (montanhas, planícies, continentes, etc.), além de ruas e prédios. A criação de conteúdo sob demanda, na qual o usuário só enxerga aquilo que está dentro do seu campo de visão, foi mantida da abordagem original.

O mundo procedural gerado por Linda [2007] aproxima-se muito do conceito buscado para a criação de conteúdo de Charack. No referido trabalho, um planeta esférico é criado a partir da divisão recursiva de uma forma geométrica e, sem seguida, funções de ruído são aplicadas à malha para a geração de conteúdo. Não existem abordagens separadas para a criação dos continentes e do relevo dentro deles, ou seja, os continentes são resultado da inundação do relevo criado pelas funções de ruído; como consequência, a costa dos continentes também não possui um tratamento diferenciado. A costa surgirá baseada na altura do nível do mar e da quantidade de ondulações presentes no relevo. O mundo criado é esférico e o jogador pode aproximar ou distanciar a câmera do terreno, porém o conteúdo não é gerado sob demanda. Charack foi criado a partir de uma evolução dessa ideia, porém aplicando-se algumas limitações. O mundo criado por Charack trata de forma completamente diferente a geração de continentes, do relevo dentro deles e da costa. Cada um desses elementos possui sua forma de atuar, que permite ao programador customizar detalhes pontuais sem interferir nas demais áreas do mundo. Esse é o caso da criação de praias mais longas na costa sem modificação da estrutura básica do relevo ou do continente na que essa praia está inserida. Além disso, Charack gera o conteúdo sob demanda num mundo com proporções maiores, entretanto sem o uso da abordagem esférica; o mundo criado está contido dentro de um plano.

Em outro trabalho analisado, o terreno criado sofre ação de erosão [Olsen, 2004]. Charack não utiliza esse conceito, porém faz uso de uma ideia semelhante para fazer o

casamento dos conteúdos gerados nos diferentes módulos da aplicação. Um exemplo simples é o encontro do relevo do continente com o oceano, no qual a altura de cada vértice é atenuada para que um relevo de praia seja criado.

Por fim o trabalho criado por Dollins [2002] aproxima-se muito do objetivo proposto. Nesse trabalho, um mundo virtual de proporções gigantescas é criado e o seu conteúdo é gerado sob demanda à medida que o usuário se move. O relevo é criado de forma parametrizada e com multi-resolução, ou seja, quanto mais próximo o usuário estiver de um determinado local no mundo virtual, maior será a quantidade de detalhes nesse local. Também não são apresentadas abordagens separadas para a criação de continentes/costa/relevo. O funcionamento da geração de relevo de Charack baseia-se nas ideias desse trabalho (geração sob demanda e parametrizável), porém a criação de continentes e costa é completamente diferente.

Analizando-se cada um dos trabalhos citados, é possível constatar que mundos virtuais de diferentes proporções são gerados, porém em nenhum deles há uma preocupação com a separação dos métodos de geração dos conteúdos (continentes, relevo, costa). Embora existam variações na forma como o relevo é criado, a geração de continentes fica a cargo da ação de um plano de água interceptando o plano de terra. Essa abordagem permite que esforços sejam concentrados na geração de conteúdo do núcleo do mundo, que são as faixas de terra, porém ela tem um aspecto simples em relação aos demais elementos. No que se refere à heterogeneidade de conteúdo, como diferentes configurações de continentes e costa, o resultado final é pobre. A principal ideia e contribuição de Charack é a geração dos diferentes tipos de conteúdo de forma separada, com abordagens agressivas e pontuais em cada um dos tópicos, entretanto criando um resultado único que contempla todos os tópicos. Como a grande maioria dos trabalhos analisados foca na criação de relevo, entende-se que a contribuição de Charack será maior se os esforços forem focados no que diz respeito à geração de continentes e costa.

### **3.3 Estrutura básica**

Para a criação do mundo virtual nos moldes pretendidos, adotou-se uma estratégia hierárquica para a geração de conteúdos. O fluxo de dados de Charack tem início na visão mais ampla do mundo, que são os continentes, evoluindo ao longo do processo até terminar na visão mais específica e pontual, que é a geração de conteúdo para cada um

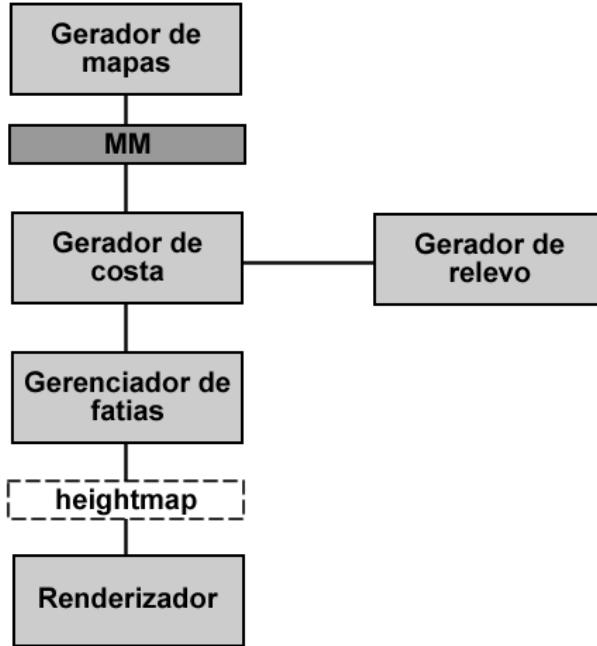


Figura 3.1: Estrutura básica para geração de conteúdo

dos vértices que serão desenhados na tela. A figura 3.1 ilustra a estrutura básica.

### 3.3.1 Gerador de mapas

No topo da cadeia encontra-se o gerador de mapas, que é responsável por criar os continentes que existirão ao longo do mundo virtual. Esse módulo é um encapsulamento da solução criada por Mogensen [2009], que foi descrita na seção 2.3.6. Quando a ferramenta é iniciada, ela utiliza uma semente definida pelo usuário para gerar todos os continentes que existirão no mundo virtual. Depois que os continentes são gerados, as informações sobre os tipos de terreno (terra, agua e costa) são armazenadas em uma matriz, chamada **macro-matriz** (MM), que é utilizada para consulta pelos demais algorítimos da ferramenta.

### 3.3.2 Gerenciador de fatias

Na metade da hierarquia existe o gerenciador de fatias (GF). Ele é responsável por recortar um trecho do mundo virtual (campo de visão do usuário) e, em seguida, fornecer essa informação ao renderizador através de um *heightmap*. O trecho recordado do mundo virtual é descrito como uma **malha regular**. **Para obter as informações necessárias para a criação do heightmap, o gerenciador de fatias faz uso do gerador de costa (GC), que por sua vez faz uso da MM e do gerador de relevos (GR).**

No contexto do GF, não existe informação sobre terra ou água, ele apenas tem conhecimento de um conjunto de vértices do mundo virtual e qual a altura de cada um deles. Utilizando a posição do usuário como guia, o GF realiza o corte do mundo virtual e, para cada um dos vértices coletados, ele faz consultas ao GC para descobrir qual a altura desse vértice.

### 3.3.3 Gerador de relevo

O gerador de relevo é responsável pela definição da altura que cada vértice que mundo virtual possui. Para calcular esse valor, o módulo utiliza uma combinação das técnicas descrita na seção 2.2, porém a grande maioria dos cálculos está baseada em funções de ruído de Perlin.

### 3.3.4 Gerador de costa

O gerenciador de costa (GC) irá mapear cada vértice do gerenciador de fatias para a MM e descobrir qual é o tipo de terreno associado. Se o vértice em questão é mapeado para um local na MM que é descrito como água, então o GC atribuirá uma altura igual ao nível do mar para o vértice e, em seguida, irá retorná-lo para o GF. Se o vértice em questão é mapeado para um local descrito como terra (simples), então o GC utilizará as informações do GR para descobrir qual a altura desse vértice, o que definirá qual relevo ele representa. Por fim, se o mapeamento terminar em um terreno descrito como costa, então o GC utilizará sua própria estrutura (em conjunto com a MM) para definir o relevo do vértice.

O mundo virtual gerado possui, tecnicamente, altura e largura definidas pelo tamanho máximo que um inteiro pode suportar, o que torna fisicamente inviável a geração de uma MM com tais proporções. Uma vez que a MM é menor que o mundo virtual, uma entrada  $(i, j)$  dela corresponde a diversos vértices dentro do mundo virtual. A figura 3.2 ilustra o mapeamento da MM para o mundo virtual.

Observando-se a figura é possível constatar que quanto menor for a MM, mais vértices do mundo virtual serão mapeados para uma mesma entrada. Se o mundo virtual tiver dimensões  $1000 \times 1000$  e a MM  $10 \times 10$ , por exemplo, isso quer dizer que para cada vértice da MM existem 100 vértices no mundo virtual; se um desses vértices da MM for do tipo costa, então existirá uma área de  $100 \times 100$  vértices no mundo virtual que precisa ser uma costa. É exatamente nesse ponto que o GC atua

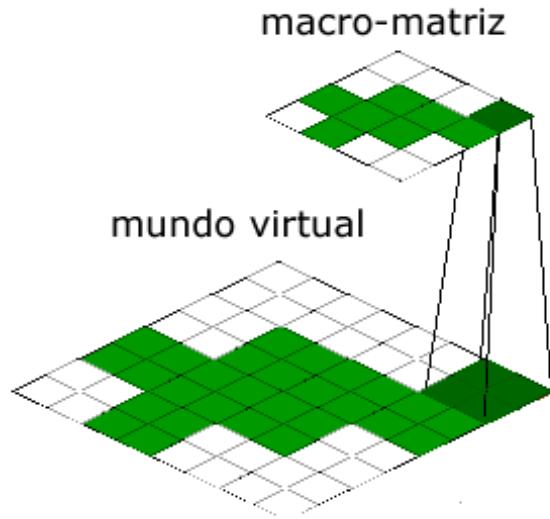


Figura 3.2: Mapeamento da MM para o mundo virtual: cada vértice da MM é mapeado para diversos vértices no mundo virtual.

**no que diz respeito à geração de conteúdo para a costa; quando qualquer um dos vértices dessa área em especial chegar aos cuidados do GC, ele irá trabalhá-los e retornará ao renderizador informações diferentes das originais, o que permitirá que uma costa seja desenhada na tela, não uma área inteiramente preenchida por terra ou por água.**

### 3.3.5 Renderizador

**O renderizador é responsável por renderizar na tela o *heightmap* criado nas demais fases do processo. O resultado final é desenhado como uma malha de triângulos que é texturizada conforme a altura de cada vértice.**

## 4 IMPLEMENTAÇÃO

A ideia original do trabalho foi desenvolver um mundo virtual. Dentre as funcionalidades previstas, encontravam-se a divisão do terreno em relevos específicos (desertos, florestas, planícies, etc), cidades/vilas, caminhos entre as cidades, rios e cadeias montanhosas.

Quando o planejamento foi finalizado, a complexidade de determinadas funcionalidades previstas tornou proibitiva a sua implementação. A grande maioria dos problemas encontrados é uma consequência da abordagem de geração dinâmica de conteúdo sob demanda (a medida que o usuário se move, novos elementos são colocados na tela). A Figura 4.1 ilustra o problema da geração de conteúdo sob demanda.

Partindo do fato que o usuário só consegue enxergar aquilo que está dentro do seu campo de visão, todos os algoritmos de geração de conteúdo, seja para relevo, caminhos ou cidades, precisam levar em consideração única e exclusivamente as informações que estão disponíveis dentro desse campo. Essa abordagem é eficiente para a utilização racional de recursos (processar somente o que o usuário está vendo), porém ela aumenta a complexidade dos algoritmos envolvidos na ferramenta.

Para o algoritmo de geração de cadeias montanhosas, por exemplo, não é possível determinar onde a cadeia termina, visto que o mundo fora do campo de visão tecnicamente não existe ainda e será gerado conforme o usuário avança pelo terreno. Uma abordagem seria utilizar uma função matemática que descrevesse a cadeia montanhosa, porém essa função não deveria depender de um ponto de início e fim, porque eles poderiam inexistir em um determinado momento. Se a função de geração de cadeias montanhosas não dependesse de um ponto de início e fim, ela precisaria, ao menos, depender da posição do usuário no mundo virtual para que o conteúdo correto fosse gerado. Depender de uma localização implicaria que a cadeia montanhosa gerada pela função fosse pré-posicionada

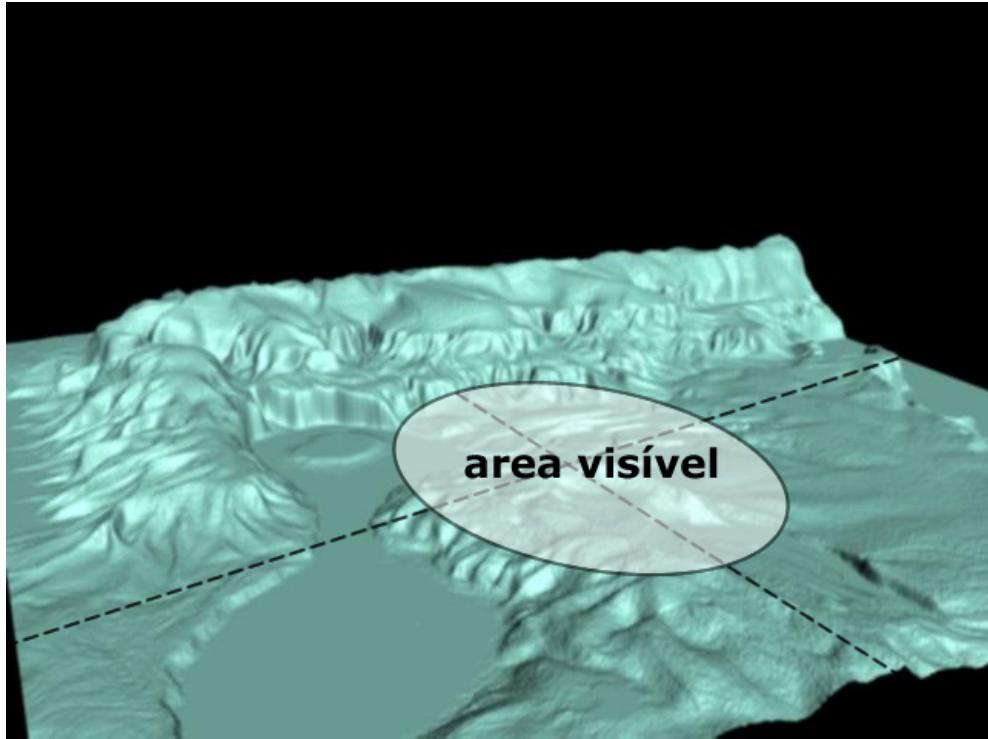


Figura 4.1: Problema da geração de conteúdo sob demanda.

no mundo virtual, o que iria contra o conceito de geração de conteúdo sob demanda.

Além disso, os algoritmos são sensivelmente afetados pelo fato de que as informações que eles recebem em um determinado instante podem desaparecer por completo na próxima iteração, visto que o usuário pode se mover e mudar o conteúdo do campo de visão. Utilizando o exemplo da geração de cadeias montanhosas, uma montanha poderia sofrer uma alteração em sua composição de forma abrupta, apenas porque os pontos que estavam sendo utilizados para a geração do relevo mudaram.

Para contornar esses problemas e focar os esforços de desenvolvimento em soluções pontuais, a geração do mundo virtual foi dividida em três grandes etapas: terreno infinito, continentes e relevo. A geração de conteúdo sob demanda afeta de forma diferenciada cada uma dessas etapas e a descrição da implementação de cada uma delas, junto com os problemas associados, é descrito nas seções seguintes.

