

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

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

DISSERTAÇÃO DE MESTRADO

Fernando Bevilacqua

Santa Maria, RS, Brasil

2009

**FERRAMENTA PARA GERAÇÃO EM TEMPO REAL
DE MUNDOS 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)

Co-orientador: Prof^a Dr Cesar Tadeu Pozzer (UFSM)

**Dissertação de Mestrado Nº 2
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
MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D**

elaborada por
Fernando Bevilacqua

como requisito parcial para obtenção do grau de
Mestre em Informática

COMISSÃO EXAMINADORA:

Prof^a Dr Cesar Tadeu Pozzer (UFSM)
(Presidente/Co-orientador)

Prof. Dr^a Andrea Schwertner Charão (UFSM)

Prof^a Dr Marcos Cordeiro d'Ornellas (UFSM)

Santa Maria, 22 de Abril de 2009.

“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 MUNDOS VIRTUAIS PSEUDO-INFINITOS PARA JOGOS 3D

Autor: Fernando Bevilacqua
Orientador: Prof. Dr Cesar Tadeu Pozzer (UFSM)
Co-orientador: Prof^a Dr Cesar Tadeu Pozzer (UFSM)
Local e data da defesa: Santa Maria, 22 de Abril de 2009.

Resumo em português [aqui](#).

Palavras-chave: MMO, mundos virtuais, geração de terreno, jogos 3D.

ABSTRACT

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

ENGLISH TITLE HERE

Author: Fernando Bevilacqua

Advisor: Prof. Dr Cesar Tadeu Pozzer (UFSM)

Coadvisor: Prof^a Dr Cesar Tadeu Pozzer (UFSM)

Here we are supposed to write an English version of our portuguese description.

Keywords: key word, word, key, english stuff.

LISTA DE FIGURAS

Figura 1.1 – Ilustração do mundo virtual de World of Warcraft [2]	12
Figura 1.2 – Ilustração do mundo virtual gerado.....	14
Figura 2.1 – Quatro primeiras iterações do fractal de floco-de-neve de Koch [5] ...	17
Figura 2.2 – À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [7]	17
Figura 2.3 – À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [7]	18
Figura 2.4 – À esquerda, <i>heightmap</i> gerado a partir de divisões estocásticas; à direita, o mesmo <i>heightmap</i> renderizado em 3D [5]	20
Figura 2.5 – Geração de relevo pela disposição do ponto médio (perspectiva 2D) [5]	21
Figura 2.6 – À direita, <i>heightmap</i> gerado a partir de ruído de Perlin. À esquerda, <i>heightmap</i> gerado a partir da combinação de camadas de ruído de Perlin (turbulência). [11]	22
Figura 2.7 – 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 [11]	22
Figura 2.8 – 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 [8]	23
Figura 2.9 – Geração de construções: cada andar é gerado pela mescla de polígonos escolhidos e centralizados aleatoriamente [8]	24
Figura 2.10 – Campo de visão do usuário: apenas as células visíveis tem o seu conteúdo gerado [8]	24
Figura 2.11 – Exemplo de cidade virtual gerada [8]	25
Figura 2.12 – Terreno texturizado com duas camadas: bloco escuro e um conjunto de texturas em formato de borda com transparência [11]	26
Figura 2.13 – Diferentes relevos gerados pela utilização de fractais e suas combinações. [17]	28
Figura 2.14 – Planetas gerados com os diversos algorítimos. (a) Falhas aleatórias. (b) Disposição do ponto médio. (c) Disposição do ponto médio multifractal. (d) Ruído de Perlin. (e) Ruído de Persil Multifractal. (f) Ruído de Perlin <i>Ridged</i> . (g) Ruído de Perlin <i>Ridged</i> Multifractal. [17]	30
Figura 2.15 –(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). [21]	31
Figura 2.16 –À direita, renderização de um mapa de altura sem a aplicação de erosão; à esquerda, o mesmo mapa com a aplicação de erosão. [21] ..	31

Figura 2.17 – <i>Quadtree</i> utilizada para a geração do terreno [5]	32
Figura 2.18 – Subdivisão de células e níveis de detalhe representados por uma árvore binária em 2D. [5]	34
Figura 3.1 – Problema da geração de conteúdo sob demanda.	36
Figura 3.2 – Organização do sistema de coordenadas do mundo virtual	37
Figura 3.3 – Campo de visão do usuário: a área visualizada corresponde a uma fatia do mundo virtual existente.	38
Figura 3.4 – Mapeamento das coordenadas do mundo virtual para as coordenadas de desenho da tela.....	40
Figura 3.5 – Exemplo de LOD aplicado a um terreno virtual	41
Figura 3.6 – Terreno renderizado com baixa resolução (a) e alta resolução (b), ambos com aplicação de LOD [22]	43
Figura 3.7 – Geração de conteúdo com base apenas nas informações do campo de visão	44
Figura 3.8 – Função de geração de relevo parametrizável e o resultado gerado	45
Figura 3.9 – Planetas aleatórios gerados com diferentes sementes de entrada. (a) e (b) mapas com informações de relevo [19]; (c) mapa com informações somente sobre o que é terra/mar.	50
Figura 3.10 –Mapeamento da macro-matriz para o mundo virtual. Como consequência da baixa resolução da matriz, a costa dos continentes é em linha reta.	51
Figura 3.11 –Resultado gráfico obtido quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher as discrepâncias de mapeamento da MM.....	52
Figura 3.12 –Funcionamento do algoritmo de quebra de linearidade da costa	53
Figura 3.13 –Representação gráfica do espectro de ruído utilizado para criação da costa	55
Figura 3.14 –Praia gerada ao longo da costa	57
Figura 3.15 –Resultado obtido com a aplicação do diferenciador de praias. (a) praia original; (b) praia que sofreu efeito do diferenciador	59
Figura 3.16 –Ilha obtida com o gerador de ilhas.....	60
Figura 4.1 – Continente gerado pela ferramenta	63
Figura 4.2 – Continentes e oceanos gerados pela ferramenta	65
Figura 4.3 – Relevo gerado pela ferramenta (<i>wireframe</i>)	66

SUMÁRIO

1 INTRODUÇÃO	11
1.1 Contexto e Motivação	11
1.2 Objetivos e Contribuição	13
1.3 Organização do Texto	14
2 REVISÃO DA LITERATURA	16
2.1 Funções de ruído e fractais	16
2.1.1 Fractais	16
2.1.2 Ruído de Perlin	17
2.2 Geração de relevo	19
2.2.1 Divisões estocásticas	19
2.2.2 Falhas geológicas	20
2.2.3 Deposição de sedimentos	21
2.2.4 Disposição do ponto médio	21
2.2.5 Ruído de Perlin	21
2.3 Mundos pseudo-infinitos	23
2.3.1 Cidade virtual	23
2.3.2 Geração e texturização por camadas	24
2.3.3 Planetas procedurais	27
2.3.4 Fractais afetados por erosão	29
2.3.5 Divisões estocásticas de uma <i>quadtree</i>	31
3 IMPLEMENTAÇÃO	35
3.1 Terreno infinito	37
3.1.1 Renderização de conteúdo	40
3.1.2 Otimização através de LOD	41
3.2 Relevo	43
3.2.1 Problemas na geração de relevo sob-demanda	43
3.2.2 Solução para geração de relevo sob-demanda	45
3.2.3 Otimização através de conservação de dados	47
3.3 Continentes	48
3.3.1 Oceano como um plano azul	48
3.3.2 Pré-processamento de faixas de terra	49
3.3.3 Visão micro e macro do mundo	50
3.3.4 Quebra de linearidade da costa	52
3.3.5 Praias	55
3.3.6 Arquipélagos e praias diferenciadas	58

4	RESULTADOS	61
4.1	Aspectos gerais da ferramenta	61
4.2	Análise de desempenho	62
4.3	Avaliação de técnicas e resultados obtidos	63
4.3.1	Avaliação dos continentes	63
4.3.2	Avaliação do relevo	65
4.3.3	Avaliação da costa	67
5	CONCLUSÃO E TRABALHOS FUTUROS	68
	REFERÊNCIAS	69

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 resultou 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, como em [9] e [6].

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 de economia e política e complexos. A quantidade de jogadores vinculados a um MMO pode chegar a ordem dos milhões, como é o caso do jogo World of Warcraft [2], com mais de 6 milhões de inscritos [3].

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 [2]

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 [24]. 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 nominados e ligados a história do jogo. A criação manual desses mundos virtuais exige uma equipe apta a 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 é útil para 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 completo, com continentes, oceanos, áreas planas, montanhas, etc, sendo que 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. A única limitação de tamanho que pode existir é a quantidade de informação armazenada em um inteiro sem sinal, visto que a posição do jogador é utilizada como semente para a geração do conteúdo a sua volta.

Além da geração de relevo, a ferramenta também é capaz de gerar oceanos e continentes, todos customizáveis. A forma dos blocos de terra é definida aleatoriamente com base em uma semente global, porém é possível que o usuário especifique se deseja mais terra do que água, maior frequência nas linhas da costa, etc. Para garantir uma boa performance na renderização dos gráficos, a malha do relevo e dos continentes utiliza o conceito de nível de detalhamento (LOD), o que faz com que as montanhas mais próximas do usuário tenham mais polígonos (resolução melhor), ao passo que as montanhas mais distantes apresentam menos detalhes.

A geração dos continentes e do relevo dentro deles são dois elementos distintos dentro da ferramenta. Embora o conceito de continentes e relevo estejam intimamente relacionados, a separação entre eles foi proposital para garantir uma maior flexibilidade no

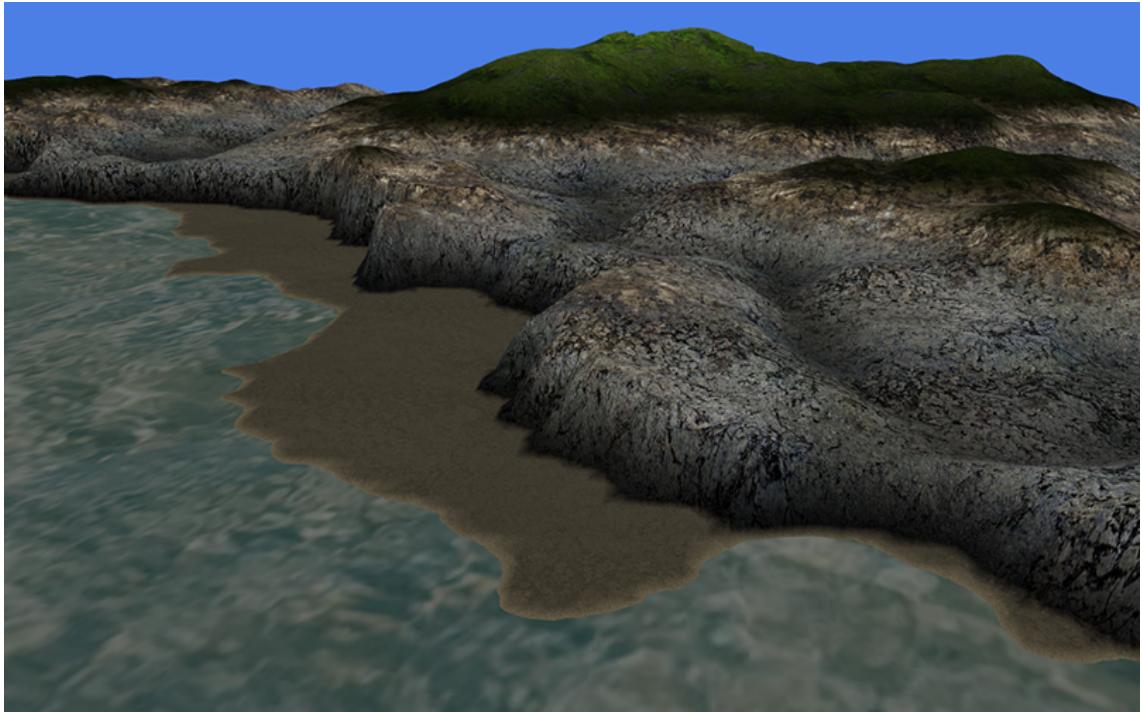


Figura 1.2: Ilustração do mundo virtual gerado

processo de gerar o mundo virtual. Utilizando essa abordagem híbrida, é possível que um planeta com poucos continentes, porém com uma alta frequência de relevo (montanhas pontudas) seja gerado facilmente. Para evitar que o jogador veja relevos atípicos frutos dessa abordagem híbrida, como a junção do continente com o oceano apresentar uma montanha partida pela metade ao invés de uma praia, a ferramenta utiliza um algorítimo para suavizar o relevo da costa. O nível de suavização pode ser customizado pelo usuário, o que garante uma maior flexibilidade ainda na geração de conteúdo. 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. 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 4 apresenta os resultados obtidos com a ferramenta desenvolvida, com análises de desempenho e ilustração das funcionalidades.