## 4.1 Terreno infinito

A base para a geração do mundo virtual proposto é a possibilidade do usuário poder andar, de forma infinita, sobre a superfície do mundo e, conforme anda, visualizar novos conteúdos. A medida que o usuário anda, a ferramenta precisa ser capaz de identificar em

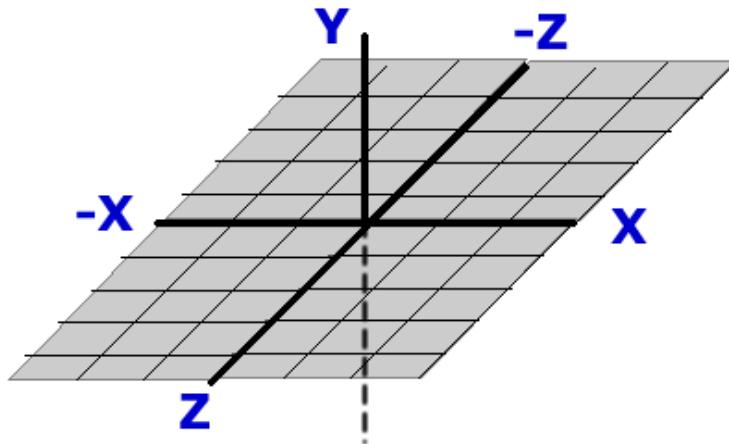


Figura 4.2: Organização do sistema de coordenadas do mundo virtual

qual local do mundo o observador se encontra para então gerar os conteúdos à sua volta.

Para solucionar esse problema, utilizou-se uma variação da técnica descrita por Greuter et al. [2005]. Na abordagem em questão, o mundo virtual pseudo-infinito é dividido em células quadradas (quadras da cidade) e, à medida que o usuário anda, as células são adicionadas e/ou removidas do campo de visão. O conjunto das células forma uma malha regular de polígonos. Cada célula possui um conteúdo próprio e auto-contido, ou seja, a célula não precisa de informações de vizinhos para gerar o seu conteúdo. Isso garante que as células não entrem em uma dependência recursiva infinita entre elas para conseguirem gerar o seu conteúdo. Além disso, essa abordagem é vantajosa para garantir o uso racional de recursos, uma vez que só serão carregados para a memória os blocos que o usuário realmente consegue ver.

Para o posicionamento do usuário no mundo virtual, os autores utilizam um vetor 3D no formato  $(x, y, z)$ . Conforme o usuário se move horizontalmente pelo mundo, as coordenadas  $x$  e  $y$  são atualizadas. Se o usuário se move verticalmente, a coordenada  $Z$  é alterada. A abordagem utilizada para a ferramenta dessa dissertação baseou-se nesses conceitos. A figura 4.2 ilustra a organização do mundo virtual.

A origem do mundo virtual é o ponto  $(0, 0, 0)$  e os eixos que definem o plano horizontal são o  $X$  e  $Z$ , sendo a altura controlada pelo eixo  $Y$ . A distância máxima que o usuário consegue percorrer em qualquer um dos eixos é o número máximo suportado por um inteiro de 32 bits com sinal.

O que o jogador consegue ver na tela em um determinado momento é um pedaço do mundo virtual existente. Esse pedaço foi chamado de *view frustum*, ou campo de visão.

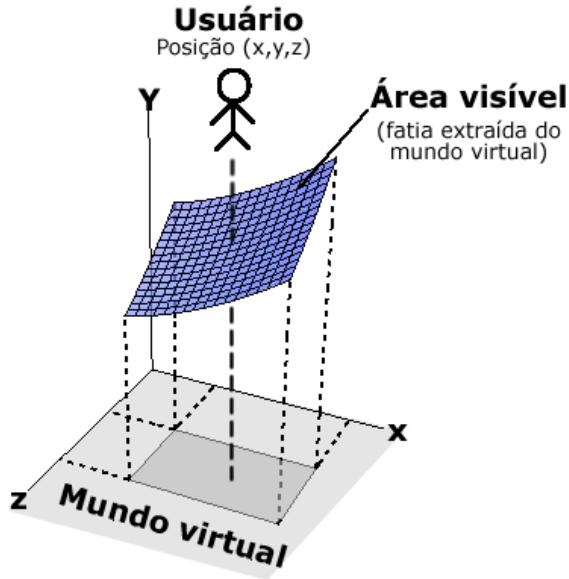


Figura 4.3: Campo de visão do usuário: a área visualizada corresponde a uma fatia do mundo virtual existente.

Diferentemente do que foi feito em Greuter et al. [2005], no qual o campo de visão é um cone, o campo de visão da presente ferramenta é um quadrado centrado no usuário. A figura 4.3 ilustra o funcionamento do campo de visão. A partir da posição ( $x$ ,  $y$ ,  $z$ ) do usuário, a ferramenta calcula qual é o conteúdo visualizável ao redor desse ponto e, então, "recorta" essa fatia do mundo e a desenha na tela. Ao chegar na borda limite do mundo, que pode ser a distância máxima de um eixo, por exemplo, o usuário é impedido de avançar e nenhum conteúdo é mostrado além da borda limite. O campo de visão em forma de quadrado centrado no usuário foi escolhido ao invés do campo em cone por questões de desempenho. Mesmo considerando-se que visão em cone permite a geração seletiva do conteúdo que está sendo visualizado pelo usuário, o custo para a geração desse conteúdo na ferramenta é muito alto, tanto em vista os diversos algoritmos e passos necessários para a criação dos dados. Uma visualização em cone seria proibitiva porque à cada movimento da visão do usuário, novos pontos teriam de ser calculados, e pontos já calculados teriam de ser descartados. Uma visão em forma de quadrado centrado no usuário permite que a ferramenta só dispare uma nova sequência de geração de conteúdo quando o usuário se aproximar da borda do campo de visão, momento no qual o ambiente ao redor do usuário é recalculado. Assumindo que a área visível é grande o suficiente para entreter o usuário, enquanto esse se movimentar próximo ao centro do quadrado, nenhum cálculo novo de geração de conteúdo será realizado.

Inicialmente planejou-se a utilização de quadrados para dividir o mundo virtual em células, conforme é feito na outra abordagem citada anteriormente. A utilização de células garantiria que o mundo virtual fosse subdividido em blocos menores, o que viabilizaria um melhor controle sobre o que o usuário consegue ver no seu campo de visão e, também, um melhor controle sobre a geração de conteúdo. O maior problema encontrado nessa abordagem, que foi a razão pela qual ela foi abandonada, é a dependência que as células precisam ter entre si para que o terreno infinito seja gerado.

Utilizando o exemplo da geração de relevo, que é tratada em mais detalhes na seção 4.2, se o mundo virtual fosse dividido em células, cada uma delas deveria possuir um relevo perfeitamente nivelado com a célula vizinha, caso contrário o relevo gerado teria diversos "degraus". Analisando o problema um pouco mais a fundo, no caso da geração de montanhas, por exemplo, se na metade de uma célula a ferramenta decidisse que uma cadeia montanhosa deveria começar, a célula vizinha deveria obrigatoriamente ter a continuação dessa cadeia montanhosa, caso contrário a montanha em questão seria fatiada pela metade. Uma das formas de corrigir esse problema é adicionar um nível mínimo de dependência entre as células: o conteúdo de uma célula é gerado com base nas informações da própria célula e também com base em alguma "dica" da célula vizinha. No exemplo da cadeia montanhosa, a célula vizinha ao começo da cadeia saberia que o conteúdo que ela deve gerar é a continuação da cadeia montanhosa, visto que a sua célula vizinha possui o começo da cadeia.

Essa dependência de conteúdo gera um encadeamento recursivo infinito entre as células. Se a célula A, por exemplo, for gerar o seu conteúdo, ela irá fazer isso com base nas suas informações e também com base nas informações de sua célula vizinha, B. A célula B, por sua vez, só poderá informar a A o seu próprio relevo quando ela o gerar; para gerar o seu relevo, ela precisa das suas informações e das informações da sua vizinha, C, e C precisa de D e assim por diante. Dessa forma, para gerar o conteúdo de A, a ferramenta teria que obrigatoriamente percorrer todas as células do mundo virtual.

Uma possível saída para esse problema da recursão infinita é definir um nível de consulta de informações. Embora essa abordagem limite o nível de consulta entre as células vizinhas, ela não soluciona o problema de continuidade de conteúdo. Se o conteúdo de A for gerado com seis níveis de recursão, por exemplo, quando o usuário se mover, novas células vizinhas serão consultadas para a geração do conteúdo; se a última célula que

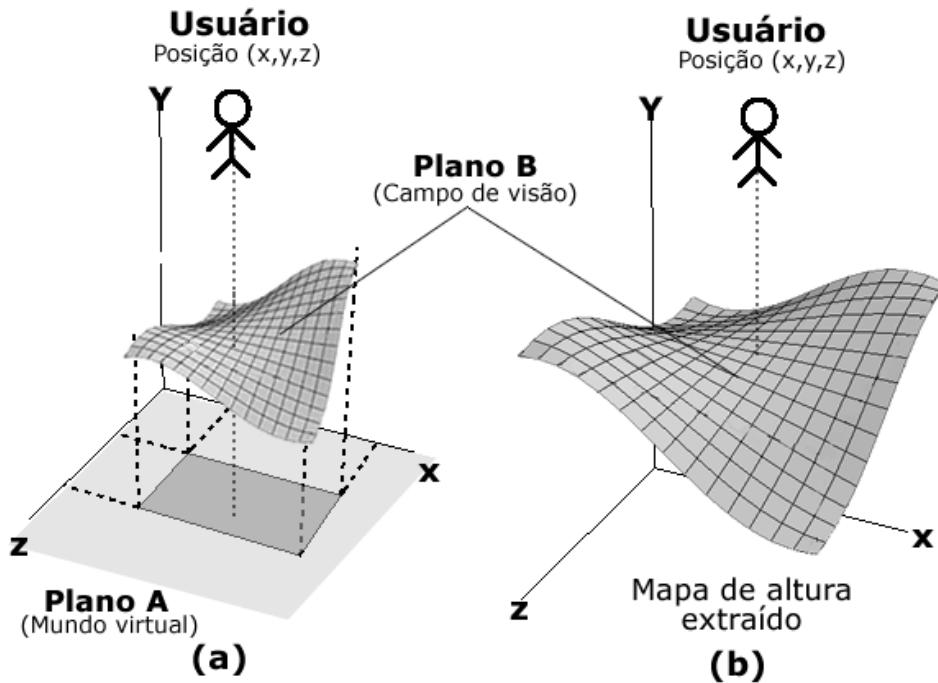


Figura 4.4: Mapeamento das coordenadas do mundo virtual para as coordenadas de desenho da tela

A consultou foi G, depois que o usuário se mover, G terá novas informações para o seu relevo, porque agora ela pode solicitar informações de suas vizinhas. Isso fará com que todas as células dependentes de G mudem o seu conteúdo, o que resultaria em um relevo diferente a cada movimentação do usuário. Em virtude da complexidade descrita e dos problemas mapeados, a abordagem de divisão do mundo virtual em células foi abandonada e substituída pelo modelo de campo de visão quadrado.

#### 4.1.1 Renderização de conteúdo

Depois que o modelo de controle da geração de terrenos infinitos foi definido, iniciaram-se os trabalhos de visualização das informações na tela. Embora o mundo virtual tenha coordenadas pseudo-infinitas, o máximo de conteúdo que o usuário enxerga na tela é a área do campo de visão, que possui sempre o mesmo tamanho e coordenadas de desenho. A figura 4.4(a) ilustra as coordenadas envolvidas no desenho do conteúdo do campo de visão.

O plano A representa o mundo virtual gerado pela ferramenta, enquanto o plano B representa a fatia do mundo virtual que o usuário consegue visualizar. A medida que o usuário se move, o centro do campo de visão é alterado e o usuário passa a ver novos conteúdos. Independente da movimentação que o usuário faça, o plano B pode sempre ser

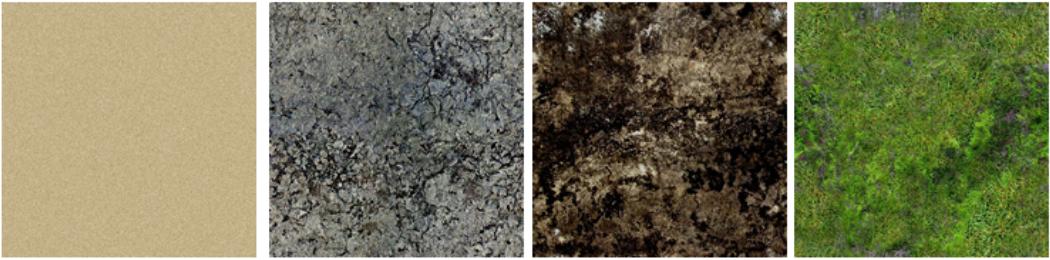


Figura 4.5: Conjunto de imagens utilizadas para texturização do terreno

mapeado como se estivesse na origem, porque ele é apenas uma fatia do mundo virtual. O que a ferramenta faz para desenhar esse conteúdo na tela é mapear essa faria para um *heightmap*. Depois que o mapa de altura é definido, as coordenadas dos eixos X e Z do mundo virtual que foram utilizadas durante a extração não são mais relevantes para o processo de desenho. Dessa forma, depois de extraído, o mapa de altura possui sempre as mesmas coordenadas nos eixos X e Z, que são as coordenadas de desenho da tela. A única informação do mundo virtual que é mantida é a altura de cada um dos pontos da malha do mapa de altura, conforme ilustra a figura 4.4 (b).

Para texturizar o conteúdo renderizado, utiliza-se um conjunto de imagens gerenciadas através da linguagem de *shaders* GLSL [OpenGL, 2007]. Baseado na altura do pixel sendo desenhado, calcula-se o peso que cada uma das texturas possui naquele local e, em seguida, cria-se uma interpolação delas. Cada uma das texturas do conjunto possuem um intervalo de altura na qual podem atuar: areia para alturas baixas, pedra para alturas médias, cascalho para alturas intermediárias e grama para as alturas maiores que as já estipuladas. Um exemplo de funcionamento desse método é a texturização da areia da praia. Nesse caso, todos os pixels da praia possuem uma altura que faz com que a textura de areia tenha o peso máximo, ao passo que as demais alturas tenham um peso zero. O resultado disso é que para os pixels da praia a interpolação do conjunto de texturas produz uma textura que é composta quase integralmente por areia. À medida que a altura da praia aumenta, o peso da textura de areia diminui e o peso da textura de pedra aumenta. O resultado produzido é uma transição suave entre a areia da praia e as pedras da costa. A figura 4.5 demonstra o conjunto de texturas existentes.

## 4.2 Relevo

A ideia original do presente trabalho foi desenvolver um mundo virtual capaz de apresentar diversos tipos de relevos, como cadeias montanhosas, planícies, vales, desertos, florestas, etc. A complexidade associada à geração de cada um desses elementos varia conforme o nível de realismo esperado e o nível de dinamismo do conteúdo gerado. Quanto mais presente for o conceito de geração de conteúdo sob demanda, mais difícil torna-se a tarefa de gerar conteúdos conexos e sem falhas abruptas na malha de relevo. As seções seguintes apresentam os problemas encontrados e a solução escolhida para a geração de relevo.

### 4.2.1 Problemas na geração de relevo sob-demanda

Mantendo-se a meta de gerar o conteúdo da forma mais dinâmica possível (sem elementos pré-posicionados, por exemplo), o primeiro grande problema encontrado durante o desenvolvimento do relevo do mundo virtual foi a forma como ele deveria ser gerado. Considerando que a ferramenta é capaz de criar um mundo pseudo-infinito, a geração de uma malha regular tão grande, de uma só vez, é computacionalmente inviável; a quantidade de vértices que precisariam ser armazenados tornaria o consumo de armazenamento da aplicação muito grande, o que poderia ser um empecilho para a utilização da ferramenta. Além do problema de armazenamento, outro tópico importante que foi considerado é a geração de conteúdo contínuo, ou seja, um relevo que não tenha falhas abruptas de continuidade, como uma cadeia montanhosa que termina inesperadamente. Partindo do fato que o mundo virtual não poderia ser dividido em células, tendo em vista os diversos problemas de continuidade de conteúdo, a solução para a geração de relevo deveria basear-se apenas nas informações disponíveis no campo de visão. Embora seja possível gerar relevo utilizando-se como valores de entrada somente as informações do campo de visão, novos problemas de continuidade foram encontrados.