Por fim o capítulo 5 apresenta as considerações finais do trabalho. Além disso, apresenta-se sugestões de melhora 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 ou as camadas de cascas de uma árvore.

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 formas matemáticas geradas recursivamente, sendo que a cada nível de recursão mais e mais detalhes da mesma forma são gerados [14]. As formas geradas são auto-similares e geralmente a cada nível de recursão a forma original é vista novamente, com algumas alterações. Fractais possuem um nível de detalhes infinito, então quando mais próximo ele é visualizado, mais detalhes ele irá gerar. 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 [1]. 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

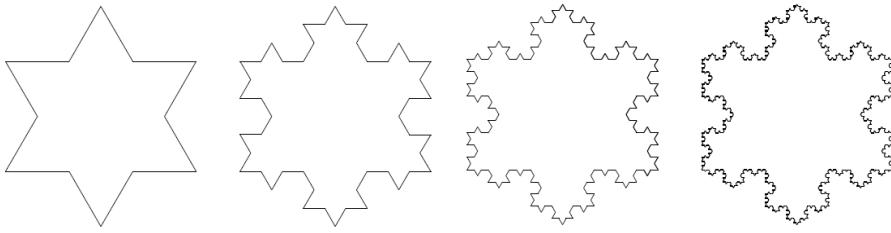


Figura 2.1: Quatro primeiras iterações do fractal de floco-de-neve de Koch [5]

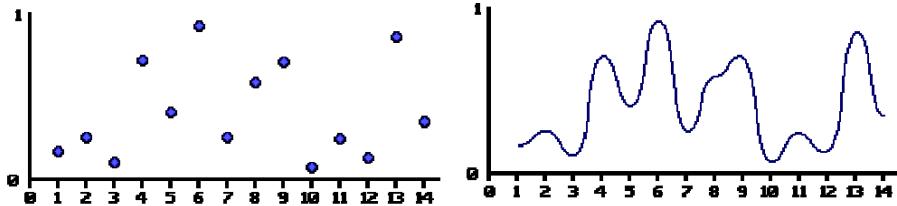


Figura 2.2: À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [7]

maioria heterogêneas, como a forma de um montanha: quando mais próximo do cume, mais pontiagudas as formas se tornam, ao passo que na base da montanha as formas são suaves e tentem a ser mais 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 in 1982, 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

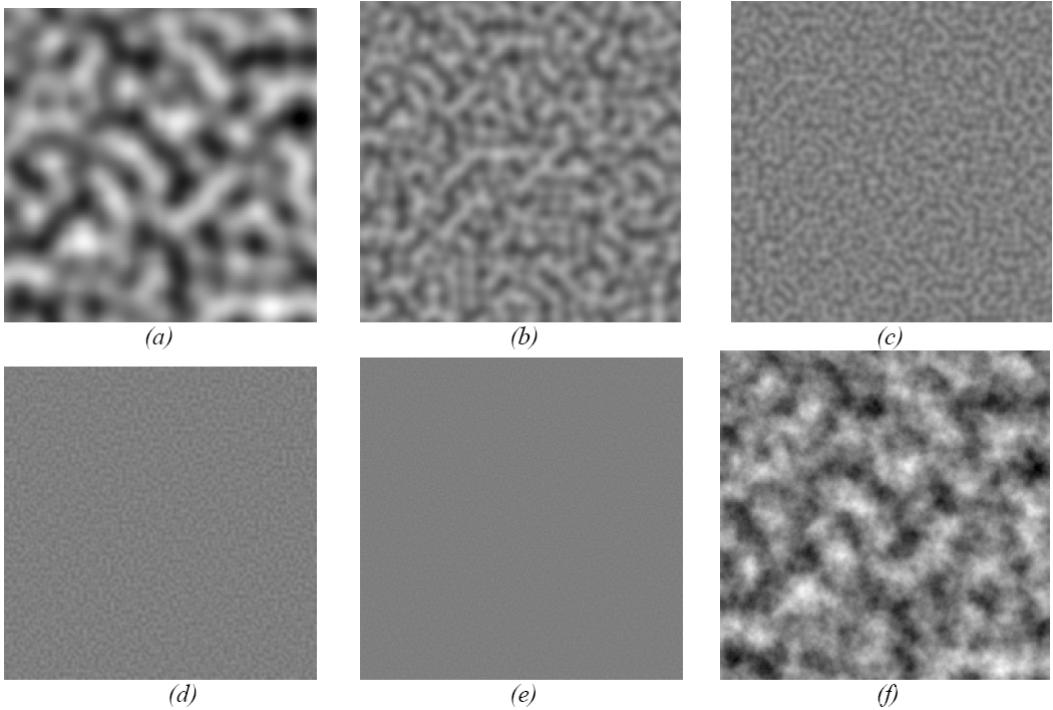


Figura 2.3: À direita, valores aleatórios gerados a partir de pontos discretos. À esquerda, interpolação suave dos pontos [7]

é criado. Em seguida, todos os pontos são interpolados entre si, criando um domínio de valores infinito. Para cada parâmetros 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 seu terreno. O terreno é o plano que irá guiar e servir como base para todos os demais elementos que compõem o mundo, como árvores, estradas, cidades, etc. Em computação gráfica, um das formas possíveis de se representar um terreno é através de um mapa de altura (*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 renderizar 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 algorítimo é o seguinte:

- Uma altura escolhida aleatoriamente é atribuída a cada um dos cantos do quadrado. Essa altura é proporcional ao tamanho do quadrado.
- O quadrado é dividido em quatro quadrados menores, com as alturas dos cantos de cada quadrado sendo calculada com base na interpolação das alturas dos cantos dos quadrados vizinhos ao quadrado original.
- O algorítimo repete os dois primeiros passos para cada quadrado gerado. A recursão é quebrada quando um certo nível de detalhes é atingido.

O resultado obtido com esse algorítimo 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 [16]. A figura 2.4 ilustra um relevo gerado a partir de divisões estocásticas.

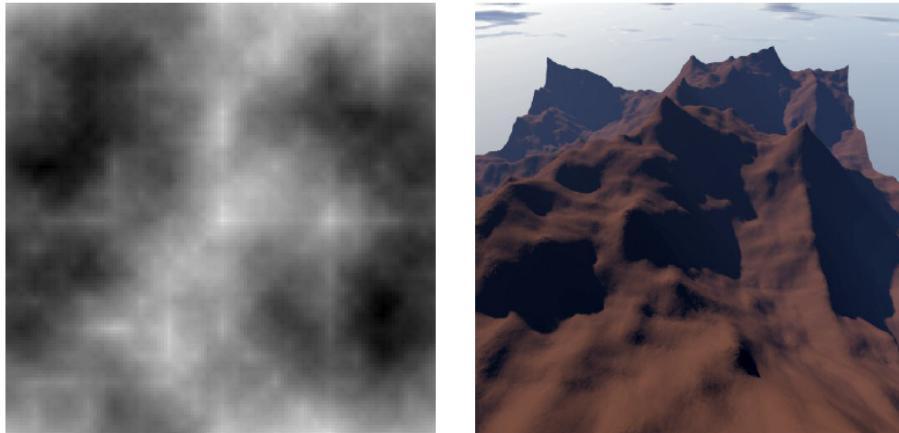


Figura 2.4: À esquerda, *heightmap* gerado a partir de divisões estocásticas; à direita, o mesmo *heightmap* renderizado em 3D [5]

Existem variações desse algorítimo 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° 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 com todos os pontos zerados, o algorítimo 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 algorítimo 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 algorítimo de subdivisões estocásticas, porém ele é muito lento para ser utilizado. Depois que a linha divisória é escolhida, todos os pontos da matriz precisam ser atualizados e isso é feito a cada iteração do algorítimo.

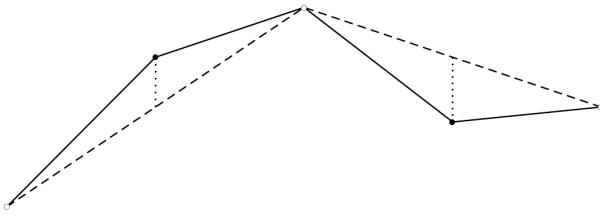


Figura 2.5: Geração de relevo pela disposição do ponto médio (perspectiva 2D) [5]

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, rolando 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 a partícula para de se movimentar.

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 algorítimo 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 algorítimo apresenta uma complexidade de implementação maior que os demais já citados.

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 algorítimo acharia o ponto médio desse segmento e, utilizando um número pseudo-aleatório, aumentaria ou diminuiria a altura desse ponto em uma determinada quantidade. Em seguida, essa quantidade utilizada para perturbação é reduzida e o algorítimo é aplicado novamente aos dois novos segmentos gerados a partir da perturbação do ponto médio. A figura 2.5 ilustra o funcionamento do algorítimo.

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 com-

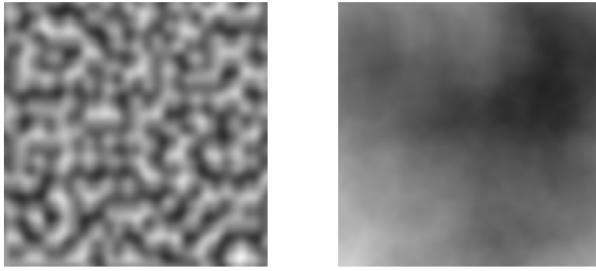


Figura 2.6: À direita, *heightmap* gerado a partir de ruído de Perlin. À esquerda, *heightmap* gerado a partir da combinação de camadas de ruído de Perlin (turbulência). [11]

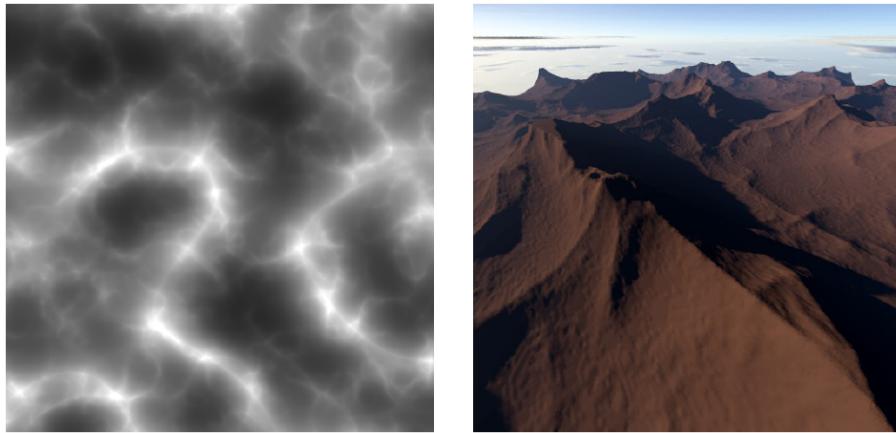


Figura 2.7: À esquerda, *heightmap* gerado a partir da técnica de ruído de Perlin *ridged*; à direita, o mesmo *heightmap* renderizado em 3D [11]

binação de diversas camadas de ruído, porém, é capaz de produzir um efeito mais natural e convincente. Cada uma dessas camadas, chamadas octavas, possui uma amplitude e uma frequência, e a combinação delas é comumente chamada de turbulência de Perlin. A figura 2.6 ilustra os resultados obtidos com essa técnica.

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 ruido 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 algorítmico original é modificado para que os valores -1 e 1 gerem relevos de baixa altitude, ao passo que os valores gerados no meio do intervalo $[-1, 1]$ geram valores de grande altitude. A figura 2.7 ilustra os resultados obtidos com essa abordagem.

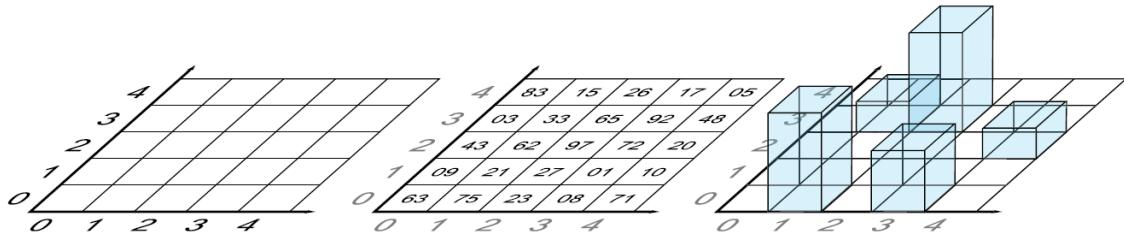


Figura 2.8: Mundo virtual dividido em células. Cada uma delas possui uma semente calculada com base em sua posição e com base uma semente global referente à cidade [8]

2.3 Mundos pseudo-infinitos

2.3.1 Cidade virtual

A geração de conteúdo 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 [8], cujo objetivo era gerar uma cidade virtual que fosse visualmente interessante e composta por construções complexas, porém cada uma sendo criada a partir de elementos mais simples.

Na abordagem utilizada, os autores dividiram 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 [26]. 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 2.8 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 realística. Depois que a geometria da construção está pronta, ela é texturizada com janelas, sendo que o tipo da janela é escolhido aleatoriamente. A figura 2.9 ilustra a geração de construções.

Para garantir o uso racional de recursos computacionais, como memória e processamento, os autores utilizaram o conceito nomeado por eles de preenchimento por *view*

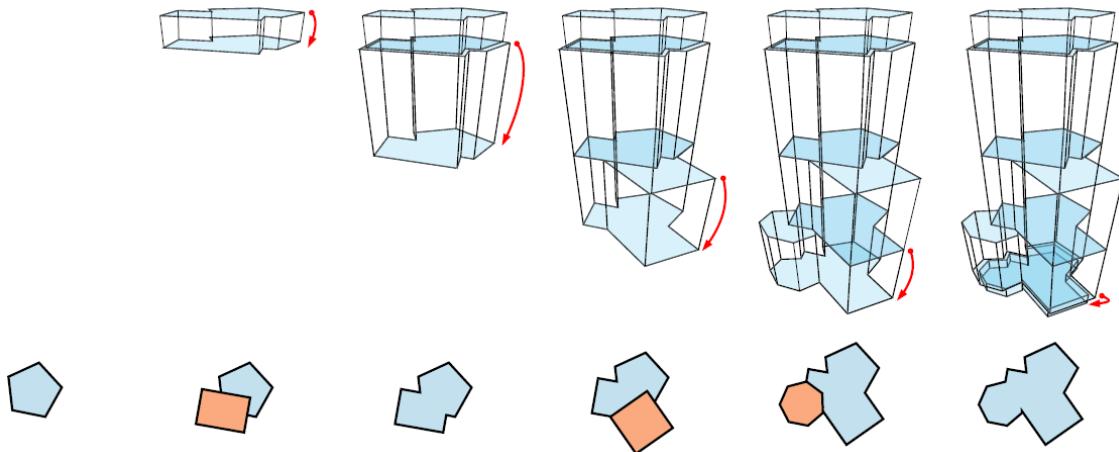


Figura 2.9: Geração de construções: cada andar é gerado pela mescla de polígonos escolhidos e centralizados aleatoriamente [8]

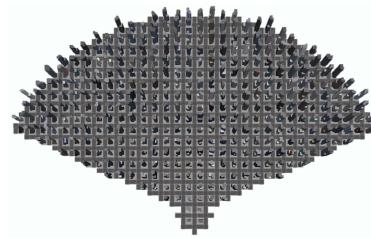


Figura 2.10: Campo de visão do usuário: apenas as células visíveis tem o seu conteúdo gerado [8]

frustum, que consiste em restringir à geração de conteúdo somente para as células que estão dentro do campo de visão do usuário. A medida que esse anda pela cidade virtual, novas células vão sendo adicionadas ao campo de visão e seu conteúdo é gerado; 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 estiverem a uma certa distância do usuário e dentro de um ângulo de 120° de visão. A figura 2.10 ilustra o funcionamento do campo de visão descrito e a figura 2.11 mostra uma cidade virtual gerada.

2.3.2 Geração e texturização por camadas

Outro trabalho analisado foi o de [11] para a construção da ferramenta SkyCastle [12], uma *engine* para jogos online multijogador com suporte à geração procedural de mundos virtuais. No referido trabalho, o autor foca o problema de mundos virtuais cada vez maiores em jogos de computador e aplicações, o que cria a necessidade de desenvolvimento de ferramentas capazes de ajudar na geração de conteúdos realistas para

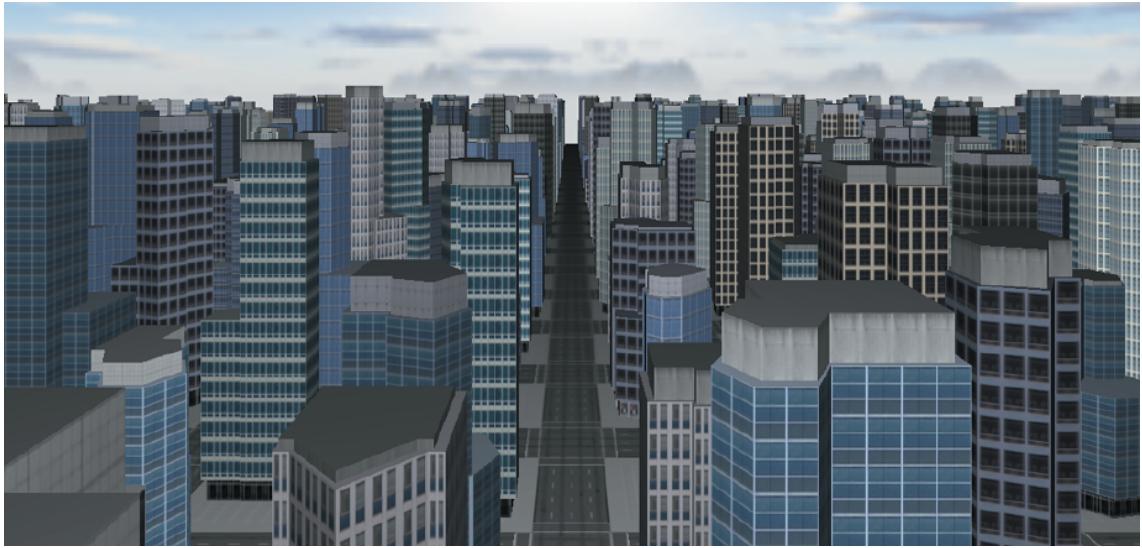


Figura 2.11: Exemplo de cidade virtual gerada [8]

esses mundos. Para a geração de relevo, o autor utiliza 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 será abordado em mais detalhes na seção 2.2.

Depois do assunto de criação de relevo, o autor aborda a funcionalidade de 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 [4]; 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 e sem falhas. 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, o autor sugere a utilização de uma abordagem proposta por [15], que consiste na utilização de um conjunto de texturas pré-definidas em formato de borda juntamente com a texturização em reticulados já citada. O funcionamento do algoritmo resume-se em aplicar as texturas em borda sobre um plano já uniformemente texturizado, porém utilizando transparência nas áreas não de-

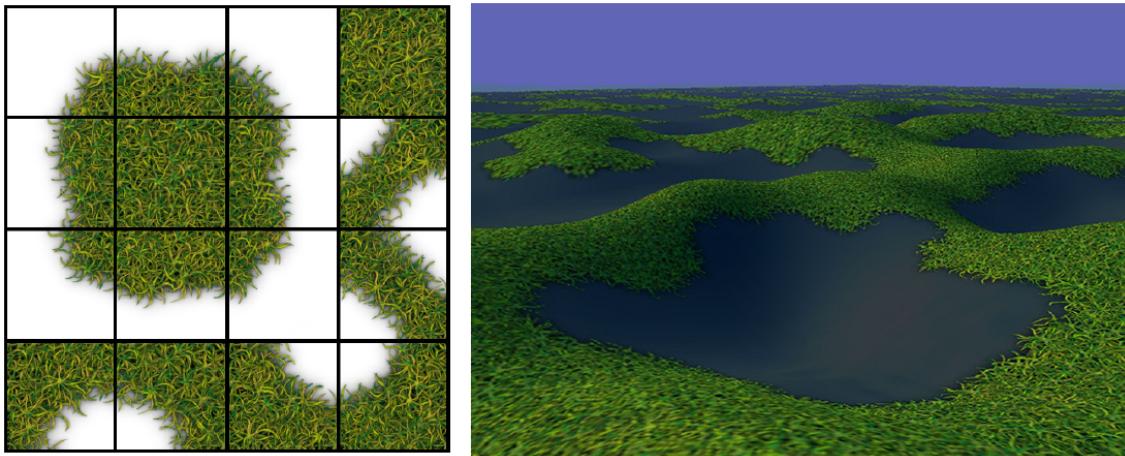


Figura 2.12: Terreno texturizado com duas camadas: bloco escuro e um conjunto de texturas em formato de borda com transparência [11]

senháveis das texturas em borda. Dessa forma, a sobreposição das bordas sobre o plano texturizado irá produzir uma paisagem menos homogênea, o que gera um resultado visual melhor. A figura 2.12 ilustra as texturas em forma de borda e a sua utilização na abordagem descrita.

Para ornamentar o mundo virtual, o autor cita a utilização de plantas proceduralmente geradas através de três métodos principais: *L-System* [23], geração baseada em componentes [18] e árvores parametrizadas [13]. 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, etc); a cadeia de caracteres resultante é obtida a partir da aplicação sucessiva de regras sobre uma cadeia base. O algorítimo de geração baseada 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

Outro trabalho analisado é a geração de planetas procedurais através de fractais e combinações deles com outros métodos [17]. Na abordagem da autora, o mundo gerado não é infinito, porém ele é esférico e simula a visualização do planeta Terra. Partindo da subdivisão recursiva de um octaedro, a autora cria um mundo esférico que serve como base para a aplicação dos algorítimos de relevo. Através de uma interface, o usuário pode estipular qual algorítmico de relevo ele deseja utilizar, e o resultado desse processo é traduzido em uma lista encadeada que descreve todos os vértices do mundo. Uma lista encadeada foi utilizada para garantir que a malha resultante possa ter seus detalhes aumentados e/ou reduzidos facilmente (através das funções de inserção e remoção de elementos da lista), mesmo que essa estrutura de dados não seja a melhor escolha para busca de um elemento em específico, por exemplo. A utilização de listas encadeadas é importante porque a autora utiliza o conceito de LOD (*level of detail* [10]) para garantir um bom desempenho da aplicação. Através dessa técnica, nos polígonos que estão longe do campo de visão do usuário são mostrados em resolução baixa (poucos triângulos), enquanto os polígonos que estão próximos ao usuário são mostrados com resolução alta (muitos triângulos), o que aumenta a riqueza de detalhes dos objetos próximos. Com a utilização da lista encadeada, a inserção/remoção de novos vértices como consequência da atuação do algorítmico de LOD torna-se uma tarefa menos complexa.

O relevo gerado pela autora é obtido através de diversas técnicas, cada uma com suas peculiaridades e resultados. Dentre as técnicas utilizadas, encontram-se: geração por falhas aleatórias, disposição do ponto médio (incluindo uma variação multifractal) e ruído de Perlin (e suas diversas variações). A figura 2.13 ilustra os relevos obtidos com essas técnicas. Depois que a malha é gerada, a autora utiliza uma combinação de técnicas para colorir os pontos da falha; as técnicas variam conforme o resultado desejado, porém todas elas são baseadas na interpolação de cores parametrizada pela altura do ponto sendo analisado. Para garantir um aspecto mais real, utiliza-se em algumas técnicas conjuntamente com ruídos e turbulência, como o ruído de Perlin.