Supondo-se que a geração de relevo utilize a posição do vértice no mundo virtual como semente para uma função pseudo-aleatória de cálculo de relevo; supondo-se também que a geração de montanhas, por exemplo, é definida por pontos chave que indicam a espinha dorsal da cadeia montanhosa. Se o algoritmo se basear apenas nesses pontos chave para calcular as montanhas, o usuário só irá enxergar a montanha quando um desses pontos chave entrar dentro do campo de visão. Isso irá causar falhas abruptas de

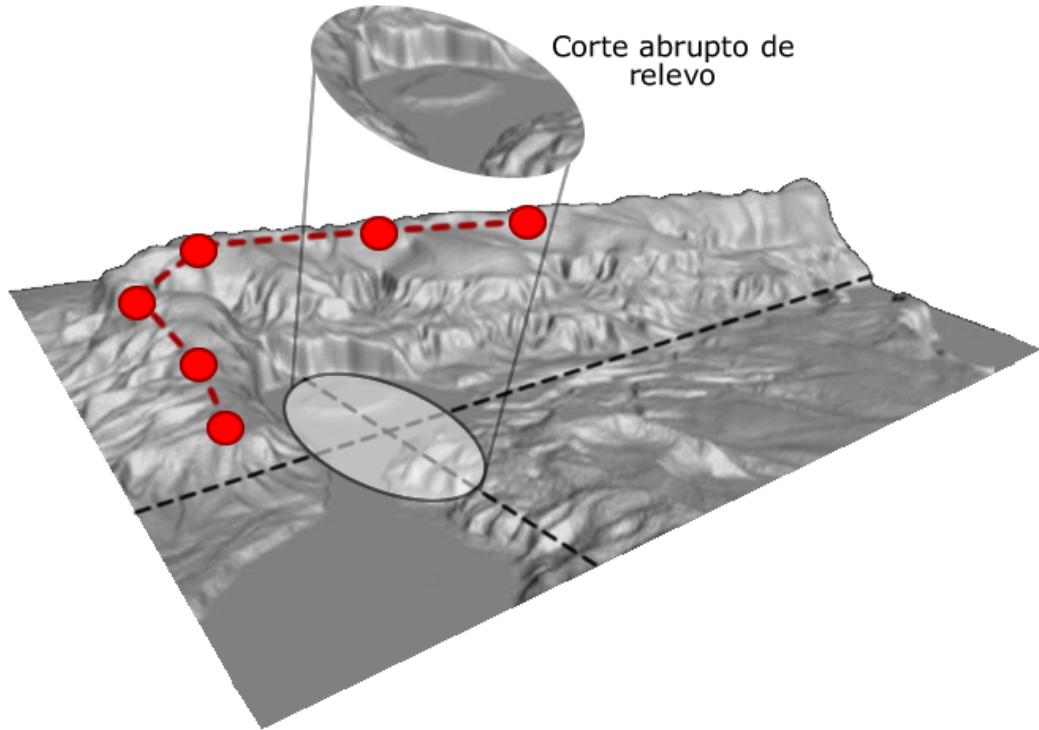


Figura 4.6: Geração de conteúdo com base apenas nas informações do campo de visão

continuidade, porque se o usuário costear a espinha dorsal da montanha, sem que os pontos chavem no campo de visão, ele não verá a cadeia de montanhas, sendo que ele deveria ver um relevo ascendente até o cume dessas montanhas. A figura 4.6 ilustra esse problema.

#### 4.2.2 Solução para geração de relevo sob-demanda

A solução encontrada para a geração de relevo foi utilizar uma função paramétrica que informa a característica de cada um dos vértices do mundo virtual. Nessa abordagem, cada ponto do mundo é utilizado como parâmetro para a função de relevo que, por sua vez, calcula a altura que esse ponto deve ter. Como resultado, tem-se uma função capaz de descrever todo o relevo do mundo virtual, independente do tamanho que ele possua; a quantidade de recursos computacionais que serão utilizados para gerar o conteúdo do campo de visão é diretamente proporcional ao tamanho do campo de visão, não ao tamanho do mundo, visto que a função de relevo utiliza as informações de cada ponto para calcular a sua característica. A figura 4.7 ilustra o funcionamento da função de relevo aplicada ao mundo virtual gerado.

Para aumentar a heterogeneidade do relevo gerado, a ferramenta utiliza duas funções

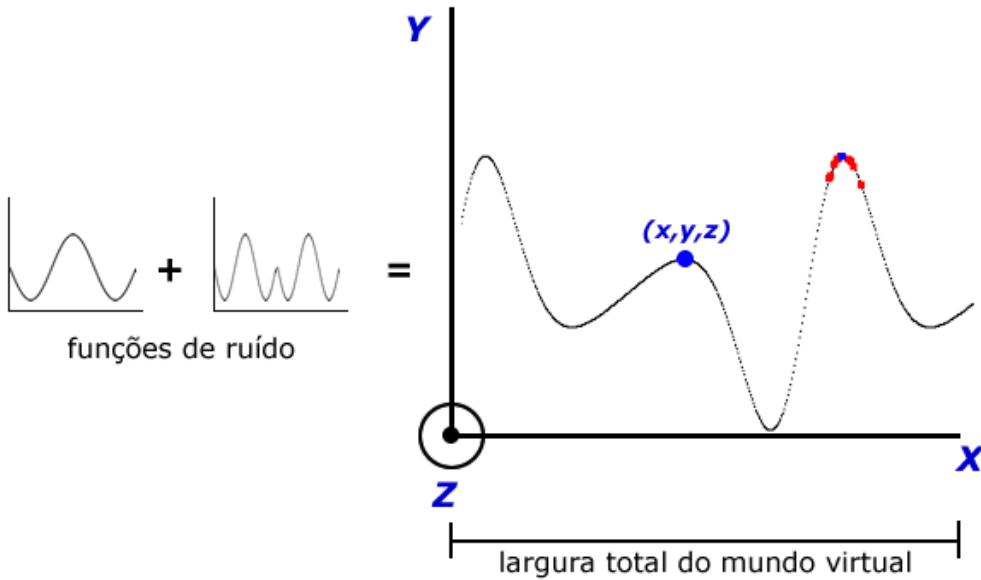


Figura 4.7: Função de geração de relevo parametrizável e o resultado gerado

de relevo, uma para o eixo X e outra para o eixo Z. Além de garantir que o relevo gerado não seja perfeitamente simétrico nos dois eixos, essa abordagem permite um maior controle sobre a geração de conteúdo. A altura de cada ponto do mapa é definido pela soma dos resultados de cada uma dessas funções. Durante a escolha dos métodos para geração de relevo, inicialmente utilizou-se uma combinação de funções seno e cosseno, porém o resultado obtido foi muito simétrico e artificial em comparação com o relevo encontrado na natureza. Para melhorar a qualidade do relevo gerado, utilizou-se então uma mescla dos conceitos apresentados nas seções 2.2 e 2.3. Na abordagem proposta por Greuter et al. [2005], o relevo apresentado é plano e a geração de conteúdo resume-se a prédios e ruas, o que não se enquadra por completo com a abordagem escolhida para o presente trabalho; aproveitou-se, então, a técnica de parametrização da geração de conteúdo através da utilização do posicionamento dos elementos como semente para uma função estocástica de geração. Buscou-se, então, um método de ruído capaz de gerar um relevo que fosse ao mesmo tempo controlável e semelhante àquele existente no mundo real. Conforme apontado por Linda [2007], esse objetivo pode ser atingido através da utilização da função de ruído de Perlin isoladamente ou através da combinação dela com outras técnicas; como pode ser visto nos resultados obtidos pela autora, o relevo gerado possui uma boa qualidade gráfica e custo computacional aceitável. Optou-se, então, por utilizar funções de ruído de Perlin para a geração de ruído.

Para a geração do relevo através de funções de ruído, uma das possíveis abordagens

seria aquela proposta por Häggström [2009], que consiste na sobreposição de um mapa de altura base com mapas de ruído pré-computados. Embora essa abordagem produza um relevo relativamente controlável e aceitável, a sua utilização exigiria algumas mudanças no funcionamento do algorítimo original. Na abordagem do autor, o relevo é gerado de uma só vez e para o mundo virtual inteiro, o que seria proibitivo para a ferramenta proposta no presente trabalho. A solução encontrada foi a ferramenta gerar em tempo real mapas de ruído do tamanho exato da fatia do mundo virtual que o usuário está vendo, que é uma malha regular.

Para controlar o nível de perturbação do relevo, como montanhas mais pontudas ou mais suaves, uma possível solução seria a aplicação de efeitos de erosão como foi proposto por Olsen [2004]. Partindo do fato que os algorítimos de erosão propostos pelo autor utilizam diversas informações referentes à vizinhança dos pontos, se essa abordagem fosse utilizada para a ferramenta, ela resultaria em má formação do relevo nas bordas da fatia do mundo virtual sendo mostrada. Isso aconteceria porque os vizinhos mais afastados dos pontos que estão na borda não estariam dentro do campo de visão do usuário, o que faria com que o algorítimo de erosão calculasse as alterações com apenas uma parte das informações necessárias; como consequência, toda vez que o usuário se movimentasse, alterações abruptas de relevo poderiam acontecer nas bordas da fatia sendo visualizada. Como saída a essa problema, para garantir um controle sobre o nível de perturbação do relevo utilizou-se apenas as propriedades da função de ruído de Perlin: para a obtenção de relevos mais "enrugados", utilizou-se mais oitavas na função de ruído (o funcionamento das oitavas está descrito na seção 2.1.2).

Utilizando a sobreposição de funções de ruído de Perlin para a geração da altura do relevo nos eixos X Z, alterando-se as oitavas de cada função de ruído para atenuar ou aumentar a perturbação dos resultados, foi possível customizar o relevo e obter resultados satisfatórios.

#### **4.2.3 Otimização através de conservação de dados**

Depois que o mapa de altura correspondente à fatia do mundo virtual sendo visualizada é gerado, ele é armazenado na memória e em seguida renderizado na tela. Enquanto o usuário não se mover no mundo, a fatia sendo visualizada não será alterada, logo nenhum cálculo relacionado a relevo deverá ser realizado. Se o usuário se move, porém, a

fatia do mundo que está sendo visualizada é alterada e novas informações de relevo devem ser calculadas. Embora o usuário tenha se movido, apenas uma pequena parcela dos dados que estão no *heightmap* que corresponde à fatia do mundo foram alteradas. Imaginando o *heightmap* como uma matriz, se o usuário se move para a direita, uma nova coluna (com dados novos) deve ser adicionada ao final da matriz e, também, a primeira coluna deve ser removida (porque ela saiu do campo de visão); as demais informações da matriz não precisam ser alteradas.

Tomando vantagem desse fato, a ferramenta utiliza um algoritmo de deslocamento de dados para evitar o reprocessamento de todo o *heightmap* cada vez que o usuário se move. Dependendo da direção para a qual o usuário está se movendo, a matriz tem suas linhas e colunas deslocadas, e apenas uma nova linha ou coluna precisa ser calculada para que a nova fatia do mundo seja mostrada. Levando-se em consideração que os cálculos necessários para a geração do relevo são custosos, já que envolvem funções de ruído de Perlin e diversas outras operações, trocar o processamento total da matriz pelo processamento de apenas uma linha ou coluna é uma vantagem que aumenta o desempenho da aplicação drasticamente. Em testes preliminares, no qual nenhum tipo de deslocamento de dados era realizado e a matriz era completamente reprocessada a cada movimentação, a aplicação apresentou um desempenho médio de 2.5 FPS numa máquina Intel(R) Core(TM)2 Duo 1.66Gz, com 2Gb de RAM e uma placa de vídeo NVidia 8600 GT. Depois que a otimização por conservação de dados foi aplicada, o desempenho saltou para 53 FPS. Mais informações sobre análise de desempenho podem ser encontradas no capítulo 5.

## 4.3 Continentes

A geração de continentes e oceanos foi proposta para quebrar a monotonia de uma paisagem composta apenas por terra no mundo virtual, bem como para aumentar o nível de semelhança com as paisagens encontradas na natureza. As seções seguintes apresentam as técnicas utilizadas e os problemas encontrados na geração de continentes.

### 4.3.1 Oceano como um plano azul

A primeira abordagem utilizada para a geração de água foi uma semelhante àquela proposta por Linda [2007], que consiste na adição de um plano azul cortando todos os

elementos do mundo virtual na altura do nível do mar. Embora essa abordagem seja muito simples, ela reduz consideravelmente a complexidade dos algoritmos de geração de relevo, uma vez que eles não precisam classificar os vértices como sendo "terra" ou "mar" (teste que é realizado analisando-se a altura do vértice em questão). Como consequência disso, não são necessários campos extras na estrutura de dados do vértice para informar que ele é um vértice do oceano, por exemplo. Além disso, não existem problemas na colorização de partes de terra que possuem relevo abrupto e tem contato com o mar, como uma falésia; no caso da marcação de vértices como "terra" ou "mar", a falésia teria uma interpolação entre a cor do mar e a sua própria cor, o que poderia resultar em uma montanha azul até a sua metade, por exemplo.

A utilização de um plano azul para a criação do mar limita a geração de continentes à forma que o relevo possui, ou seja, se houver montanhas largas e com um relevo não pontudo, existirão continentes grandes, caso contrário existirão apenas pequenas ilhas, que podem ser apenas o cume de uma montanha. Todo o relevo que ficou posicionado acima do nível do mar ficou visível ao usuário, porém aquele que ficou abaixo desse nível foi excluído da visualização. A menos que seja interessante que o oceano apresente seu próprio relevo submarino, todas as montanhas e planícies que ficaram debaixo d'água foram perdidas, o que reduz a riqueza de detalhes da cena. Em virtude desse problema, optou-se por utilizar uma abordagem mais controlada para a geração de faixas de terra.

#### **4.3.2 Pré-processamento de faixas de terra**

A solução encontrada para garantir que os continentes sejam mais customizáveis foi pré-processar as faixas de terra existentes no mundo e, então, armazenar essa informação para os demais cálculos da ferramenta. Com essa abordagem, a geração de conteúdo sob-demanda foi parcialmente quebrada, uma vez que os continentes serão gerados por completo antes de todos os demais conteúdos, porém isso garante um melhor controle sobre o que é oceano e o que é terra.