Tento em vista que autora explora diversos métodos para geração de relevo, é importante salientar quais foram os resultados obtidos com a utilização de cada um deles frente

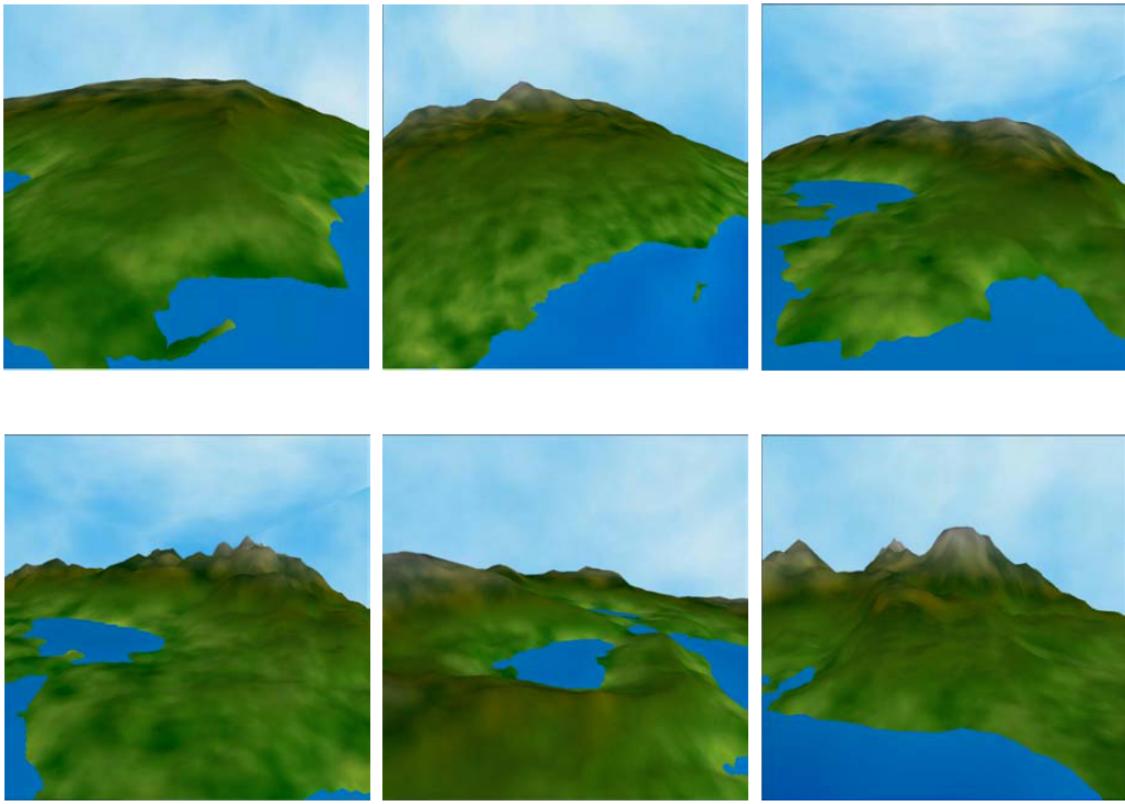


Figura 2.13: Diferentes relevos gerados pela utilização de fractais e suas combinações. [17]

ao problema de geração de continentes para um mundo esférico. Uma visão mostrando todos os planetas gerados é mostrada na figura 2.14. O algorítimo de geração por **fallhas aleatórias** produz terrenos que parecem quebrados e diversas ilhas pequenas; de acordo com autora, os resultados obtidos em sua grande maioria são muito aleatórios e na média o algorítimo produz um grande continente e um grande oceano. O algorítimo de **disposição do ponto médio** e sua variação multifractal geram continentes grandes e compactos 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 (tento em vista que há grandes variações de altitude), segundo a autora os resultados não são bons quanto os obtidos com ruído de Perlin. O algorítimo 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 mostram 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 afetados por erosão

Outro trabalho analisado foi a geração de terrenos procedimentais em tempo real com a utilização de fractais afetados por erosão [21]. Conforme ressaltado pelo autor, o aumento do poder de processamento de computadores domésticos possibilitou que jogos de computador façam uso de simulações de erosão em tempo quase real. Utilizando essa premissa, o autor agregou 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 *pontos 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 algorítimo de geração de ruído para produzir montanhas menos pontiagudas e, para remover as linhas bem definidas criadas pelo diagrama de Voronoi, um filtro de perturbação ainda é aplicado. A figura 2.15 ilustra o processo descrito.

Para a simulação de erosão no mapa de altura gerado, o autor utiliza 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 do relevo. Aplicando uma série de otimizações sobre o algorítimo original proposto por [20], o autor obteve 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. O autor criou, então, uma mescla entre as duas técnicas, alterando o método de erosão hidráulica para que ele ficasse mais próximo ao método de erosão termal. O resultado obtido é uma técnica capaz de gerar resultados visuais tão bons quanto

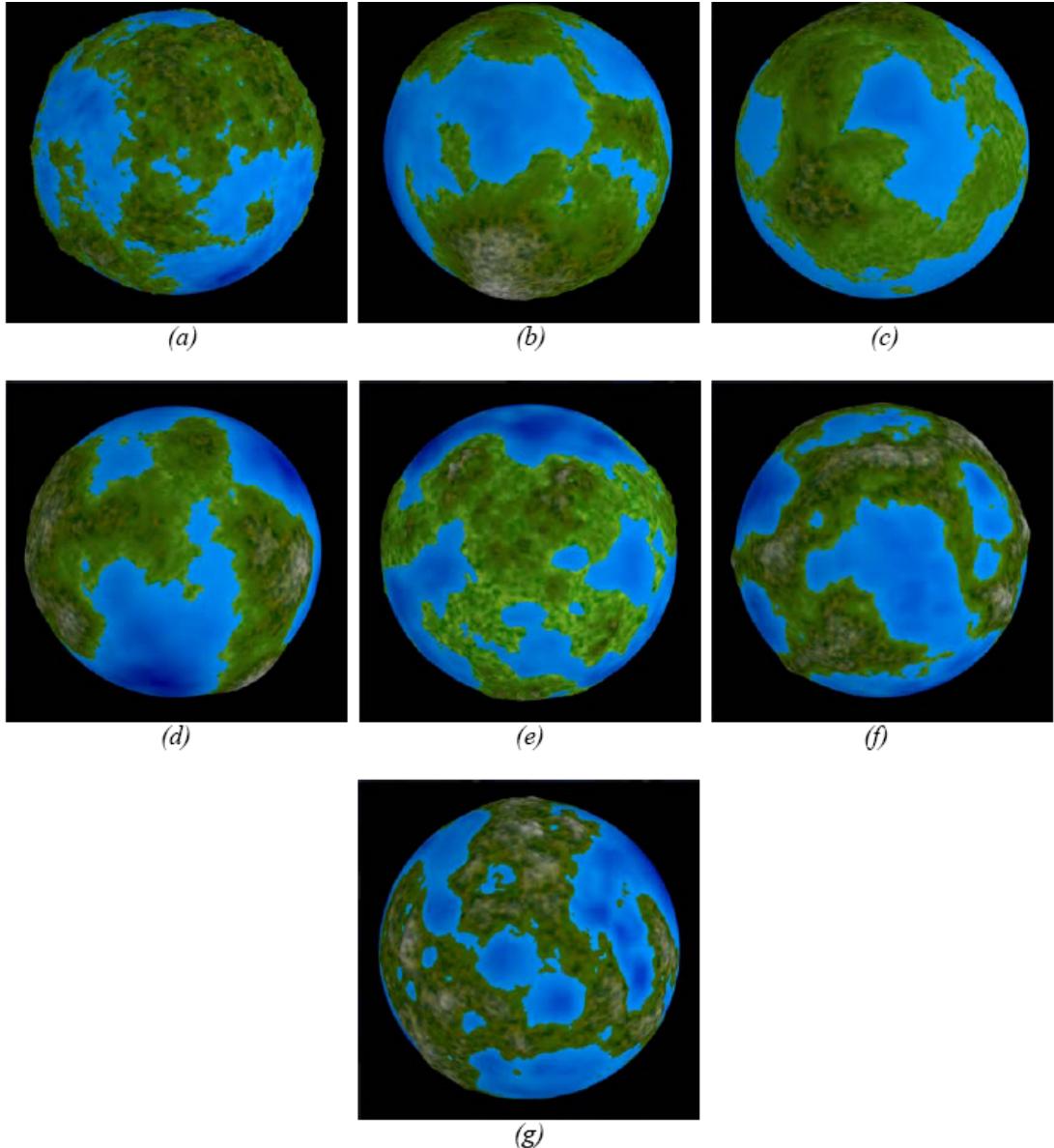


Figura 2.14: Planetas gerados com os diversos algorítimos. (a) Falhas aleatórias. (b) Disposição do ponto médio. (c) Disposição do ponto médio multifractal. (d) Ruído de Perlin. (e) Ruído de Persil Multifractal. (f) Ruído de Perlin *Ridged*. (g) Ruído de Perlin *Ridged* Multifractal. [17]

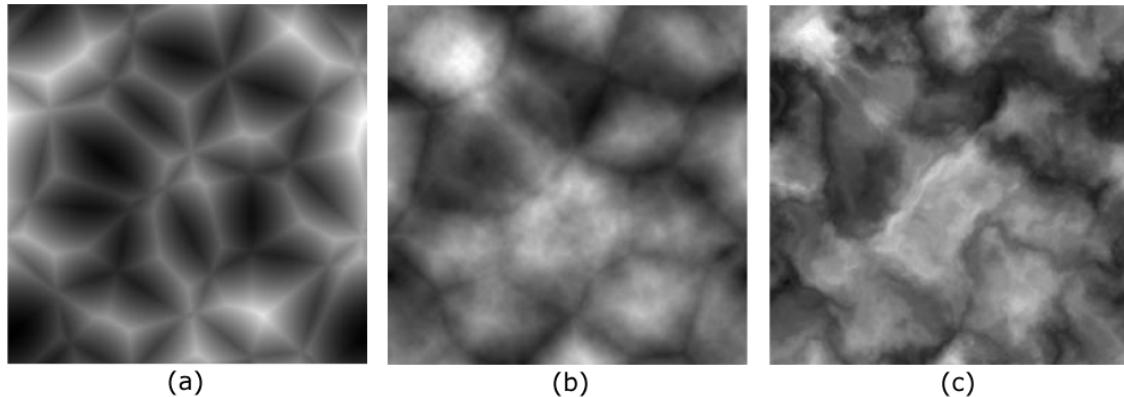


Figura 2.15: (a) Mapa de altura gerado a partir de um diagrama de Voronoi. (b) Aplicações de funções de ruído em (a). (c) Mapa resultante da aplicação de um filtro de perturbação em (b). [21]

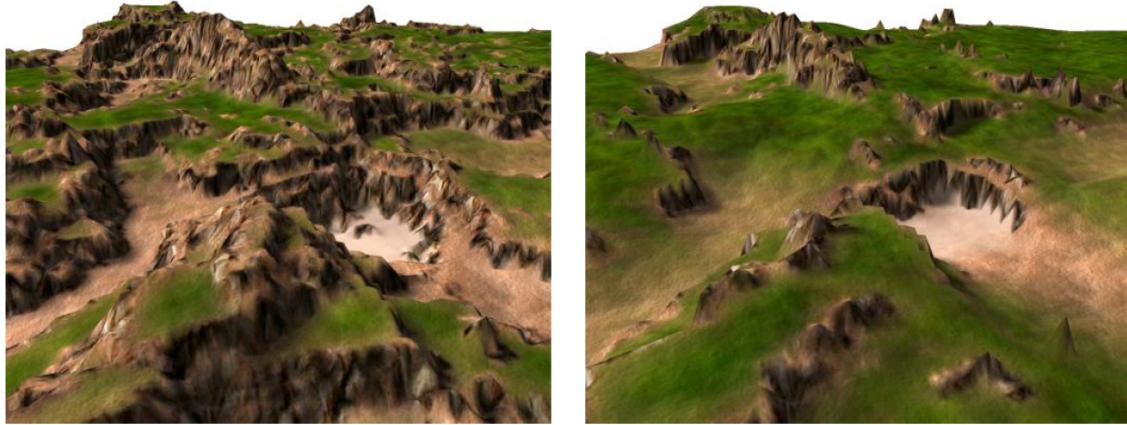


Figura 2.16: À direita, renderização de um mapa de altura sem a aplicação de erosão; à esquerda, o mesmo mapa com a aplicação de erosão. [21]

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 2.16 ilustra o resultado final obtido pelo autor.

2.3.5 Divisões estocásticas de uma *quadtree*

Outro trabalho analisado foi a geração de mundos virtuais proceduralmente através de técnicas de subdivisão estocásticas [5]. O objetivo do autor foi criar um mundo virtual de grandes proporções, porém gerando o seu conteúdo sob-demanda através procedimentos parametrizáveis e com multi-resolução. Para a geração do relevo do terreno, o autor utiliza sub-divisões de uma *quadtree* em um processo recursivo, como ilustra a figura 2.17. 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

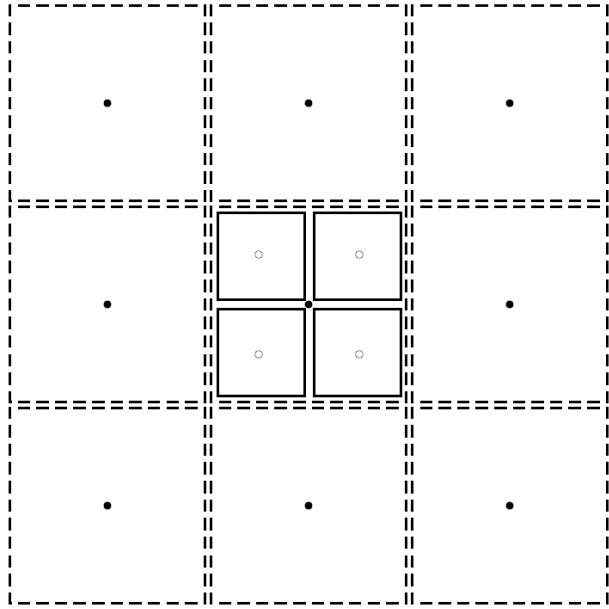


Figura 2.17: *Quadtree* utilizada para a geração do terreno [5]

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 2.17, 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, o autor necessitou 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, o autor utilizou 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 removidos. A figura 2.18 ilustra o funcionamento do processo através de uma árvore binária; 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. Como ilustrado na figura, à medida que a câmera se move para a direita, 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 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, o autor buscou 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 pelo autor para essa busca é utilizar 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 pelo autor, 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.

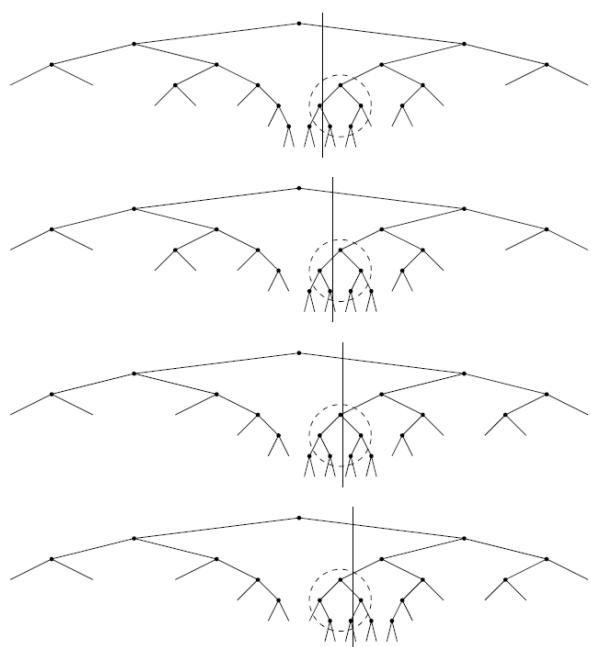


Figura 2.18: Subdivisão de células e níveis de detalhe representados por uma árvore binária em 2D. [5]

3 IMPLEMENTAÇÃO

A ideia original do trabalho foi desenvolver um mundo virtual completo, semelhante em grande parte com o mundo real. 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. A combinação de todos esses elementos seria capaz de criar um mundo virtual muito próximo da realidade, fato que seria de suma importância para garantir um bom resultado da ferramenta.

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 3.1 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, ele 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

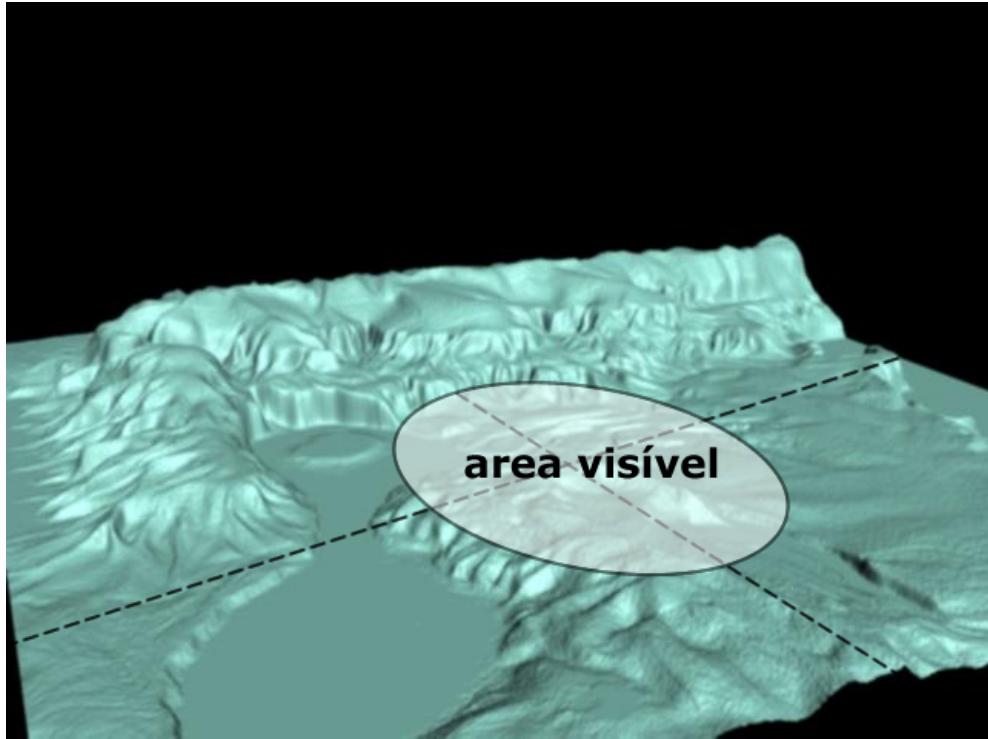


Figura 3.1: Problema da geração de conteúdo sob demanda.

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 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.

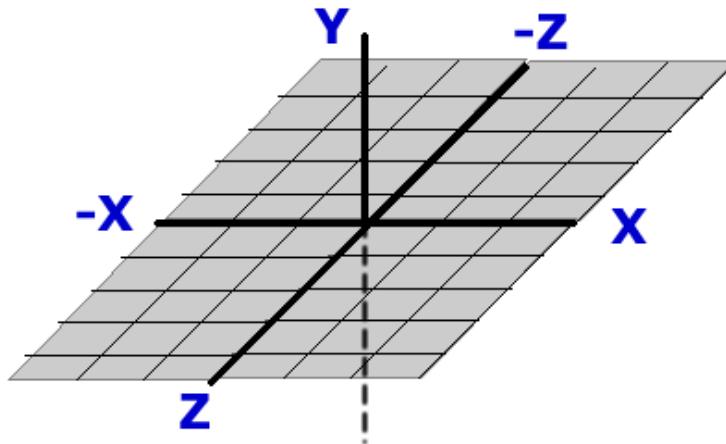


Figura 3.2: Organização do sistema de coordenadas do mundo virtual

3.1 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 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 [8]. Na abordagem dos autores em questão, o mundo virtual pseudo-infinito é dividido em células quadradas e, à medida que o usuário anda, as células são adicionadas e/ou removidas do campo de visão. 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 3.2 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

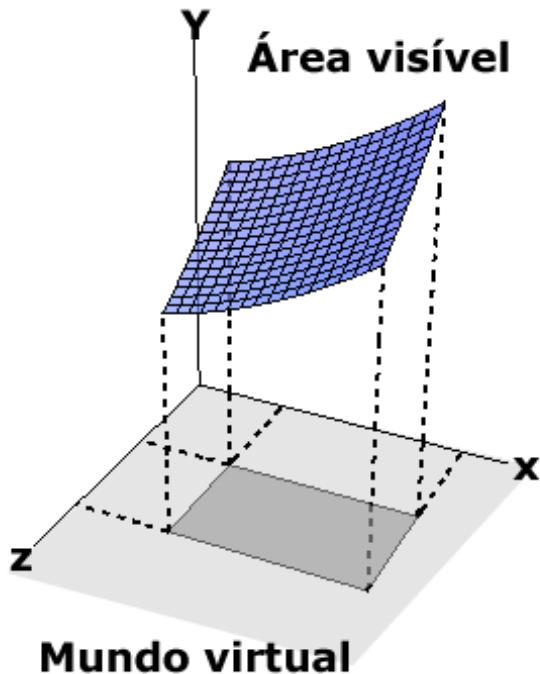


Figura 3.3: Campo de visão do usuário: a área visualizada corresponde a uma fatia do mundo virtual existente.

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. Diferentemente do que foi feito em [8], no qual o campo de visão é um cone, o campo de visão da presente ferramenta é um quadrado centrado no usuário. A figura 3.3 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.

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 3.2, 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 saída alternativa 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 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.

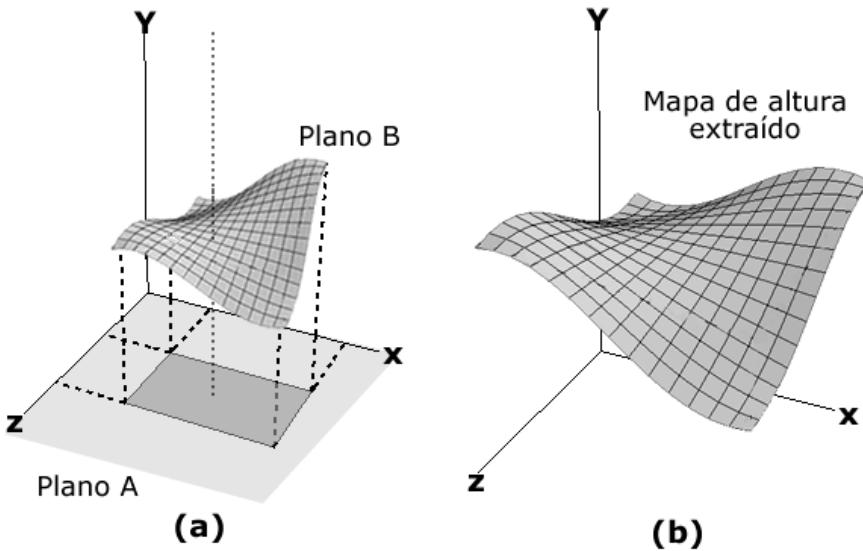


Figura 3.4: Mapeamento das coordenadas do mundo virtual para as coordenadas de desenho da tela

3.1.1 Renderização de conteúdo

Depois que o modelo de controle da geração de terrenos infinitos foi definido, iniciaram-se os trabalhos de computação gráfica para que as informações pudessem ser desenhadas 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 3.4(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 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 extrair essa fatia e, então, mapeá-la para um mapa de altura (*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 3.4 (b). O mapa de

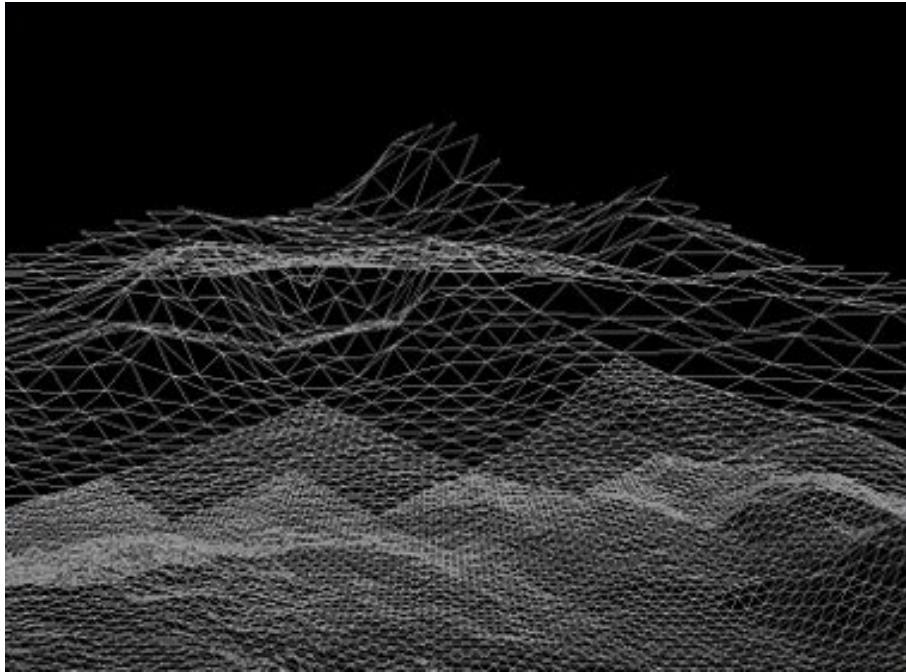


Figura 3.5: Exemplo de LOD aplicado a um terreno virtual

altura gerado é renderizado como uma malha triangular.