A solução encontrada para esse problema foi a criada por Mogensen [2009], descrita em detalhes na seção 2.3.6. Utilizou-se esse gerador de planetas porque ele possui diversas opções de parametrização, como estipulação da semente que será usada para os cálculos aleatórios, definição da altura/largura do mapa gerado, suavidade das curvas e zoom. Isso aumenta o controle da forma como os continentes serão gerados, que foi a

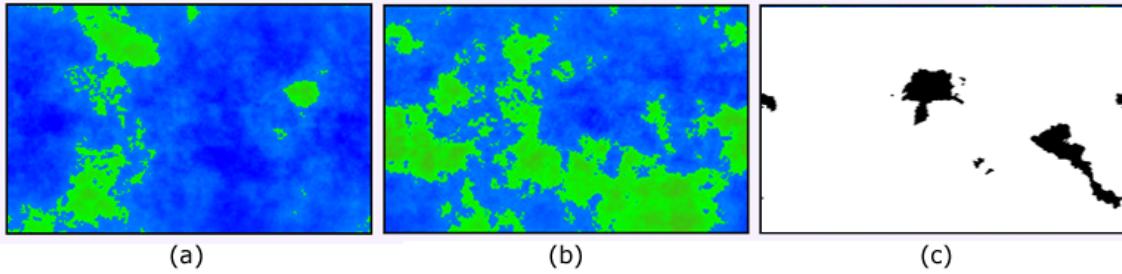


Figura 4.8: Planetas aleatórios gerados com diferentes sementes de entrada. (a) e (b) mapas com informações de relevo [Mogensen, 2009]; (c) mapa mostrando apenas informações sobre terra e mar.

razão principal pela qual optou-se por utilizar uma abordagem diferente de um plano azul para gerar os oceanos e continentes. As figuras 4.8 (a) e (b) ilustram os resultados obtidos com o gerador de planetas de Mogensen [2009]. A figura 4.8 (c) mostra um mapa gerado através da mesma técnica, porém apenas duas cores foram utilizadas para o desenho; pixels representando terra são pintados de preto e pixels representando água são pintados de branco.

A geração do relevo da ferramenta do presente trabalho é feita através da sobreposição dos resultados da função de ruído de Perlin, como está explicado na seção 4.2. Por essa razão, as informações de relevo que são criadas através do gerador de planetas citado são dispensáveis. Em virtude disso, o gerador de planetas foi alterado para gerar um mapa apenas com informações sobre o que é terra e o que é água. A figura 4.8(c) mostra uma mapa gerado contendo apenas informações sobre o que é terra (em preto) e o que é água (em branco).

#### 4.3.3 Visão micro e macro do mundo

Cada pixel da MM é mapeado para diversos pixels no mundo virtual. Uma consequência direta desse mapeamento desproporcional é a geração de grandes faixas de terra retilíneas; se fosse possível a geração de uma matriz com o tamanho exato do mundo virtual, ela conteria a resolução necessária para a ferramenta decidir com precisão se um determinado pixel é ou não terra, em uma proporção de 1:1 (um pixel da MM corresponde a um pixel do mundo virtual). Essa abordagem, porém, não é viável, visto que uma matriz com tais proporções consumiria muitos recursos para ser construída e processada. Embora a ferramenta permita que o tamanho da MM seja ajustado para qualquer valor arbitrário, testes realizados mostraram que um tamanho de 800x800 pixels é o bastante para que

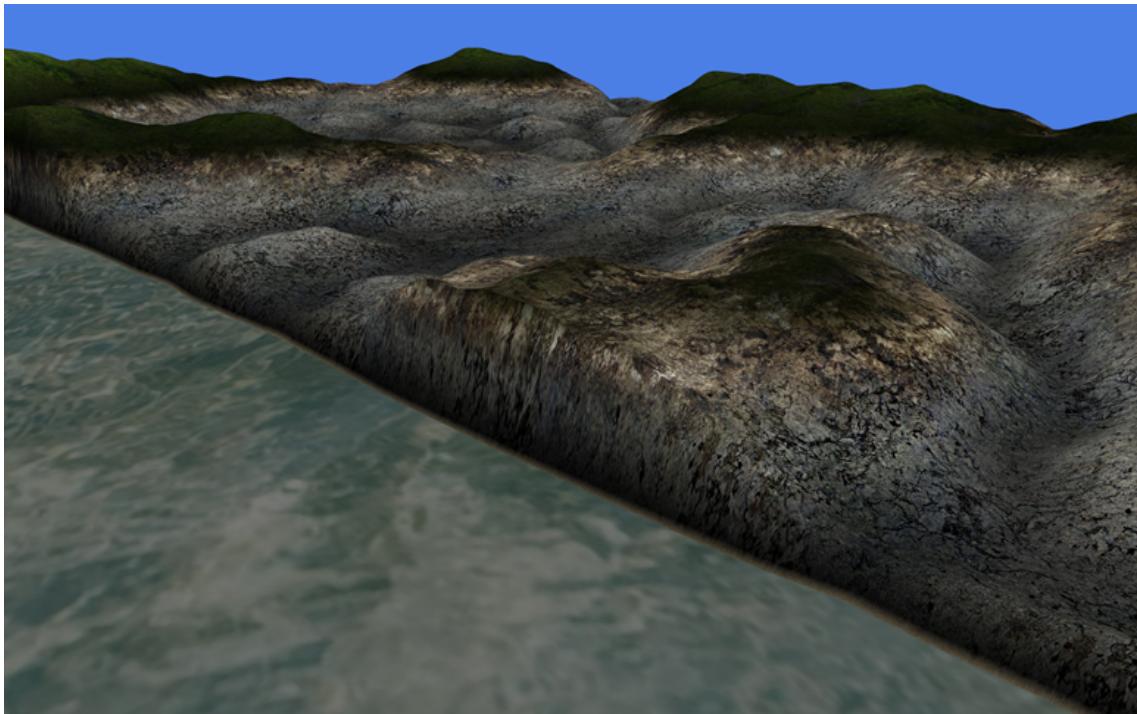


Figura 4.9: Resultado gráfico obtido quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher as discrepâncias de mapeamento da MM

informações suficientes sejam processadas pelos demais algorítimos da ferramenta.

A figura 4.9 ilustra o resultado gráfico obtido pela ferramenta quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher os espaços retilíneos entre um mapeamento e outro.

Nessa ilustração está sendo mostrado um local no mundo virtual que representa a transição entre dois pontos diferentes da MM (um ponto de terra e outro ponto de água). Para explicar o que acontece, são necessárias duas suposições: 1) a ferramenta está desenhando o mundo na posição  $(x, y, z)$ , que é o mapeamento do ponto  $(i, j)$  na MM, e 2) esse mapeamento corresponde a uma faixa de água. À medida que a ferramenta incrementa a coordenada do mundo para poder desenhá-lo, supondo  $(x + 1, y, z)$ , esse ponto também é mapeado para a MM. Se o resultado do mapeamento da coordenada  $(x + 1, y, z)$  ainda for o ponto  $(i, j)$  na MM, então a ferramenta irá novamente desenhar um pixel de água na tela. Supondo que somente no ponto  $(x + 10, y, z)$  o mapeamento mude na MM para  $(i + 1, j)$  (e que esse ponto na MM seja terra), então todos os pontos anteriores a esse serão água e todos os pontos seguintes serão terra (até que o mapeamento na MM mude novamente).

A figura mostra claramente o momento em que o mapeamento na MM muda, que é

quando a ferramenta substitui a renderização da água pela renderização da terra. Como não há algoritmos de geração de conteúdo atuando nessa transição, o usuário pode andar por essa linha reta da costa e não verá qualquer alteração em sua forma ou direção, a menos que outra quebra de mapeamento seja encontrada. Nesse caso, o usuário poderá ver outra costa retilínea perpendicular à aquela que ele está seguindo ou outra costa no mesmo sentido (que é a continuação da costa atual).

#### **4.3.4 Quebra de linearidade da costa**

O mapeamento dos pixels do mundo virtual com as informações da MM em baixa resolução resulta em efeitos gráficos muito irreais. As praias naturais possuem uma curvatura característica e dificilmente terão um comprimento de 20Km em um configuração perfeitamente retilínea, como as praias geradas pela ferramenta. Embora o objetivo do presente trabalho não seja criar paisagens fotorrealistas, praias tão irreais não são aceitáveis. Para contornar esse problema, criou-se uma forma de quebrar a linearidade da costa através da adição de conteúdo aos locais onde o mapeamento da MM é calculado entre dois pontos, um de água e outro de terra. O funcionamento do algoritmo é descrito a seguir.

A MM possui a descrição completa do que é terra e do que é água no mundo virtual. Cada um de seus pixels possui um descritor associado, que informa aos demais algoritmos da ferramenta para qual tipo de terreno um pixel do mundo virtual está sendo mapeado na MM. Inicialmente a ferramenta foi criada com três tipos de terreno: água, terra (continente) e costa (terra em contato com a água). Depois que as faixas de terra são pré-processadas e essas informações são armazenadas na MM, existem apenas informações sobre terra (continente) e água. Esse é o resultado criado pelo algoritmo de criação de mundos citado na seção 4.3.2.

A partir desse momento, o primeiro passo do algoritmo de quebra de linearidade da costa é executado. Utilizando como entrada a MM atual, o algoritmo varre cada um dos seus elementos, atualizando o descritor de informação dos pixels que são costa. Um pixel é dito costa quando pelo menos um de seus vizinhos é água. Depois que o algoritmo termina o seu processamento, a MM contém os três tipos de terreno descritos anteriormente. O próximo passo para a quebra de linearidade da costa é a geração de conteúdo com base no descritor de informação de cada um dos pixels da MM. Quando a ferramenta estiver

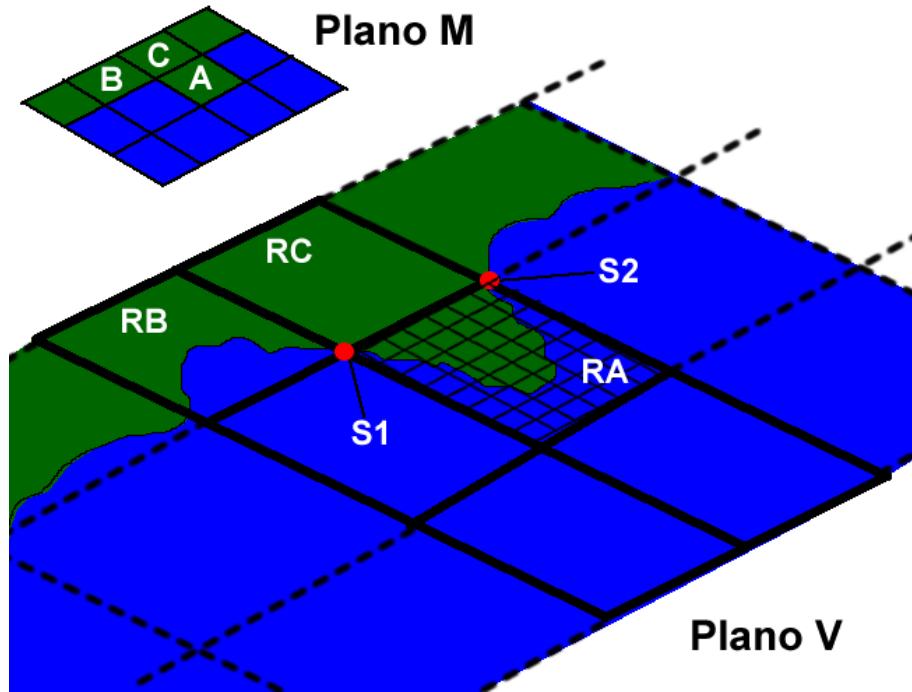


Figura 4.10: Funcionamento do algoritmo de quebra de linearidade da costa

gerando conteúdo para desenhar na tela, o procedimento de geração testa qual o tipo de terreno que está descrito no mapeamento da MM para os pixels que ela está desenhando atualmente. Se o mapeamento terminar em um pixel da MM que é terra, então a função irá gerar um relevo para aquele ponto. Se o mapeamento terminar em um pixel da MM que é água, então a função irá gerar o relevo para o oceano (que é uma altura padrão representando o nível do mar). Se o mapeamento terminar em um pixel da MM que é costa, então a função irá criar alterações nas informações de terra/água nesse mapeamento, o que irá resultar em uma costa não-retilínea e mais realista. A figura 4.10 ilustra o funcionamento do algoritmo.

Os pixels A e B da MM possuem um descritor de informação indicando que eles são costa. O Plano M descreve a MM e o Plano V descreve o resultado do mapeamento dela no mundo virtual, que é uma malha regular. Cada um dos blocos do Plano V é composto por vários pixels, enquanto cada bloco do Plano M corresponde a apenas um pixel da MM. Para fins de clareza, apenas os vértices e arestas da malha do bloco RA estão sendo mostrados, essa informação foi ocultada nos demais blocos. O pixel C da MM é mapeado para um bloco maciço de terra no Plano V, já que o seu descritor informa à ferramenta que ele é um pixel de terra. O pixel A seria mapeado para um bloco maciço de terra também, porém com a atuação do algoritmo de quebra de linearidade ele é mapeado

para uma configuração diferenciada. No momento da geração de conteúdo para os pixels do mundo virtual que estão dentro do bloco RA, o algoritmo de quebra de linearidade distorce as informações de terra/mar para cada um dos pixels, o que faz com que o bloco não seja composto inteiramente de pixels de terra.

Para o processo de distorção das informações terra/mar, foi criado um algoritmo que pudesse gerar costas mais realistas, porém sem tirar do programador o controle sobre quanta terra e quanta água existirá dentro do bloco. Duas abordagens foram planejadas e estudadas para atingir esse resultado, porém apenas uma delas foi utilizada. Na abordagem descartada, a ferramenta identificaria quais são as quinas do bloco RA que se tocam com alguma porção de terra (as quinas seriam os pontos S1 e S2 da figura 4.10). Depois que as quinas fossem identificadas, a ferramenta criaria uma linha entre elas e, então, a distorceria para criar uma linha não uniforme. Em seguida, uma sub MM seria criada, voltada exclusivamente para prover informações sobre terra/mar do bloco RA. A sub MM seria criada com base nas quinas encontradas, na linha criada entre elas e nas informações de água/terra já existentes no bloco (o que evitaria que água fosse inserida para dentro do continente ao invés de uma nova costa ser criada). Um dos principais problemas dessa abordagem foi a complexidade para se criar uma linha de costa aceitável, visto que quanto mais complexa a abordagem, mais processamento era exigido, o que é proibitivo para uma ferramenta de tempo real. A ferramenta teria de desenhar uma linha entre as duas quinas à nível de matriz, ou seja, manipulando os pontos da sub MM criada. Isso poderia ter sido feito através de uma função de ruído ou de números aleatórios que deslocaria cada ponto da linha à medida que eles fossem inseridos, porém as linhas criadas poderiam apresentar falhas entre um pixel e outro, visto que existe um erro inerente no processo de desenho de curvas em um espaço pequeno de pixels. Além disso, existiria o custo computacional extra para que a ferramenta descobrisse onde está a terra e onde está o mar, afim de preencher as faixas de terra e água criadas na sub MM. Depois de descobrir a localização desses elementos, a sub MM teria de ser preenchida com terra e água, o que poderia ser um processo lento para uma MM de grandes proporções. Todo esse processo deveria ser realizado para cada um dos blocos do mundo virtual que fossem mapeados para costa na MM, o que exigiria ainda mais processamento e consumo de memória.

A abordagem que foi adotada para o processo de distorção das informações também é baseada em ruídos e números aleatórios, porém sem a criação de uma sub MM explícita.

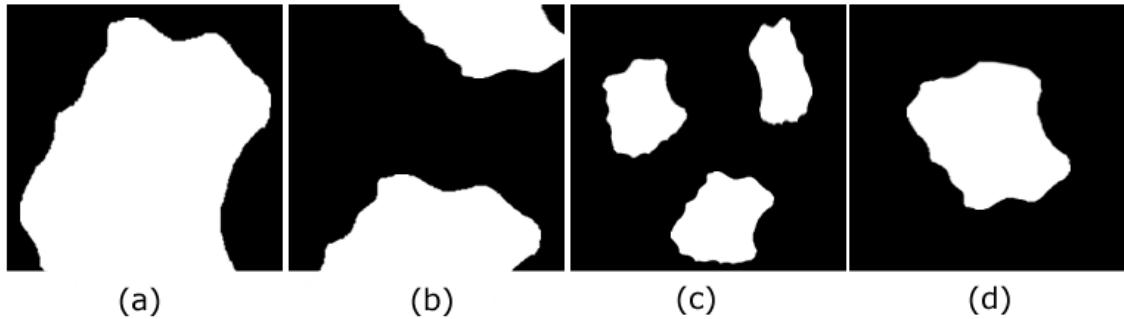


Figura 4.11: Representação gráfica do espectro de ruído utilizado para criação da costa, com diferentes níveis de granularidade. (a) Granularidade média. (b) Granularidade alta. (c) Granularidade muito alta. (d) Granularidade muito baixa.

Na abordagem adotada, a ferramenta utiliza uma função paramétrica para decidir o que é terra/mar dentro de um bloco que é mapeado para costa na MM. Utilizando como base a posição do pixel dentro do bloco RA, a função mapeia essa informação para dentro de um espectro de valores criados a partir de uma função de ruído de Perlin. Em linhas gerais, o que a função faz é testar se o hash do pixel em questão está dentro ou fora do espectro, conforme ilustra a figura 4.11 (a); o processo pode ser imaginado como um teste de altura em um *heightmap* de pequenas proporções (que é criado como resultado do espectro de ruído): se o retorno da função de ruído para o pixel em questão for maior que um determinado valor (que é a granularidade do bloco sendo analisado), então ele é terra, caso contrário ele é água. Na figura 4.11, a área clara mostra todos os pixels cujo hash é maior que o valor estipulado para granularidade do bloco; a área escura da figura, por sua vez, mostra todos os pixels cujo hash é menor que o valor usado como limite. Essa abordagem elimina a complexidade de desenho de linhas em sub matrizes, além de dar maior flexibilidade para a geração de conteúdo: quanto maior a granularidade do bloco, maior será a quantidade de terra no local.

A figura 4.11 (b) ilustra outro *heightmap* fictício gerado pela função de quebra de linearidade, porém com utilização alta granularidade. O bloco do mundo virtual que for mapeado para esse *heightmap* fictício apresentará uma costa diferenciada como resultado da existência de informações na base do *heightmap*, porém a alta granularidade gera uma ilha como efeito colateral (informação existente no topo do *heightmap*). Essa ilha gerada tem sua extensão terminada abruptamente (canto superior direito do bloco), porque a sua continuação está fora do bloco sendo mapeado. Quando essa ilha for renderizada pela ferramenta, o corte abrupto de conteúdo será amenizado pelo diferenciador de praias,

que irá adicionar informações extra ao longo das praias retas afim de torná-las menos uniformes e mais realistas.

A figura 4.11 (c) ilustra o caso no qual uma granularidade muito alta é utilizada. Como consequência, o algoritmo de quebra de linearidade da costa cria diversos trechos de terra espalhados pelo bloco ao invés de um único trecho mais compacto. Os efeitos colaterais da alta granularidade são diversos, como manutenção da costa retilínea já existente no local, criação pequenas ilhas próximas à costa e formação de lagos, esse último sendo causado quando duas ou mais faixas de terra geradas se mesclam entre si ou ao continente de forma parcial.

A figura 4.11 (d) ilustra o caso no qual uma granularidade muito baixa é utilizada. Se a faixa de terrada gerada for muito pequena, o algoritmo pode quebrar a linearidade da costa apenas em um determinado local, que é onde existem informações no *heightmap* fictício gerado; além disso, é possível que a faixa gerada esteja longe do continente, o que irá produzir uma ilha, que terá a suas praias afetadas pelo diferenciador de praias.

#### 4.3.5 Praias

A quebra de linearidade da costa elimina em grande parte o problema de linhas irreais nos continentes, porém o resultado final ainda tende para algo que não é aceitável na natureza. Quando a ferramenta está renderizando uma fatia do mundo, para cada pixels que é descrito como terra um relevo é associado a ele; o mesmo se aplica para os pixels que são descritos como água, porém nesse caso o revelo criado possui sempre a mesma altura (o nível do mar). Como uma consequência direta disso, se a ferramenta estiver desenhando um conjunto de pixels que descreve uma cadeia montanhosa e, logo em seguida, os próximos pixels são descritos como água, a paisagem resultante apresentará um "degrau". Isso acontece porque a cadeia montanhosa foi gerada muito próxima da água, o que faz com que a sua renderização seja abruptamente interrompida no momento que a ferramenta encontra água. Inicialmente a ferramenta não fazia qualquer tratamento para esse caso, o que resultava em costas repletas de falésias, que são um tipo de costa marítima formada por rochas escarpadas e íngrimes. Embora existam falésias no mundo real, elas não estão presentes em todas as costas, somente em algumas, diferentemente do que acontecia no mundo virtual gerado. Para contornar esse problema, criou-se um algoritmo capaz de gerar praias em determinadas áreas, o que torna a paisagem gerada mais realista.

O algoritmo de criação de praias atua pouco antes do conteúdo ser renderizado para a tela. Depois que a ferramenta mapeia para a MM os pixels e depois que o algoritmo de quebra de linearidade da costa atua, o resultado é um *heightmap* pronto para ser renderizado. Antes de ser desenhado na tela, esse *heightmap* é tratado pelo algoritmo de criação de praias, como descrito na figura 4.12. O procedimento varre cada um dos pixels existentes no mapa e, para cada um deles, checa qual a distância que o pixel atual está de um pixel de tipo água à sua volta. A checagem é feita em quadro direções (direita, esquerda, cima e baixo), sendo que a ferramenta avança até encontrar um pixel água ou até que  $N$  pixels sejam consultados. O valor de  $N$  pode ser configurável, porém quanto maior ele for, mais processamento será exigido para a análise da vizinhança e maior será a praia gerada. Em seguida, as quatro distâncias até um pixel de tipo água são somadas e utilizadas em uma fórmula para o cálculo da altura da praia. Os resultados possíveis são os seguintes:

- Se o pixel analisado estiver à uma distância de  $4N$ , isso quer dizer que a ferramenta percorreu às quatro direções possíveis e não encontrou água. Nesse caso, o pixel em questão não tem a sua altura recalculada; esse caso descreve o que acontece com todos os pixels que estão dentro do continente ou na costa porém longe da água: eles não formam uma praia e sua altura é definida pela função de relevo principal;
- Se o pixel analisado estiver à uma distância inferior a  $4N$ , então a sua altura será recalculada. Quanto maior for a distância calculada, maior será a altura do pixel, porém essa variação da altura é calculada dentro de um intervalo definido  $[T, B]$ , onde  $T$  é a altura máxima e  $B$  é a altura mínima de uma praia, respectivamente. O resultado dessa abordagem é uma praia com declive.

A figura 4.13 ilustra o resultado obtido com a geração de praias. Depois que essa abordagem foi colocada em prática, porém, a ferramenta apresentou uma queda brusca de desempenho. A razão dessa degradação são as iterações aplicadas para que água seja encontrada na vizinhança dos pixels. Para cada pixel desenhado na tela, a ferramenta deve fazer quatro iterações, sendo que para alguns pixels o limite máximo de iterações permitidas será atingido. Além disso, a solução apresentava outro problema ligado à geração de conteúdo sob-demanda, que fazia com que as praias sumissem e aparecessem conforme o usuário se movia. A explicação para o problema é que o algoritmo de geração de praia trabalhava apenas com os pixels que estavam dentro da fatia que a ferramenta

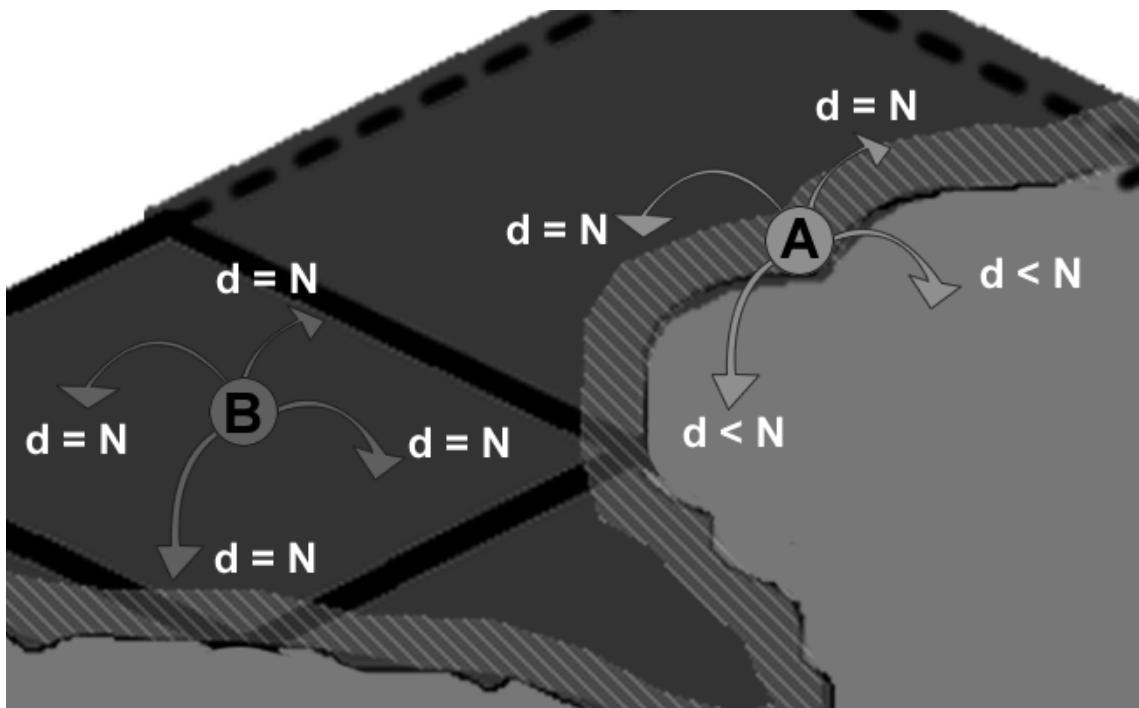


Figura 4.12: Funcionamento do algoritmo de geração de praias



Figura 4.13: Praia gerada ao longo da costa

recortou do mundo virtual; isso quer dizer que se o usuário se move e a água sai do campo de visão, todos os pixels que estão na fatia não tem mais água como vizinha, o que faz com que a praia seja removida.

A solução encontrada para esses dois problemas foi utilizar uma nova abordagem para consulta dos pixels vizinhos. Em vez de iterar sobre cada um dos pixels em cada caminho em busca de água, a ferramenta faz um salto de distância  $N$  a partir do pixel atual em cada um dos caminhos. Em termos práticos, é como se apenas uma iteração fosse realizada para cada uma das direções. Além disso, as informações do pixel consultado não são obtidas da fatia recortada, mas sim do mapeamento direto dele com a MM (ou com o seu respectivo mapeamento, no caso do pixel a ser analisado ser uma costa, caso que exige um mapeamento diferente).

Utilizando a nova abordagem para geração da praia, o desempenho da aplicação teve um decréscimo em virtude do processamento extra que foi inserido, porém essa perda foi aceitável. Como o resultado obtido com a adição de praias superou a leve perda de desempenho, optou-se por manter essa funcionalidade.

#### **4.3.6 Arquipélagos e praias diferenciadas**

Utilizando os algoritmos de quebra de linearidade e criação de praias ao longo da costa, a ferramenta passou a gerar paisagens mais realistas. O resultado final em relação à costa e à praia, porém, apresentou um padrão muito definido, o que é incomum de acontecer no mundo real, no qual as linhas e paisagens naturais tendem a seguir um princípio aleatório ou menos padronizado. Se o usuário viajasse pelo mundo virtual apenas pela costa, ele veria praias com a mesma configuração (mesmo tamanho) e nenhuma ilha ou arquipélago ao longo do oceano. Para melhorar esse aspecto, criaram-se dois novos algoritmos que atuam na costa: um diferenciador de praias e um gerador de arquipélagos.

O **diferenciador de praias** atua perturbando a distância utilizada para o cálculo dos pixels água vizinhos a um determinado pixel. Em vez de utilizar uma distância fixa  $N$  para o cálculo da distância até a água, o diferenciador utiliza a posição do pixel como semente para uma função de ruído, que tem como retorno a nova distância que será utilizada nos cálculos. Utilizando essa técnica, o diferenciador é capaz de alterar o tamanho e forma da praia, o que faz com que determinadas regiões apresentem uma maior quantidade de areia do que outras. A figura 4.14 ilustra os resultados obtidos.

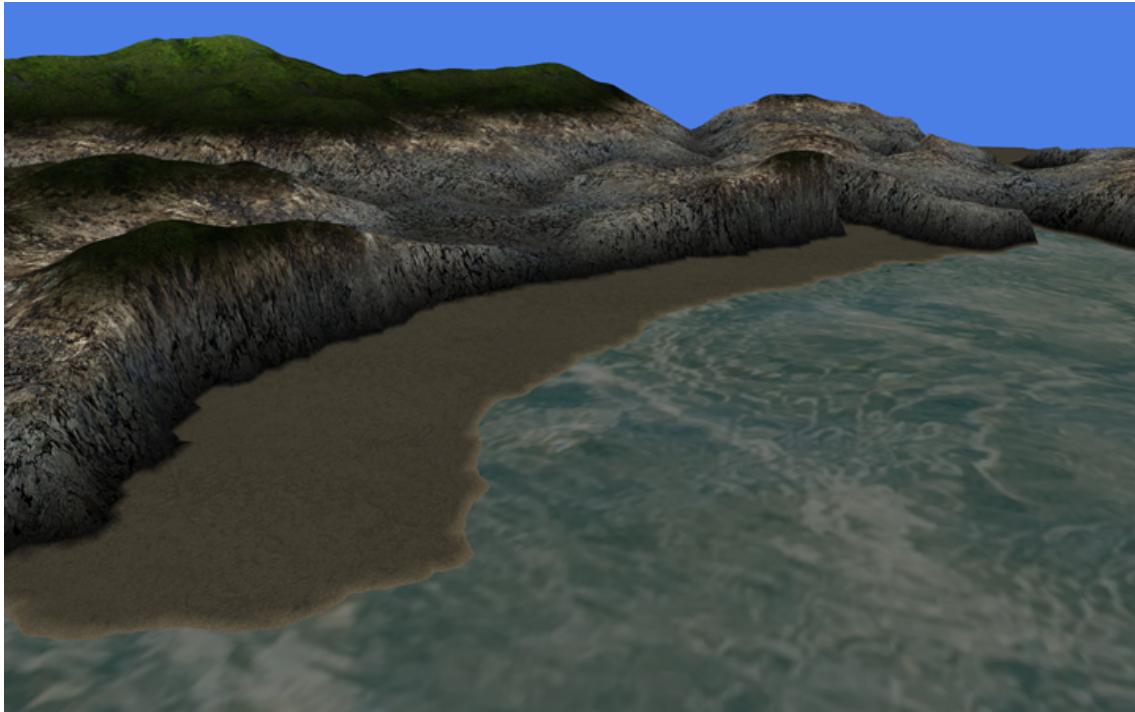


Figura 4.14: Resultado obtido com a aplicação do diferenciador de praias

O gerador de arquipélagos atua criando novas faixas de terra em determinados pixels da MM. Depois que a MM é criada e todos os descritores de informação de cada um de seus pixels é configurado, o gerador de arquipélago itera sobre os pixels que representam a costa e, para alguns deles, adiciona a informação que essa região possui ilhas. No momento em que a ferramenta estiver renderizando os pixels que são mapeados para essa região da MM, o descritor de informação será consultado e a ferramenta saberá que essa região necessita de um conteúdo novo, além do conteúdo utilizado para a quebra de linearidade da costa. Esse conteúdo é criado utilizando a mesma abordagem do algoritmo de quebra de linearidade, porém utilizando como pixels de análise aqueles pixels que são água. Para cada pixel sendo analisado, a sua posição é utilizada como um hash que é testado contra um espectro criado por uma função de ruído. Os ruídos utilizados para esse espectro são diferentes daqueles do algoritmo de quebra de linearidade, visto que o resultado esperado são pequenas porções de terra, não um bloco maciço dela. Para conseguir esse efeito, aumentou-se o número de oitavas da função de ruído de Perlin e aumentou-se o limite que é utilizado nos testes para saber se o pixel é ou não terra. O resultado do gerador de arquipélagos é combinado com o o algoritmo de quebra de linearidade, o que faz com que as ilhas sejam geradas, em determinados casos, muito próximas à costa. O resultado final obtido com o gerador de arquipélagos são ilhas de