3.1.2 Otimização através de LOD

Durante a implementação da malha do terreno, observou-se que o desempenho da aplicação caia consideravelmente à medida que o tamanho da área visível pelo usuário era aumentada. Isso pode ser explicado pelo consumo cada vez maior de processamento necessário para que o grande número de polígonos na tela seja renderizado. Para contornar esse problema, utilizou-se o conceito de LOD (*Level of Detail*). A técnica de LOD consiste em aumentar a resolução dos elementos que estão mais próximos da visão do usuário e, em contra-partida, reduzir a resolução dos elementos que estão longe (justamente por estarem distantes do observador, esses elementos não são expressivos). A figura 3.5 ilustra o funcionamento da técnica de LOD em um terreno virtual; a parte do terreno que está mais próxima do usuário é composta por muito mais triângulos que a parte mais distante (uma espécie de montanha, ao fundo).

Uma das melhores abordagens de LOD encontradas para aplicação na ferramenta proposta foi a renderização de terrenos massivos utilizando-se LOD em bloco [25]. O trabalho analisado apresenta resultados bons em relação a aumento de desempenho frente a uma grande quantidade de polígonos, que é o caso do terreno do mundo virtual, porém essa técnica é voltada para um conjunto de dados estático. Partindo do fato que a ferra-

menta utiliza um conjunto de dados mutável (cada vez que o usuário se move, uma nova fatia do mundo virtual é coletada), uma outra abordagem teve de ser utilizada. A solução desejada, então, deveria obrigatoriamente trabalhar com um conjunto variável de dados e, ao mesmo tempo, apresentar um desempenho tão bom quanto ou melhor ao que a solução de LOD analisada previamente. Optou-se, então, pela utilização de um algoritmo de LOD contínuo em tempo real de apenas uma passada [22].

Na abordagem desenvolvida por [22], uma *quadtree* é utilizada para gerenciar os quadrantes que compõem o terreno. Depois que a árvore é construída na memória, ela é atualizada com as informações referentes às alturas de cada um dos pontos do terreno; independente da variação do conjunto de dados, a árvore nunca é destruída, ela é apenas atualizada, o que garante um aumento ainda maior de desempenho. Depois que todos os dados são coletados e catalogados, o algoritmo percorre a árvore recursivamente analisando os pontos da malha que estão em cada um dos quadrantes e, depois que essa análise é finalizada, o quadrante em si é desenhado na tela. Em linhas gerais, o algoritmo verifica a resolução (nível de detalhamento) da malha nos quadrantes vizinhos em comparação com a resolução da malha do quadrante que está sendo analisado e, também, a distância que o quadrante está do observador. Se a resolução do quadrante sendo analisado é muito baixa em relação aos seus vizinhos, ou se a distância entre ele e o observador é próxima o suficiente para exigir mais detalhes, então o quadrante em questão é dividido em mais quadrantes e, para cada um dos filhos gerados, o algoritmo é aplicado novamente. Quando a resolução do quadrante sendo analisado estiver em conformidade com os seus vizinhos e com a visão do observador, então a recursão termina. Ao final, os quadrantes que possuíam muitas informações e/ou estavam próximos ao observador sofrem diversas subdivisões e formam uma malha com muitos polígonos, enquanto os quadrantes com poucas informações ou mais distantes do observador formam uma malha com menos polígonos. A figura 3.6 ilustra os resultados obtidos com o algoritmo descrito. Na regiões com maior quantidade de informações (deformações e quinas do desfiladeiro), há uma quantidade muito maior de polígonos do que as regiões com menos informações (as áreas planas ao redor do desfiladeiro).

A árvore utilizada pelo algoritmo é construída e mantida por um vetor, o que garante uma indexação mais eficiente dos elementos e um ganho de desempenho considerável. Aliando-se esse melhoramento com a abordagem de apenas uma passada mais a fun-

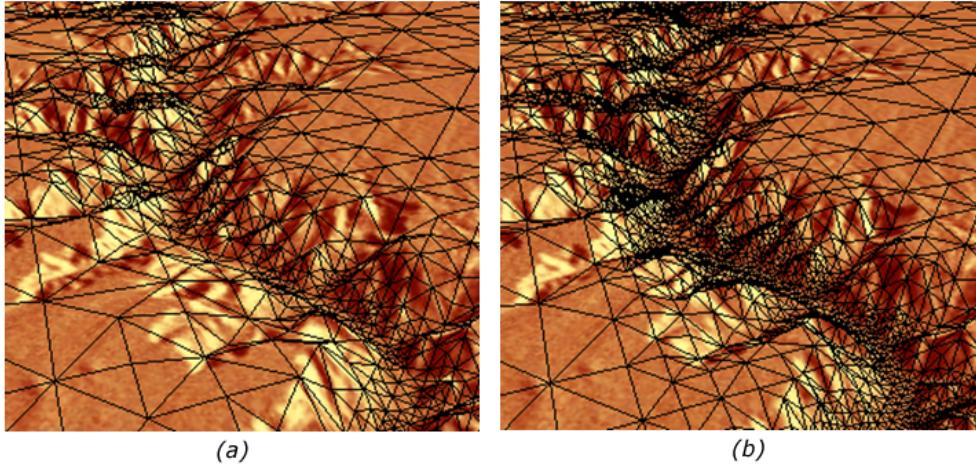


Figura 3.6: Terreno renderizado com baixa resolução (a) e alta resolução (b), ambos com aplicação de LOD [22]

cionalidade de que a árvore só precisa ser atualizada, não destruída e reconstruída cada vez que novos dados são coletados, tem-se um algorítimo de LOD flexível e rápido.

3.2 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.

3.2.1 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 de relevo tão grande, de uma só vez, é fisicamente 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.

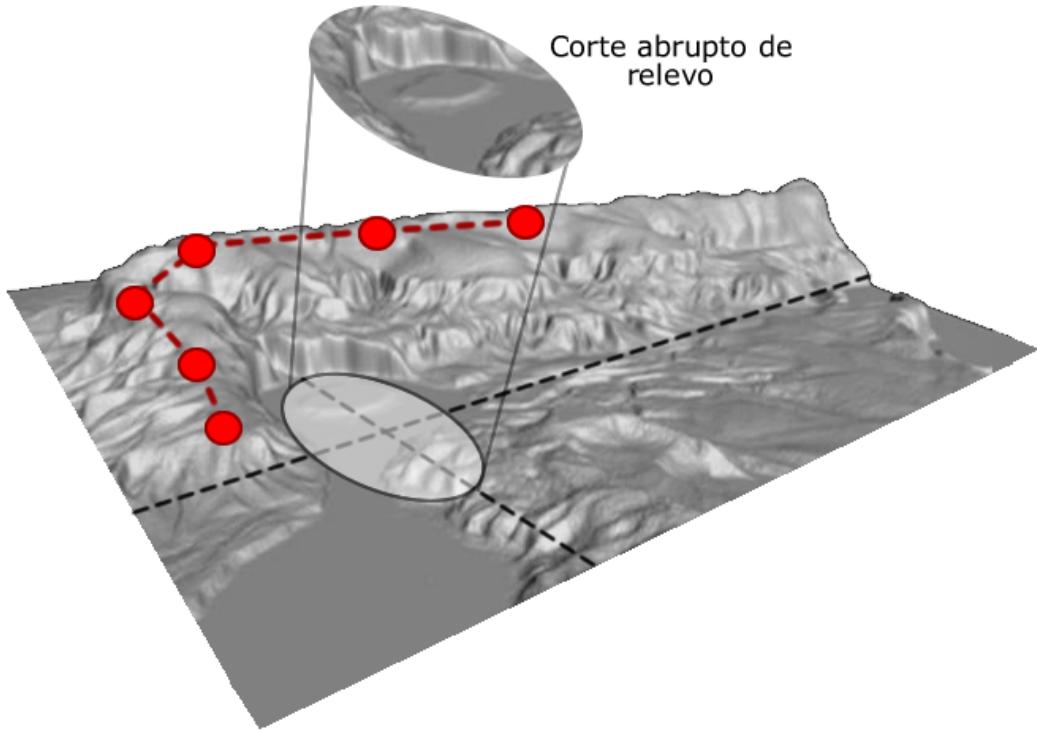


Figura 3.7: Geração de conteúdo com base apenas nas informações do campo de visão

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 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 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 continuidade, porque se o usuário costear a espinha dorsal da montanha, sem que os pontos chave entrem 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 3.7 ilustra esse problema.

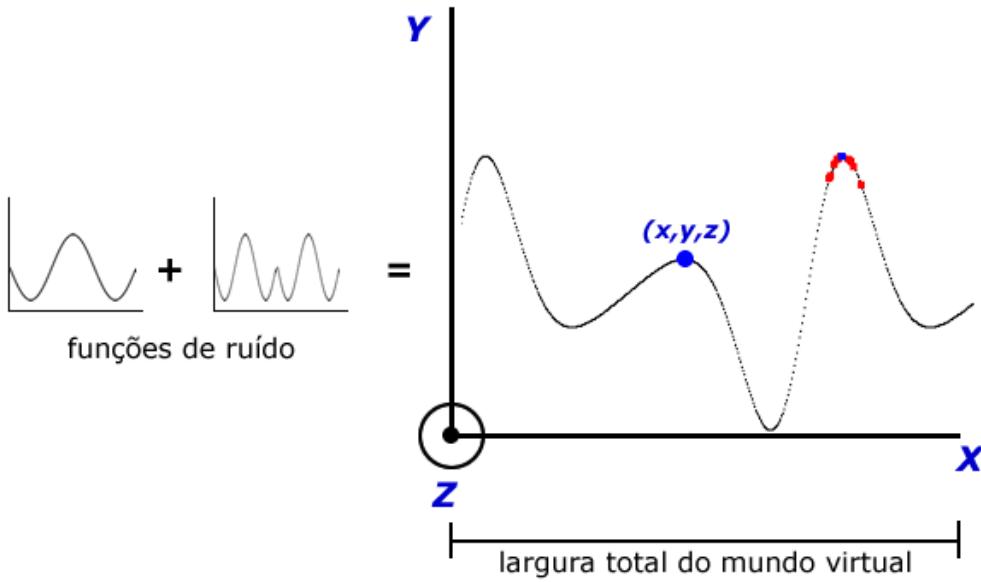


Figura 3.8: Função de geração de relevo parametrizável e o resultado gerado

3.2.2 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 3.8 ilustra o funcionamento da função de relevo aplicada ao mundo virtual gerado.

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

[8], 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 existe no mundo real. Conforme apontado por [17], 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 [12], 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. Para contornar esse problema, substitui-se a sobreposição de mapas de ruído pré-computados pela sobreposição em tempo real apenas do ruído dos pontos que estão dentro do campo de visão do usuário. Em suma, a ferramenta gera em tempo real mapas de ruído do tamanho exato da fatia do mundo virtual que o usuário está vendo e, então, sobrepõe esses mapas com o mapa de altura original (que também é uma fatia do mundo virtual).

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 [21]. 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 oitas 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.

3.2.3 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 alterados.

Tomando vantagem desse fato, a ferramenta utiliza um algorítimo 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 de 2 a 3 FPS. 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 4.

3.3 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.

3.3.1 Oceano como um plano azul

A primeira abordagem utilizada para a geração de água foi uma semelhante à proposta por [17], 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 algorítimos de geração de relevo, uma vez que eles não precisam classificar os vértices como sendo "terra" ou "mar". 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. Em testes realizados, o relevo gerado foi equilibrado o suficiente para produzir continentes grandes e, dentro deles, variações de alturas consideráveis como montanhas e planícies. 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.

3.3.2 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.

O algoritmo utilizado para a criação dos continentes é uma versão simplificada do gerador de planetas de [19] que consiste na geração de mundos redondos através da subdivisão recursiva de um tetraedro. Inicialmente um tetraedro irregular que envolve a esfera do mundo é gerado e valores aleatórios de altura e semente são atribuídos a cada um dos vértices. Em seguida, ele é dividido em dois por um plano que corta a figura no ponto médio entre o vértice mais distante e os vértices opostos a esse. Em suma, cada um dos dois tetraedros resultantes terão três vértices a partir do tetraedro original mais um novo vértice posicionado no ponto médio do vértice mais distante. A altura e a semente dos novos vértices são calculados a partir da altitude e semente dos pontos finais da linha que é dividida. O processo é repetido recursivamente para cada um dos dois tetraedros resultantes. O resultado final do gerador é um planeta completo, com demarcação de continentes e oceanos, além de informações sobre altura do relevo e profundidade do mar. 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 razão principal pela qual optou-se por utilizar uma abordagem diferente de um plano azul para gerar os oceanos e continentes. A figura 3.9, (a) e (b), ilustra os resultados obtidos com o gerador de planetas de [19].

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 3.2. 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 3.9, ilustração (c),

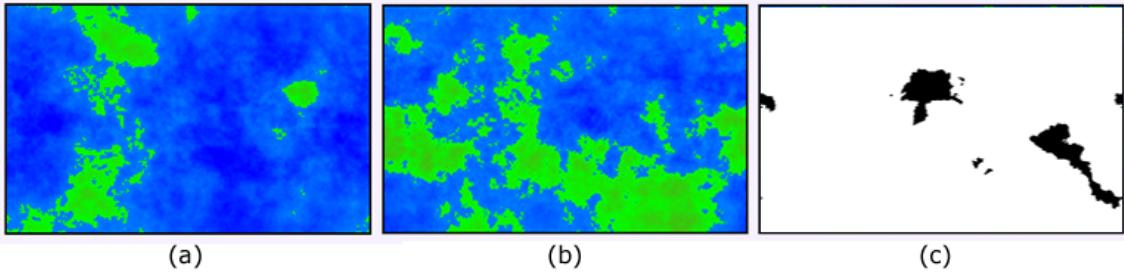


Figura 3.9: Planetas aleatórios gerados com diferentes sementes de entrada. (a) e (b) mapas com informações de relevo [19]; (c) mapa com informações somente sobre o que é terra/mar.

mostra uma mapa gerado contendo apenas informações sobre o que é terra (em preto) e o que é água (em branco).

3.3.3 Visão micro e macro do mundo

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 o que é terra e o que é mar são armazenadas em uma matriz, chamada **macro-matriz** (MM), que é utilizada para consulta pelos demais algorítimos da ferramenta. 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 macro-matriz com as proporções do mundo inteiro. Uma vez que a MM é menor que o mundo virtual, uma entrada (i, j) dela corresponde a diversos pixels dentro do mundo virtual. A figura 3.10 ilustra o mapeamento da MM para o mundo virtual.

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 píxel 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 é o bastante para que informações suficientes sejam processadas pelos demais algorítimos da ferramenta.

A figura 3.11 ilustra o resultado gráfico obtido pela ferramenta quando nenhum algo-

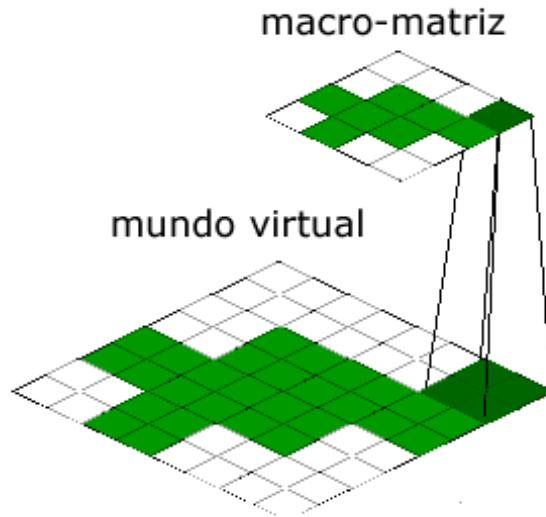


Figura 3.10: Mapeamento da macro-matriz para o mundo virtual. Como consequência da baixa resolução da matriz, a costa dos continentes é em linha reta.

ritmo 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 rednerização 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

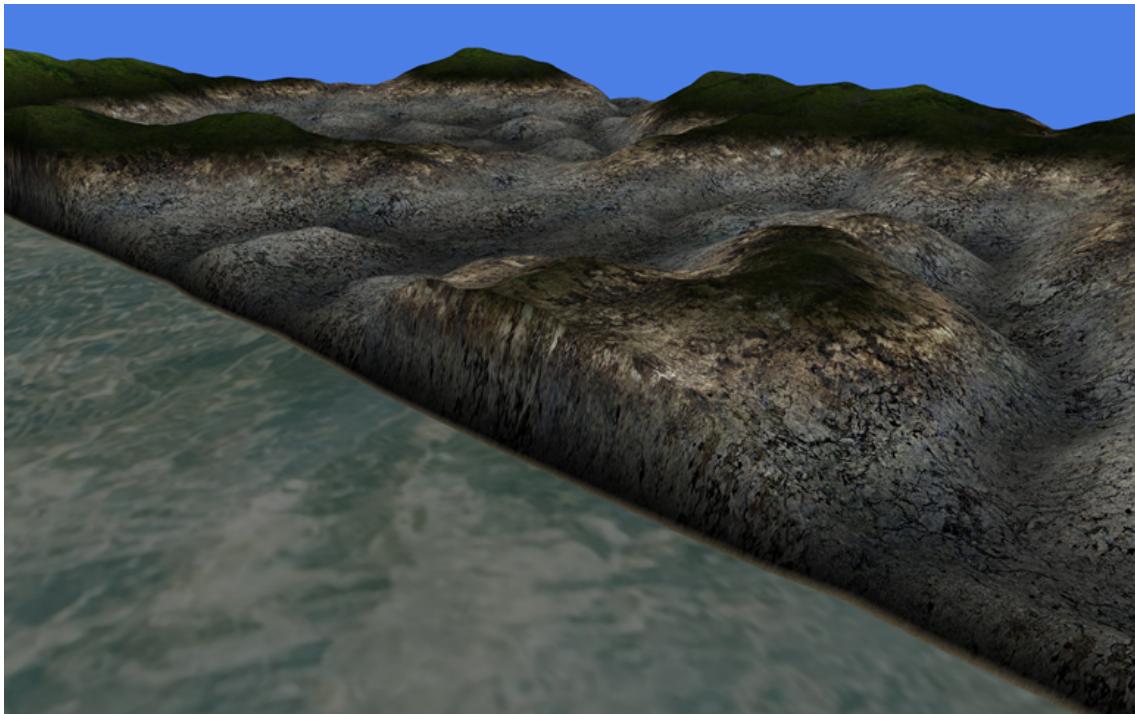


Figura 3.11: Resultado gráfico obtido quando nenhum algoritmo extra de geração de conteúdo é utilizado para preencher as discrepâncias de mapeamento da MM

mesmo sentido (que é a continuação da costa atual).

3.3.4 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 uma 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 lineariedade 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-

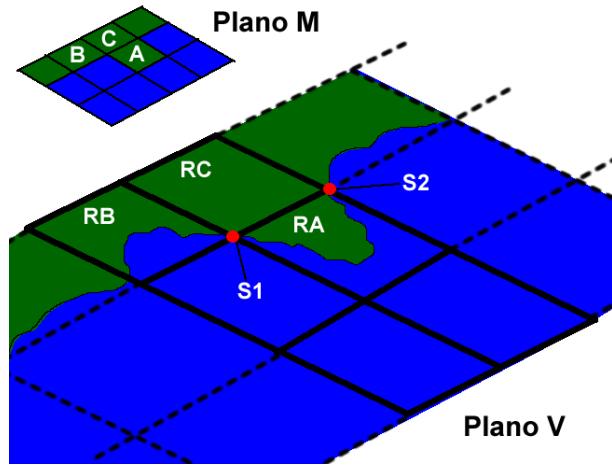


Figura 3.12: Funcionamento do algorítmo de quebra de linearidade da costa

processadas e essas informações são armazenadas na MM, existem apenas informações sobre terra (continente) e água. Esse é o resultado criado pelo algorítmo de criação de mundos citado na seção 3.3.2.

A partir desse momento, o primeiro passo do algorítmo 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 algorítmo 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 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 desse mapeamento, o que irá resultar em uma costa não-retilínea e mais realista. A figura 3.12 ilustra o funcionamento do algoritmo.

Os pixels A e B da MM possuem um descritor de informação indicando que eles são costa. Alguns outros pixels da figura também são costa, porém eles não serão detalhados por fins de clareza. O Plano M descreve a MM e o Plano V descreve o resultado do

mapeamento dela no mundo virtual. Embora não esteja descrito na ilustração, 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. 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). 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,

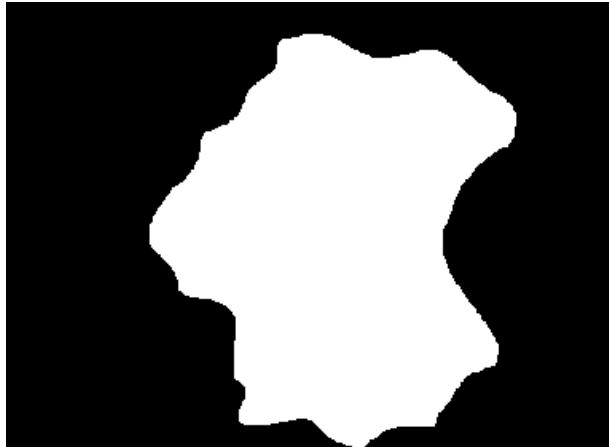


Figura 3.13: Representação gráfica do espectro de ruído utilizado para criação da costa

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. 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; 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. 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 3.13 ilustra o desenho do *heightmap* fictício gerado pela função de quebra de linearidade no bloco RA.

3.3.5 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 na-

tureza. 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 terminadas 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. 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



Figura 3.14: Praia gerada ao longo da costa

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 que inicia alta no continente e, à medida que se aproxima da água, fica mais baixa.

A figura 3.14 ilustra o resultado obtido com a geração de prais. 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 todas essas iterações serão executadas até o fim. 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 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.

3.3.6 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 prais 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 prais e um gerador de arquipélagos.

O **diferenciador de prais** 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 3.15 ilustra os resultados obtidos.

O **gerador de arquipélagos** atua criando novas faixas de terra em terminados 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

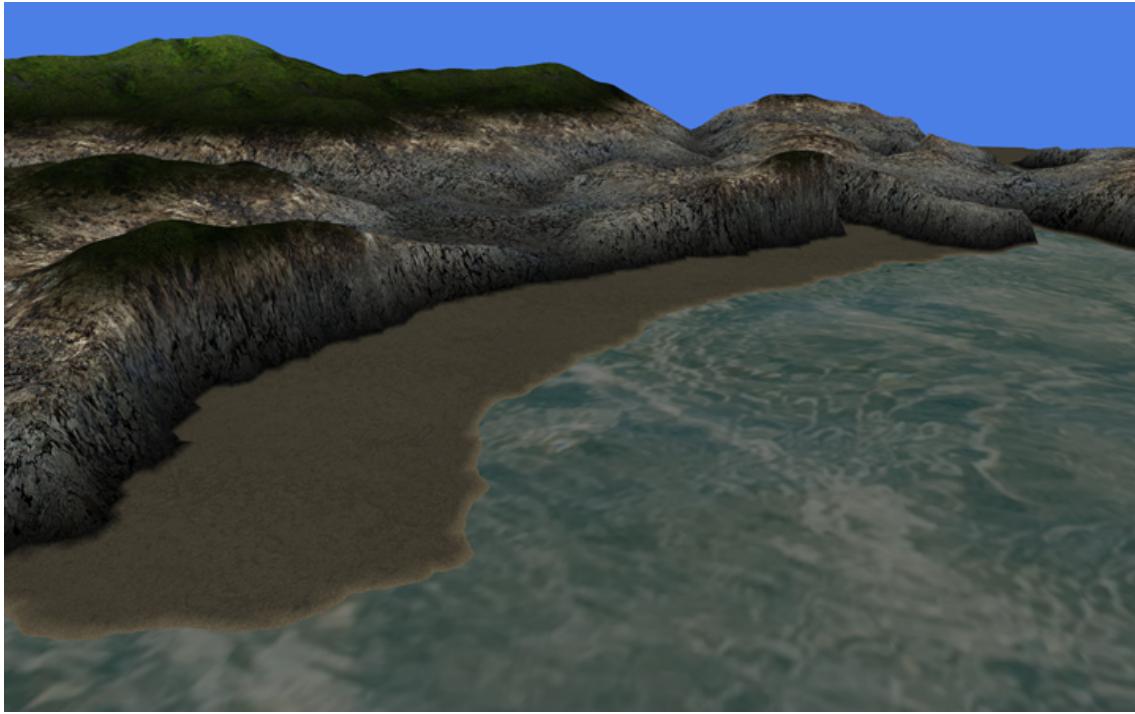


Figura 3.15: Resultado obtido com a aplicação do diferenciador de praias. (a) praia original; (b) praia que sofreu efeito do diferenciador

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 necessida 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 maciso 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 diversos tamanhos ao longo da costa em determinados locais do mundo virtual. A figura 3.16 ilustra os resultados obtidos.