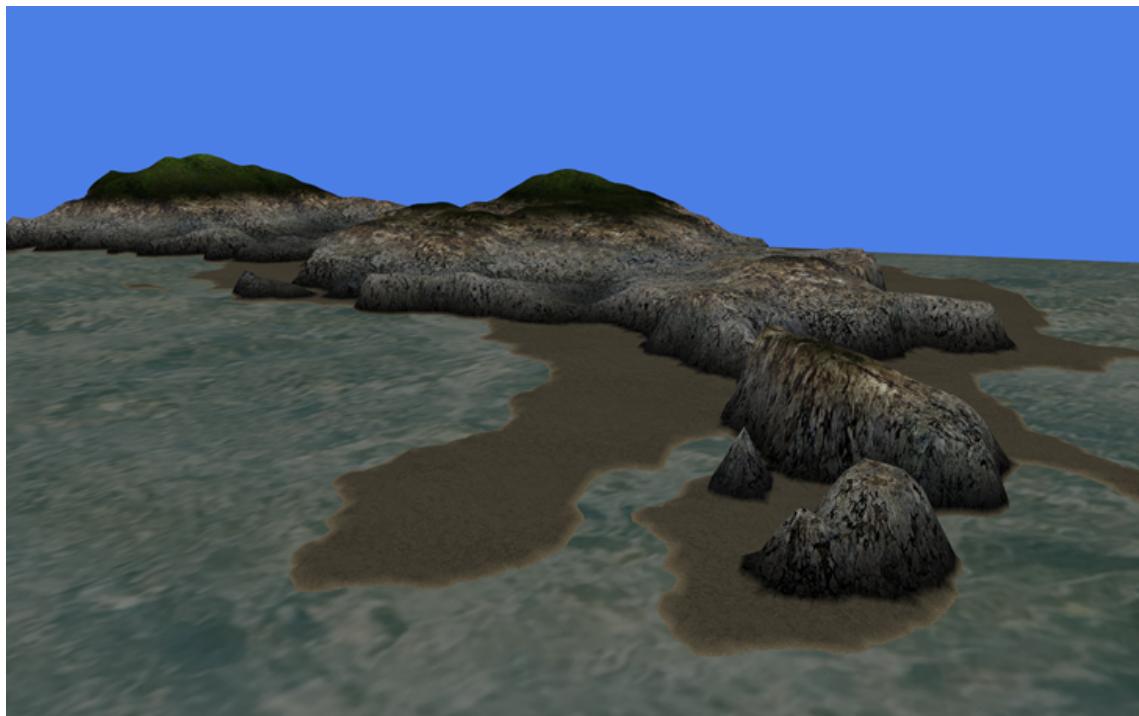


Figura 4.15: Ilha obtida com o gerador de ilhas

diversos tamanhos ao longo da costa em determinados locais do mundo virtual. A figura 4.15 ilustra os resultados obtidos.

## 5 RESULTADOS

Através da utilização de técnicas para geração de conteúdo e relevo, criou-se uma ferramenta capaz de gerar mundos virtuais complexos em tempo real para utilização em jogos 3D. Esse capítulo apresenta uma análise dos resultados obtidos, reforçando-se os pontos positivos e negativos da ferramenta desenvolvida; além disso as principais funcionalidades implementadas são explanadas, com imagens dos resultados e análises de desempenho.

### 5.1 Aspectos gerais da ferramenta

A ferramenta desenvolvida possui como característica principal a capacidade de gerar um mundo virtual pseudo-infinito com continentes, oceanos, áreas planas, montanhas, baías, praias e falésias, com enfoque na geração da borda dos continentes. O resultado final pode ser utilizado para a criação de cenários em jogos 3D.

A aplicação foi projetada para ser utilizada como uma API externa, integrável no código-fonte de um jogo ou *engine*, para adicionar funcionalidades relacionadas à geração de cenários. Toda a ferramenta foi encapsulada dentro de uma única classe que, por sua vez, é composta de diversos outros componentes menores, acessáveis e customizáveis. Para garantir que a ferramenta não fosse intrusiva ao ponto de modificar o fluxo de dados ou de desenho da aplicação na qual ela está integrada, todas as funcionalidades oferecidas podem ser utilizadas separadamente.

Em um exemplo simples, se a ferramenta está sendo utilizada para o desenvolvimento de um jogo de estratégia no qual um pequeno terreno é necessário, o programador não precisa utilizar as funções de desenho da ferramenta para renderizar o terreno gerado; é possível gerar um mundo com as proporções necessárias e, então, acessar somente o mapa de altura criado. Isso fará com que o código do jogo aproveite-se apenas da funcionalidade

de geração de um terreno com variações de relevo, porém não irá alterar a forma como o jogo já havia sendo desenhado na tela, por exemplo. Além disso, é possível que o programador acesse outros dados gerados, como a matriz que define o que é continente e o que é oceano, se o interesse estiver apenas em gerar continentes e mares para o jogo.

## **5.2 Avaliação de técnicas e resultados obtidos**

Essa seção tem por objetivo avaliar cada uma das técnicas utilizadas na geração de conteúdo da ferramenta, mostrando os resultados que foram obtidos com cada abordagem. Leva-se em consideração a flexibilidade proporcionada pela ferramenta durante a utilização da funcionalidade e o resultado gráfico alcançado.

É importante frisar que o objetivo da ferramenta, no escopo do presente trabalho, não é a geração de conteúdos reais ou foto-realísticos, mas sim de elementos que possam ser utilizados para a criação de um cenário num jogo 3D. Entende-se por graficamente aceitável todo o resultado obtido que se enquadre dentro de um jogo e não destoe daquilo esperado pelo jogador, como uma montanha composta por ondulações senoidais suaves ao invés de um conjunto de cristas piramidais.

### **5.2.1 Avaliação dos continentes**

Essa seção tem por objetivo analisar os continentes gerados pela ferramenta, além de analisar a técnica utilizada para que essa funcionalidade fosse implementada. Os continentes gerados não são compostos por partes de terra pré-computados; eles são calculados como resultado de um conjunto de ações aleatórias guiadas por uma semente inicial. Variando-se a semente de entrada, é possível gerar continentes com as mais variadas formas, que vão desde um planeta fortemente tomado por faixas de terra até um cenário com ilhas pequenas e esparsas. Utilizando-se um algorítimo de divisão recursiva de um tetraedro, o módulo de geração de continentes cria um mapa que é uma representação plana de um planeta esférico. Conforme pode ser visto na figura 5.1, analisando-se os continentes em uma linha horizontal, pode-se perceber que as faixas de terra coincidem nas duas extremidades do plano (direita e esquerda), o que produz um fluxo contínuo de terra. Analisando-se o mapa no sentido vertical, nota-se que as duas extremidades do plano não casam, o que é o comportamento esperado para uma projeção plana de uma esfera. Se o plano descrito fosse utilizado para "embrulhar"uma esfera, haveriam distorções nos con-

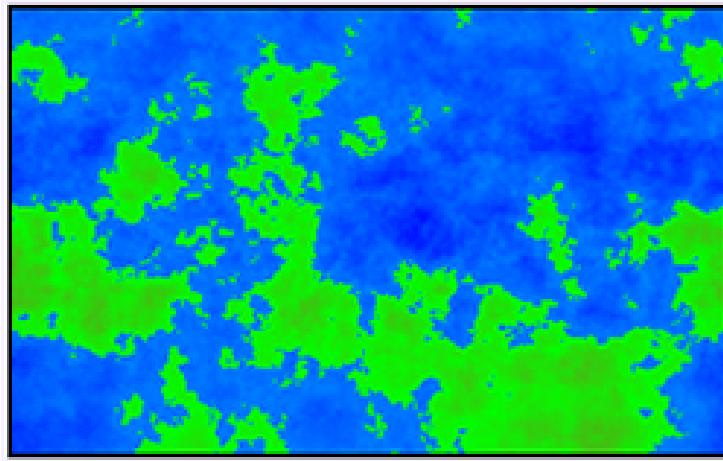


Figura 5.1: Representação 2D dos continentes gerados pela ferramenta

tinentes localizados próximo aos polos, porém o resultado final seria um planeta esférico com uma continuidade em seus elementos.

Essa característica permite que o mundo virtual gerado simule o que existe no mundo real, no qual uma pessoa que caminhe na linha do Equador irá dar voltas ao redor do planeta vendo o seu conteúdo (relevos, oceanos, etc) se repetir ao longo do tempo. Essa funcionalidade de fazer o mundo virtual simular o comportamento esférico do planeta Terra não foi implementada na ferramenta; o usuário pode andar sobre os continentes gerados como se estivesse caminhando sobre um plano e, ao chegar em uma de suas bordas, ele é impedido de prosseguir.

O algorítimo original desenvolvido por Mogensen [2009] para a geração de continentes foi levemente alterado para que pudesse ser integrado à ferramenta. A modificação principal realizada foi a remoção das informações de altura existentes ao longo dos continentes e oceanos, o que poderia ter sido utilizada como base para a geração do relevo. Embora essa fosse uma funcionalidade interessante, ela foi abandonada em favor da geração de relevo completamente parametrizável e independente de elementos pré-computados. O módulo de geração de continentes, porém, baseia-se inteiramente nas alturas calculadas para poder gerar os continentes, o que exigiu que um passo a mais de pós-processamento fosse incluído no algorítimo para transformar o mapa com informações de altura em um mapa somente com valores indicando o que é terra e o que é mar. Como consequência dessa abordagem, a ferramenta gasta um tempo considerável nas divisões recursivas para gerar os continentes e o seu revelo, porém esse último é completamente descartado ao final do processo. Esse fator aumenta o tempo de geração dos continentes e aplica uma



Figura 5.2: Continentes e oceanos gerados pela ferramenta

perda considerável no conceito de tempo-real proposto pela ferramenta, porém, mesmo assim, os benefícios associados aos continentes gerados (como continuidade de conteúdo) compensam.

O tempo para a geração dos continentes é diretamente proporcional ao tamanho da macro-matriz especificada. Em testes realizados, a redução da macro-matriz para valores inferiores a  $800 \times 800$  rendeu aumentos de desempenho consideráveis. Em contra partida, quanto menor o tamanho da macro-matriz, mais quadrados e lineares serão as linhas da costa de cada um dos continentes, o que pode produzir um resultado gráfico não muito aceitável. Para contornar esse problema, é possível ajustar o algorítimo de geração da costa para que ele faça alterações mais agressivas nas bordas dos continentes, o que irá quebrar a linearidade gerada por uma macro-matriz de baixa resolução. O ajuste desses dois elementos abre um leque de possibilidades, visto que pode-se encontrar um ponto de equilíbrio entre tempo de geração dos continentes e linearidade da costa. A figura 5.2 mostra os resultados obtidos em relação a continentes e oceanos gerados pela ferramenta.

### 5.2.2 Avaliação do relevo

O relevo é parte importante do mundo virtual, porém uma abordagem mais aprofundada em relação a esse tópico está fora do escopo do trabalho, visto que o enfoque buscado está na geração de continentes e costas, e não necessariamente no interior dos mesmos.

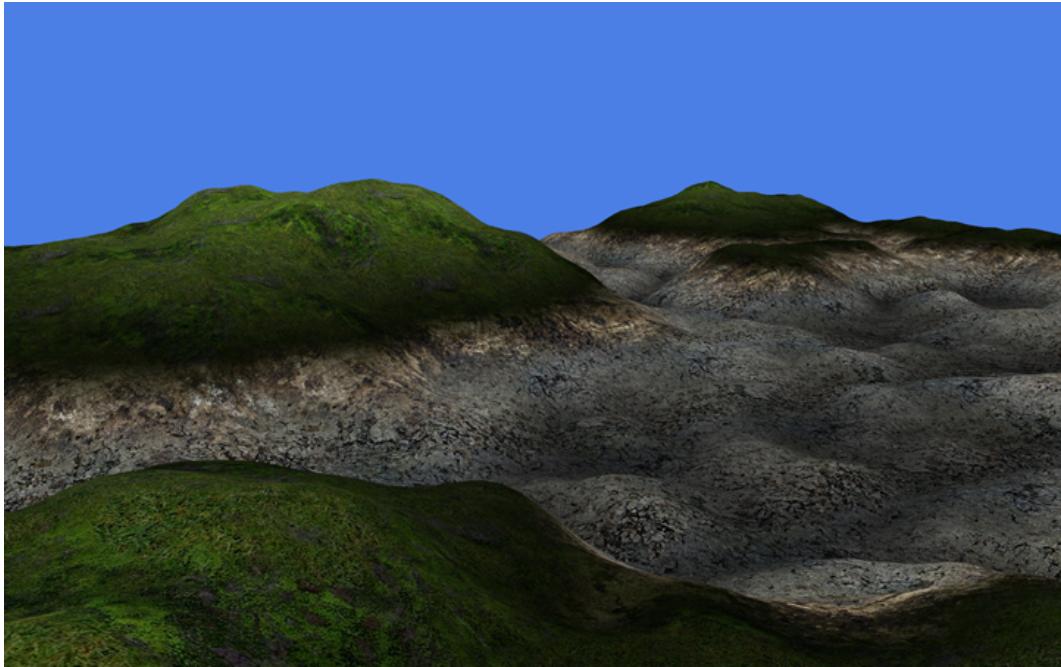


Figura 5.3: Relevo gerado pela ferramenta

(planaltos, planícies, etc). Em relação à geração de relevo na ferramenta, seguindo um protótipo de função pré-definido, o programador pode criar qualquer código para a geração de relevo, o que garante flexibilidade nesse aspecto. A ferramenta possui embutido um gerador de relevo baseado na função de ruído de Perlin, que produz uma paisagem semelhante àquela encontrada no mundo real. A figura 5.3 ilustra o relevo criado pela ferramenta utilizando-se as funções de geração embutidas.

A função de relevo que foi implementada na ferramenta é capaz de gerar paisagem que imitam de forma simples o que é encontrado na natureza, porém a grande maioria dos esforços de desenvolvimento não foram empregadas sobre esse tópico. Como consequência, a ferramenta não possui um grande leque de possibilidades de geração de conteúdos para o interior dos continentes, como cadeias montanhosas, planícies e planaltos. O relevo gerado atualmente é simplista no sentido de não apresentar grandes diversidades geológicas, entretanto se o objetivo da utilização da ferramenta não for criar um terreno rico em diversidade, o objetivo é plenamente atingido.

A geração atual de relevo foi desenvolvida visando-se a obtenção de ondulações de média/baixa frequência nas funções de ruído, o que faz com que não existam cadeias montanhosas pontudas ou depressões abruptas. A baixa frequência nos ruídos da função de geração de relevo faz com que o usuário veja montanhas com um ângulo muito suave de inclinação, além de áreas com poucas ondulações, que podem ser interpretadas como

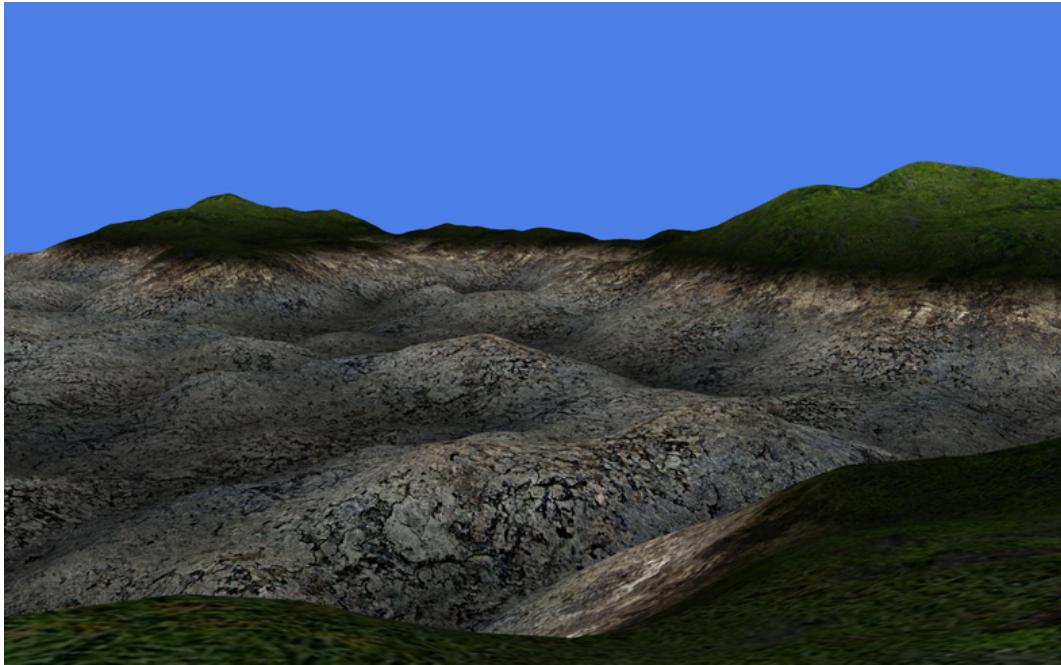


Figura 5.4: Área com relevo de baixa frequência

planaltos. A figura 5.4 ilustra uma área do mundo virtual que possui relevo com baixa frequência.

Se a função de geração de relevo utilizar frequências muito baixas e o seu espectro de valores for estendido à todo o mundo virtual, as montanhas e ondulações que serão produzidas serão muito suaves. Esse fenômeno acontece porque baixas frequências não criam alterações acentuadas de altura nos polígonos do relevo. Na utilização de altas frequências, a grande quantidade de ruído cria montanhas mais pontiagudas, porém as áreas intermediárias entre elas são muito ondulada, que é o reflexo da propagação de um espectro de valores muito ruidoso para o mundo virtual. Uma das formas de contornar esse problema é replicar a extensão do espectro de valores, ou seja, em vez de utilizar um espectro que cubra o mundo inteiro, utilizar esse mesmo espectro três vezes ao longo do mundo. Isso permite que pouco ruído seja utilizado, porém evita que o relevo resultante tenda ao plano. A ferramenta utiliza essa abordagem para criar um relevo com montanhas consideravelmente onduladas, porém sem áreas de ondulação estranha entre elas. A figura 5.5 ilustra os diferentes tipos de relevo obtidos com a variação do tamanho do espectro de valores.

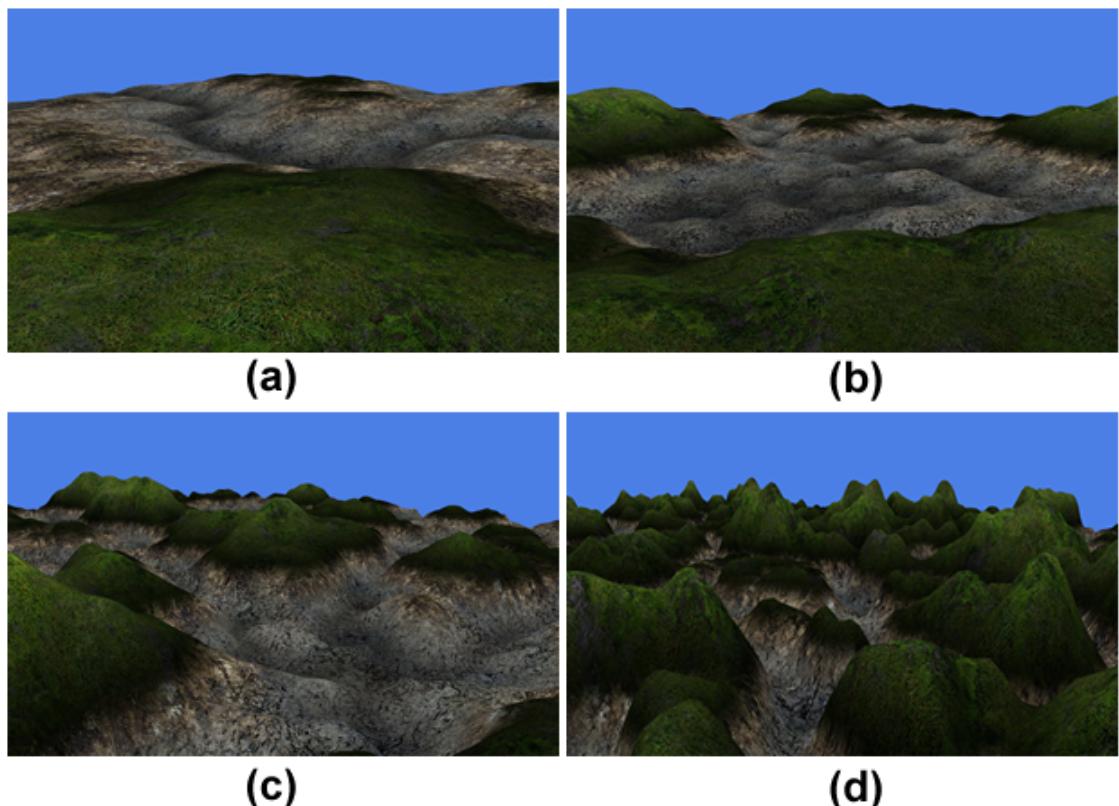


Figura 5.5: Diversos relevos gerados a partir de diferentes espectros de ruídos. (a) Pouco ruído estendido para o mundo inteiro sem replicação. (b) Pouco ruído replicado 200 vezes. (c) Bastante ruído com replicação de 200 vezes. (d) Bastante ruído sem replicação.

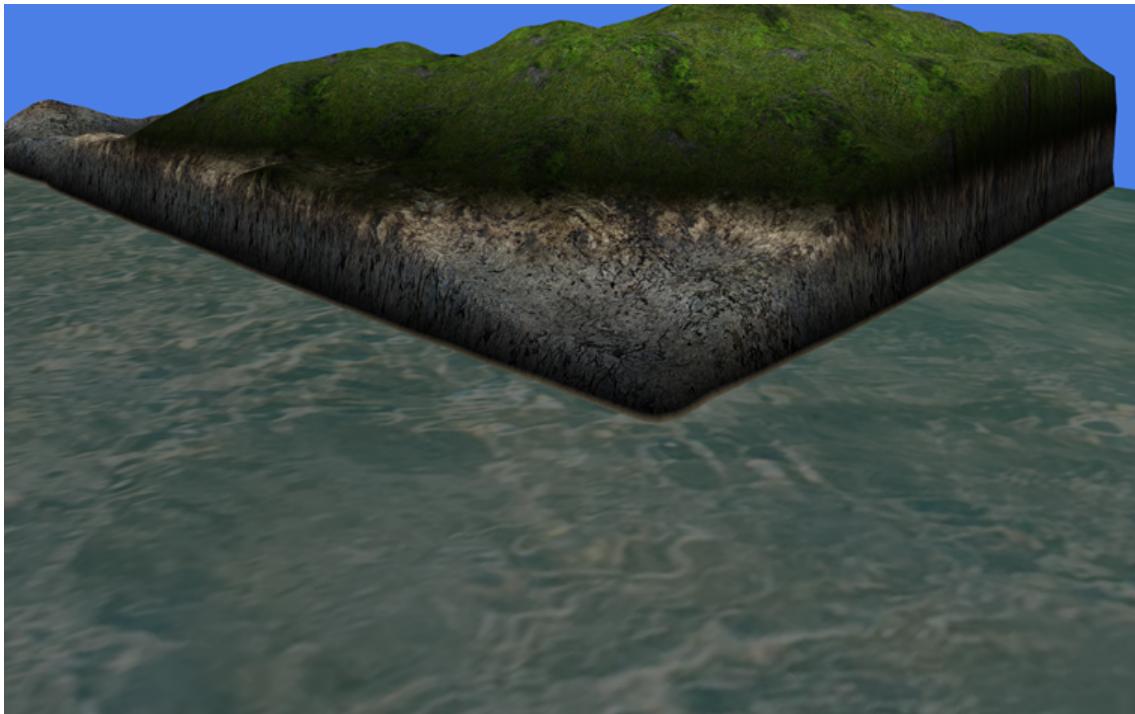


Figura 5.6: Local de encontro de duas linhas da costa sem a aplicação de qualquer algoritmo de geração de conteúdo extra

### 5.2.3 Avaliação da costa

A geração da costa é composta por dois pilares principais, um de aspecto mais global e outro mais local. No aspecto global, a ferramenta utiliza apenas dados encontrados na MM para criar as linhas que compõem a costa, conforme descrito na seção 4.3.3. O resultado final para essa abordagem são costas completamente retilíneas, o que é irreal do ponto de vista do usuário. A figura 5.6 ilustra duas linhas da costa completamente retilíneas.

Na abordagem inicial da ferramenta, não se planejou a adição de conteúdos extra às linhas da costa geradas pelo algoritmo de criação do mundo. A primeira ideia desenvolvida era baseada na utilização integral e exclusiva das informações da MM para a geração das linhas da costa, visto que o algoritmo de geração do mundo é capaz de criar costas suaves e convincentes. As proibições de tamanho da MM, porém, minaram essa abordagem e fizeram com que a geração da costa ficasse limitada a linhas muito retas devido à falta de resolução disponível na MM. Embora seja possível encaixar a utilização de costas tão retas em determinados contextos (como em jogos no qual oceanos não são relevantes), isso limita pelo menos pela metade as opções de utilização da ferramenta. Jogos que precisam apresentar uma interação do jogador com o oceano ficariam impossíveis.



Figura 5.7: Pequena baía com rochas

bilitados de utilizar os recursos criados ou teriam de trabalhar em cima dos resultados da ferramenta para contornar o problema da geração de costas muito retilíneas.

A solução encontrada para contornar esse problema foi a adição de conteúdo utilizando uma abordagem local. Diferentemente da criação global, que se baseia somente nas informações amplas da MM, a geração de conteúdo local se concentra em utilizar as informações da MM como sementes para algoritmos de geração de conteúdo. A aplicação desses algorítimos supre a ausência de informações que existente entre o mapeamento de coordenadas do mundo para a MM resultantes da diferença de resolução entre elas. Como consequência, a ferramenta é capaz de gerar novos conteúdos em níveis de resolução diferentes, permitindo inclusive que o programador defina a granularidade do conteúdo a ser gerado. Conforme explicado na seção 4.3.4, a utilização de uma função baseada em ruído de Perlin para a geração da costa não limita as curvas que podem ser criadas ao longo da borda dos continentes, o que pode resultar nos mais diversos tipos de configurações para a costa. Aliando-se esse comportamento com à geração de praias de tamanho diferenciado, a ferramenta é capaz de adicionar um certo teor de aleatoriedade ao conteúdo criado, o que tende a imitar com mais realidade a natureza vista no dia-a-dia.

As figuras 5.7, 5.8 e 5.11 ilustram a criação de pequenas baías em certos locais da costa. Isso acontece porque nesses locais o algoritmo de quebra de linearidade da costa

criou braços de terra curvos partindo da linha reta do continente e, ao mesmo tempo, o algorítmico de criação de praias reduziu ao máximo a quantidade de areia encontrada no local. A ferramenta também é capaz de criar golfos, que são baías de grandes proporções, porém não é possível prever o local exato que isso irá acontecer, porque tal resultado depende da combinação de valores e coordenadas que variam sensivelmente ao longo do mundo virtual.

As figuras 5.9 e 5.10 ilustram a adição de conteúdo à costa retilínea do continente, resultando numa paisagem mais convincente para o usuário. As duas figuras mostram com bastante detalhe a combinação da função de geração de relevo, o diversificador de praias e o algorítmico de quebra de linearidade da costa na criação de conteúdo; a figura 5.9 apresenta uma rocha imediatamente à esquerda da falésia principal, resultado obtido a partir de um espectro de ruído que gerou dois pontos principais indicando a existência de terra: o maior sendo a falésia e o menor sendo a rocha grande à esquerda. Além disso, o diversificador de praias removeu a areia na parte de baixo da falésia, porém à esquerda ele fez o processo contrário. Em conjunto, o algoritmo de geração de relevo com replicação de valores criou pontas na extremidade esquerda da falésia.

A figura 5.10 apresenta a mesma combinação do algoritmo da figura anterior, porém o resultado obtido variou em função da coordenada do mundo virtual no local. Ao contrário do que aconteceu anteriormente, o diversificador de praias adicionou areia à base da falésia e, à esquerda da imagem, criou uma extensão da praia em forma de "braço". Também à esquerda, o diversificador de praias removeu a areia que fica na base do continente, criando um aspecto menos padronizado de conteúdo. À direita, é possível observar que o algoritmo de geração de relevo criou um declive que termina de forma relativamente suave na praia (relevo de baixa frequência), ao passo que na parte esquerda da imagem o relevo termina de uma forma mais abrupta (relevo de alta frequência).

### **5.3 Análise de desempenho**

Todos os testes de desempenho foram realizados num computador Intel(R) Core(TM)2 Duo 1.66Gz, com 2Gb de RAM e uma placa de vídeo NVidia 8600 GT, utilizando-se o Microsoft Visual C++ 2008 Express Edition como IDE de desenvolvimento e o sistema operacional Windows Vista como ambiente de teste.

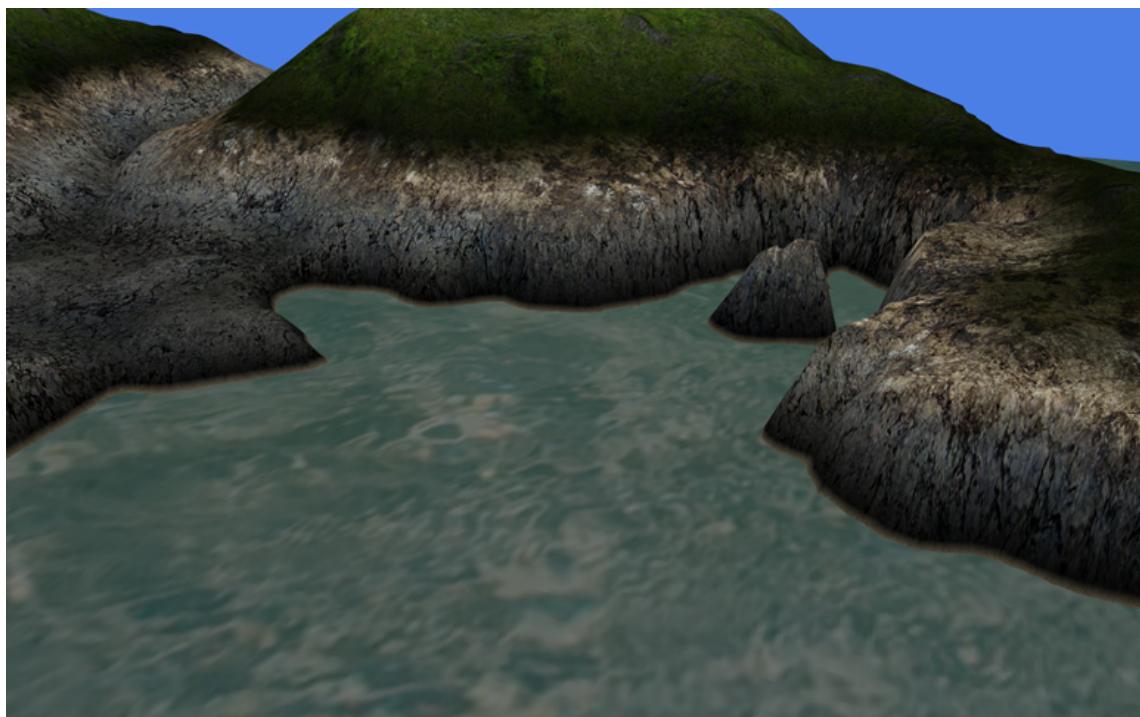


Figura 5.8: Baía de médio porte gerada a partir da criação de um braço de terra originada no continente



Figura 5.9: Extremidade de um continente

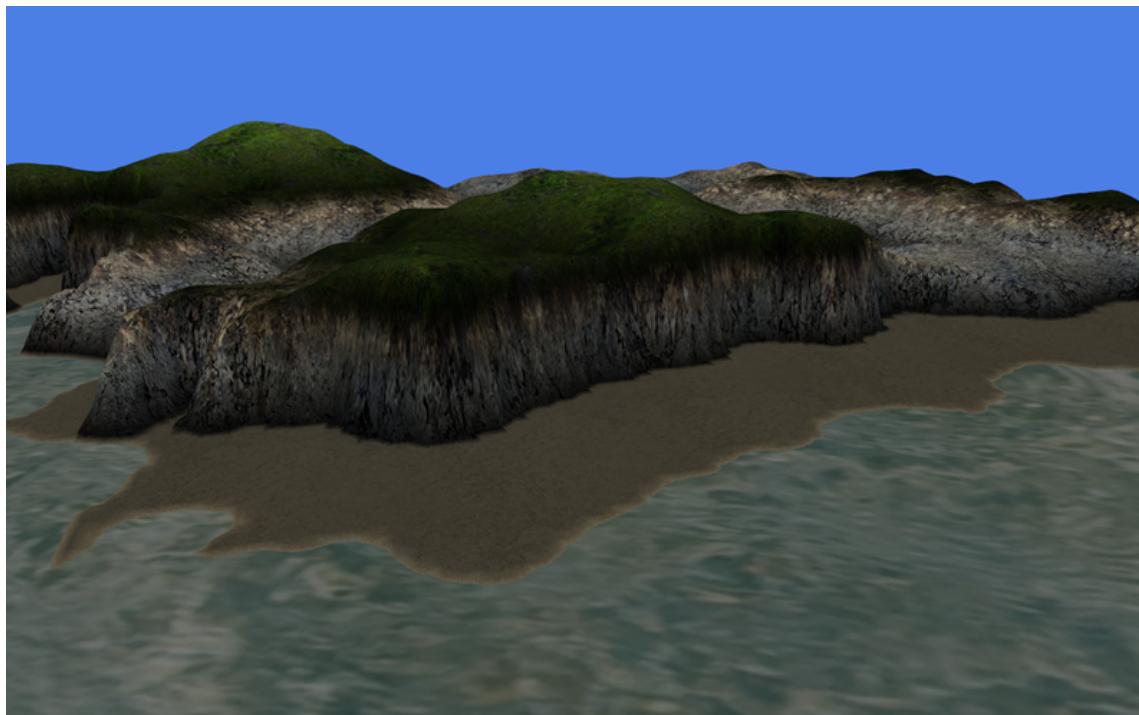


Figura 5.10: Costa de um continente com praias de tamanho variável



Figura 5.11: Baía criada a partir da geração de dois braços de terra originados no continente

## 6 CONCLUSÃO E TRABALHOS FUTUROS

A criação automatizada de mundos virtuais procedimentais é uma das formas existentes para auxiliar desenvolvedores de jogos a criarem ambientes mais ricos em detalhes, em menos tempo e com menos recursos humanos. Ao contrário da abordagem puramente não automatizada, no qual um *game designer* ou artista deve desenhar e modelar o mundo virtual por completo, na abordagem automatizada a adição de detalhes e modelagem de locais fica a cargo da ferramenta geradora. Dependendo do grau de realismo e da complexidade dos algoritmos de geração de conteúdo, uma ferramenta automatizada pode ser capaz de gerar um mundo virtual aceitável para diversos escopos de projeto.

Existem diversas pesquisas em relação à essa área, com abordagens diferentes e focadas em resultados diferenciados. Pode-se destacar nelas a preocupação em criar um mundo virtual de proporções infinitas sem que isso comprometa o desempenho da aplicação. Para atingir esses resultados, diversas técnicas são utilizadas, como aplicação de LOD [Ulrich, 2004, Pozzer et al., 2004] e *quadtree* para gerenciamento de malhas, além de diversos algorítimos procedimentais para a geração de conteúdo, como criação de relevo a partir de fractais, multi-fractais e/ou combinação de funções de ruído.

Este trabalho apresentou o desenvolvimento de uma ferramenta capaz de gerar mundos virtuais pseudo-infinitos com diversificação de formas e relevos ao longo de sua extensão. Utilizando uma combinação de algorítimos e métodos de gerenciamento de conteúdo, a ferramenta é capaz de criar praias, ilhas/arquipélagos, baías e costas que imitam as paisagens encontradas na natureza. Além disso, a possibilidade de parametrizar cada um desses elementos dá ao desenvolvedor um controle maior sobre o resultado que será obtido.

Dentre as inovações apresentadas, estão a criação de um terreno virtual de vastas proporções com enfoque mais detalhado no que diz respeito à geração da costa. Fruto de um

aprofundamento nas pesquisas já realizadas na área, o desenvolvimento da ferramenta foi focado em tratar de forma diferenciada a criação de conteúdo para os diversos elementos existentes (como continentes, relevo, etc). Uma das prioridades do trabalho foi a criação de bordas dos continentes, não a criação de conteúdo para o seu interior. A ferramenta é capaz de gerar continentes contendo falésias, praias e rochedos em suas extremidades. Em testes realizados, o tempo que um jogador levaria para chegar de uma ponta do mundo virtual até a outra num ambiente com o tamanho máximo suportado por um inteiro sem sinal, andando 100 pixel por segundo, seria de um 1 ano e 3 meses.

Partindo do que foi desenvolvido nesse trabalho, abaixo encontra-se uma lista com as possíveis continuações na área de pesquisa:

- Melhoramento do algoritmo de quebra de linearidade da costa. Em certos locais, o espectro criado pela função de ruído não é grande ou granular o suficiente para criar uma costa não retilínea, o que faz a ferramenta produzir praias retas em determinados locais.
- Criação de outros tipos de conteúdo para a MM além dos já existentes (água, terra e costa). Pode-se criar tipos como cordilheiras, desertos, vulcões e florestas, que informariam à ferramenta para acentuar ou atenuar a frequência do ruído nessas áreas. A adição desses novos conteúdos pode seguir a mesma abordagem do gerador de costa, que é um passo intermediário entre o gerenciador de fatias e a MM. A diferença estaria no fato de que o novo gerador produziria conteúdo para ser aplicado em terra firme (como é o caso da criação de desertos). Para criar um deserto, por exemplo, basta que o novo gerador de conteúdo atenue de forma agressiva o relevo gerado para um determinado local; isso removeria o aspecto de relevo montanhoso e transformaria o local em um emaranhado de dunas de areia. A criação de cordilheiras segue o processo inverso, ou seja, o gerador de conteúdo aumentaria e/ou adicionaria um pouco mais de ruído aos valores de relevo produzidos para uma determinada área, o que resultaria em picos montanhosos. Uma evolução dessa ideia seria alterar a estrutura descrita na figura 3.1 para a estrutura descrita na figura 6.1, na qual o gerador de costas foi substituído por um *gerador de conteúdo*. Esse novo gerador teria a capacidade de analisar informações da MM e, então, gerar conteúdo utilizando um conjunto de classes internas (sendo uma delas o atual gerador de costas).

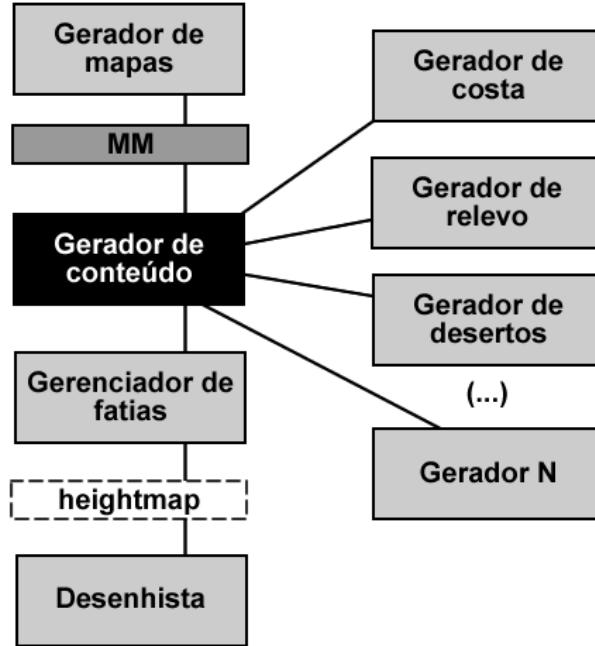


Figura 6.1: Alteração na estrutura atual para permitir uma maior flexibilidade na geração de conteúdos para diferentes tipos de terreno.

- Criação de um algoritmo de adição de conteúdo semelhantes àquele utilizado para criar arquipélagos, porém aplicado à porção de terra que está dentro do continente afim de criar áreas que descrevam cidades ou vilas. Em linhas gerais, a ideia é criar "ilhas" dentro dos continentes que teriam seu conteúdo preenchido por um algoritmo que gera cidades.
- Criação de rios; para alcançar esse objetivo, uma sugestão é a MM receber um novo tipo de terreno (rio) e o algoritmo de geração de relevo estar ciente que tais elementos da MM devem ser desenhado de um forma diferenciada. O grande desafio nessa abordagem é garantir a continuidade dos rios ao longo do mundo virtual tento em vista os problemas de resolução no mapeamento entre a MM e o mundo virtual.
- Portar os algoritmos mais pesados da aplicação para CUDA [nVidia, 2009], que é uma arquitetura de computação paralela de propósito genérico realizado diretamente na GPU da placa de vídeo. Utilizando CUDA, pode-se efetuar os cálculos matemáticos pesados diretamente na placa de vídeo, livrando a CPU dessa tarefa. Em testes realizados, constatou-se que o gargalo de processamento encontra-se na geração da fatia do mundo virtual visível pelo usuário, não na renderização dessa fatia. Embora o aumento da fatia aumente também a quantidade de vértices que

serão desenhados na tela, o impacto maior ocorre nos algoritmos de geração de conteúdo, como criação da costa e do relevo. Quanto maior for a fatia, mais cálculos terão de ser realizados (funções de ruído, interpolações, iterações, etc) para que a nova porção do mundo seja mostrada na tela. A geração do relevo de cada pixel é um processo auto-contido, ou seja, um pixel não interfere no relevo do outro. Dessa forma pode-se utilizar o grande poder de processamento da GPU em relação a tarefas paralelizáveis para o cálculo do conteúdo de cada pixel. Tendo em vista a natureza paralelizável da geração de relevo, espera-se um aumento de desempenho linear.

- Portar a geração do mundo virtual para uma plataforma de 64 bits, o que permitiria que o mundo gerado tivesse o tamanho máximo de  $2^{64}$  pixels de comprimento. Se esse objetivo for atingido, um jogador, andando a 100 pixels por segundo, levaria pouco mais de 5 bilhões de anos para percorrer o mundo virtual inteiro.
- Melhoramento no algoritmo de geração de praias para torná-lo apto a gerar a altura dos pixels da praia baseados em sua própria função de conteúdo, não com base na atenuação da altura de pixels vizinhos. Isso produziria praias com relevo diferenciado, o que tornaria as paisagens ainda mais convincentes para o usuário.
- Criar um gerenciador de parâmetros global para controlar a customização dos elementos da ferramenta. Atualmente a customização de cada elemento está distribuída ao longo das classes. A criação de um controle central reduziria o número de parâmetros e atributos de diversas classes, o que tornaria o código mais limpo e legível, além de tornar a parametrização da ferramenta mais fácil de ser implementada.

## REFERÊNCIAS

MERCATOR ARRUMAR. Arrumar, 2007. ARRUMAR.

Michael F. Barnsley. Fractals everywhere, 1993.

Blizzard Entertainment. World of warcraft, 2007. Disponível em:  
[<http://www.blizzard.com>](http://www.blizzard.com).

Neils L. Clark. Addiction and the structural characteristics of massively multiplayer online games. Master's thesis, University of Hawai, Hawai, 2006.

M. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *Siggraph 03 Conference proceedings*, 2003.

Steven C. Dollins. *Modeling for the Plausible Emulation of Large Worlds*. PhD thesis, Brown University, Estados Unidos da América, 2002.

Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, and Robert J. Moore. Alone together exploring the social dynamics of massively multiplayer online games. In *CHI 2006 Proceedings*, 2006.

Hugo Elias. Perlin noise, 2009. Disponível em:  
[<http://freespace.virgin.net/hugo.elias/models/mperlin.htm>](http://freespace.virgin.net/hugo.elias/models/mperlin.htm).

Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Realtime procedural generation of 'pseudo infinite' cities, 2005.

Mark D. Griffiths, Mark N.O. Davies, and Darren Chappell. Breaking the stereotype the case of online gaming, 2003.

Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proceedings of Graphics Interface 94*, 1994.

Hans Häggström. Real-time generation and rendering of realistic landscapes. Master's thesis, University of Helsinki, Finlândia, 2006.

Hans Häggström. Skycastle - free multiplayer game engine focusing on player creativity and world simulation, 2009. Disponível em: <<http://www.skycastle.org/>>.

JasonWeber and Joseph Penn. Creation and rendering of realistic trees, 1995.

George Kelly and Hugh McCabe. A survey of procedural techniques for city generation, 2004.

Sylvain Lefebvre and Fabrice Neyret. Pattern based procedural textures. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, 2003.

J. P. Lewis. Generalized stochastic subdivision. In *ACM Transactions on Graphics*, 1987.

Ondrej Linda. Generation of planetary models by means of fractal algorithms. Technical report, Czech Technical University, 2007.

Bernd Lintemann and Oliver Deussen. A modelling method and user interface for creating plants, 1998. Computer Graphics Forum.

Torben Æ. Mogensen. Instant planet generator, 2009. Disponível em: <<http://www.eldritch.org/erskin/roleplaying/planet.php>>.

F. Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. The synthesis and rendering of eroded fractal terrains, 1989. Computer Graphics, Volume 23, Number 3, July 1989, pages 41 to 50.

nVidia. Cuda, 2009. Disponível em: <<http://www.nvidia.com/cuda>>.

Jacob Olsen. Realtime synthesis of eroded fractal terrain for use in computer games, 2004.

OpenGL. Opengl shading language, 2007. Disponível em: <<http://www.opengl.org/documentation/glsl/>>.

Ken Perlin. An image synthesizer. In *SIGGRAPH*, pages 287–296, 1985.

Cesar Tadeu Pozzer, Marcelo Dreux, and Bruno Feijó. A real-time single-pass continuous lod algorithm for regular height maps, 2004.

Prusinkiewicz Przemyslaw and Aristid Lindenmayer. The algorithmic beauty of plants, 1990. Springer-Verlag.

Maurice Ribble. Using various techniques to generate height maps used in terrain generation, 2001.

Sony Entertainment. Everquest, 2007. Disponível em:  
<http://everquest2.station.sony.com/>.

Thatcher Ulrich. Chunked lod: Rendering massive terrains using chunked level of detail control, 2004. Course at SIGGRAPH 02, <http://www.tulrich.com/geekstuff/chunklod.html> (01-07-2004).

Thomas Wang. Integer hash function, 2000. Available at:  
<http://www.concentric.net/~Ttwang/tech/inthash.htm>.