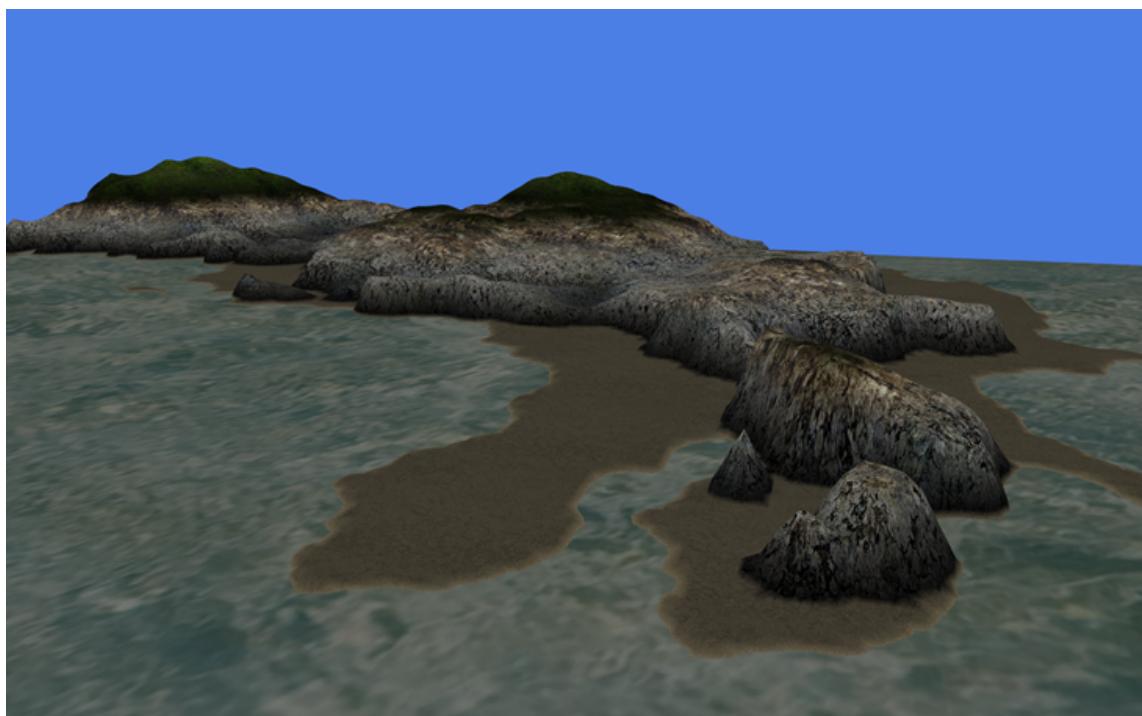


Figura 3.16: Ilha obtida com o gerador de ilhas

4 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 explanas, com imagens dos resultados e análises de desempenho.

4.1 Aspectos gerais da ferramenta

A ferramenta desenvolvida possui como característica principal a capacidade de gerar um mundo virtual pseudo-infinito que pode ser utilizado para a criação de cenários em jogos 3D. Embora a ferramenta tenha suporte para gerar um mundo infinito, sem qualquer limitação de tamanho, ela está condicionada a gerar um mundo contido dentro do tamanho máximo suportado por um inteiro sem sinal.

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.

A utilização parcial dos recursos da ferramenta é uma opção, porém, dependendo dos recursos utilizados, pode-se perder algumas funcionalidades inerentes, como é o caso da otimização da malha de terreno através de LOD. Se os métodos de renderização da ferramenta não forem utilizados para que o mundo virtual seja desenhado, por exemplo, então nenhuma redução de polígonos será feita e o programador deverá aplicar suas próprias técnicas para garantir a redução de polígonos desnecessários.

Visando um escopo de utilização genérico, concentraram-se os esforços de desenvolvimento no método de geração de relevo e nas técnicas de gerenciamento de conteúdo (acesso a um conteúdo infinito criado sob demanda). Excluindo-se as funcionalidades de otimização da malha do terreno virtual através do uso de LOD, a ferramenta não apresenta mecanismos de controle e acesso de dados que não estejam intimamente ligados a terreno. Um exemplo de mecanismo extra é a capacidade de renderização de um céu ao redor do terreno gerado; entende-se que essa tarefa é de responsabilidade da aplicação que está utilizando a ferramenta, uma vez que o propósito da mesma está focado no conteúdo do mundo virtual em si, mais propriamente em seu relevo. Outro exemplo de mecanismo extra é a iluminação da cena, cuja única tarefa que a ferramenta realiza nesse aspecto é calcular a normal de cada um dos triângulos da malha do terreno, o que permite que o programador ilumine-a e obtenha um resultado aceitável. Em relação à texturização do terreno, a ferramenta apresenta meios que permitem ao programador customizar as cores da malha, como aplicar uma função de cor em cada um dos vértices antes que esses sejam desenhados na tela. Além disso, é possível também utilizar uma das funcionalidades já embutidas na própria ferramenta para a aplicação de cores no terreno, como a texturização através da utilização de funções de ruído de Perlin.

4.2 Análise de desempenho

(A análise de desempenho ainda está em desenvolvimento)

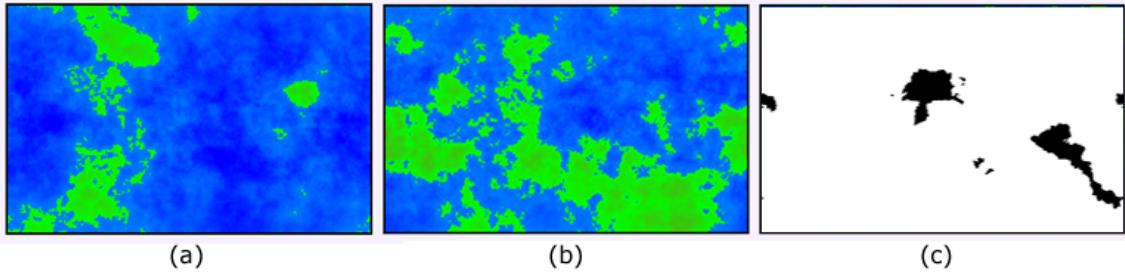


Figura 4.1: Continente gerado pela ferramenta

4.3 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, o desempenho obtido 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.

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.

4.3.1 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. A figura 4.1 ilustra um dos continentes gerados pela ferramenta.

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 algoritmo 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 4.1, analisando-se os continentes em uma linha horizontal, pode-se perceber que as faixas de terra casam 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 continentes 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 ciclicamente 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 [19] 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 perda considerável no conceito de tempo-real proposto pela ferramenta, porém, mesmo assim, os benefícios associados aos continentes gerados (como continuidade de conteúdo) compensam.

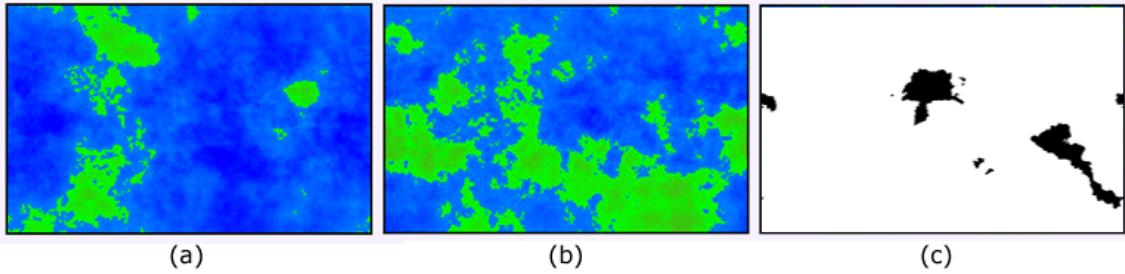


Figura 4.2: Continentes e oceanos gerados pela ferramenta

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 800×600 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 para o programador, que pode encontrar um ponto de equilíbrio entre tempo de geração dos continentes e linearidade da costa. A figura 4.2 mostra os resultados obtidos em relação a continentes e oceanos gerados pela ferramenta.

4.3.2 Avaliação do relevo

O relevo gerado pela ferramenta é inteiramente parametrizável e customizável pelo programador. Seguindo o protótipo da função esperada pela ferramenta para o cálculo do relevo, o programador pode criar qualquer código para a geração de relevo, o que garante flexibilidade nesse aspecto. A ferramenta possui embutida um gerador de relevo baseado na função de ruído de Perlin, que produz uma paisagem semelhante àquela encontrada no mundo real. A figura 4.3 ilustra o relevo criado pela ferramenta utilizando-se as funções de geração embutidas.

Em relação à malha de relevo gerada, existem dois grandes gargalos que reduzem o desempenho da aplicação: o método de geração de relevo e a quantidade de polígonos a serem mostrados. Ambos os problemas estão intimamente ligados, visto que quanto maior a malha de relevo que será mostrada na tela, maior será o processamento para a sua

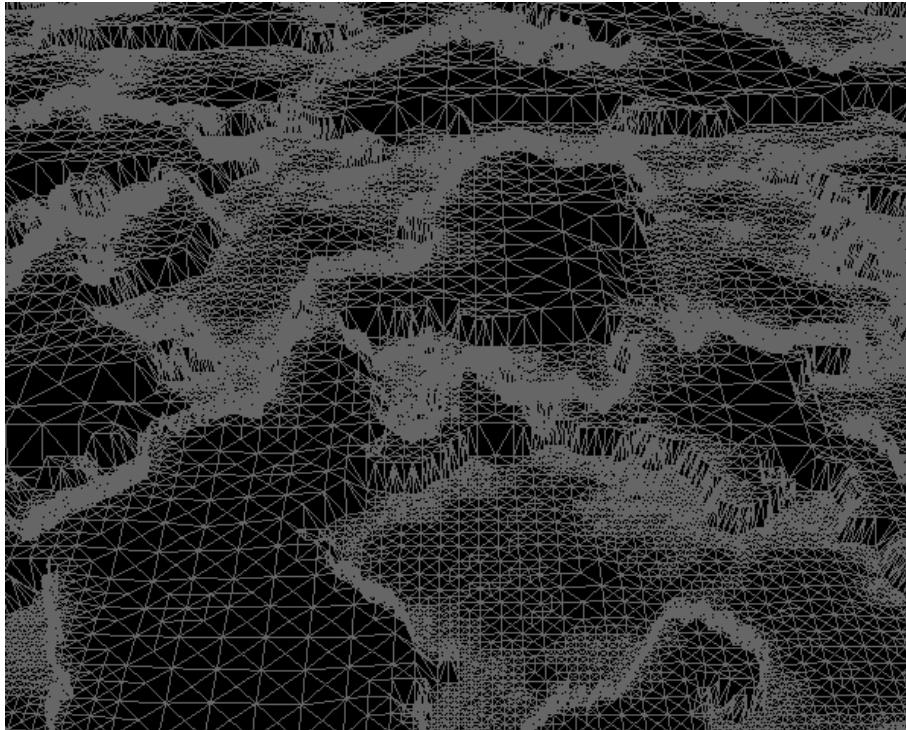


Figura 4.3: Relevo gerado pela ferramenta (*wireframe*)

renderização e geração. Em relação ao gargalo no número de polígonos, esse problema foi amenizado através de otimizações por LOD; sem a remoção dos polígonos pouco relevantes à visão do usuário, o desempenho da aplicação caia drasticamente com o aumento do campo de visão, independente do tipo de relevo que estava sendo visualizado (mais planícies, montanhas, etc.). Quando maior o campo de visão, menor é a impressão que o jogador tem de que está vendo uma fatia do mundo virtual e maior é a ideia de que o mundo é realmente infinito. Com a aplicação de LOD, houve um aumento de até 80% no desempenho da renderização das cenas, especialmente naquelas que apresentam planícies mais distantes da câmera. A tabela 4.1 mostra um comparativo de desempenho entre o funcionamento da aplicação com diferentes tamanhos de mapa.

Tabela 4.1: Desempenho da ferramenta com diferentes tamanhos de mapa

Tamanho (pontos)	Triângulos (mil)	Consumo LOD (Mb)	FPS
512	31	17	75
1024	43	67	60
2048	180	277	30
4096	877	1109	7

A primeira coluna *tamanho* mostrada na tabela é a largura (e altura) da fatia que a ferramenta irá coletar do mundo virtual. Se o tamanho for de 256, por exemplo, então

a ferramenta irá alocar uma matriz de tamanho 256×256 e irá coletar a informação de 65536 pontos do mundo virtual. A segunda coluna mostra o número de triângulos que são desenhados na tela. A terceira coluna mostra quanto de memória o algorítimo de LOD consome para poder aplicar as a redução de polígonos (a memória é utilizada na alocação da quad-tree). A quarta e última coluna mostra os quadros por segundo obtidos com a renderização da cena.

4.3.3 Avaliação da costa

5 CONCLUSÃO E TRABALHOS FUTUROS

–Trabalhos futuros: – Melhoramento nas funções que geram relevo; – Texturização conforme altura do terreno e afins; – Rios; – GPU;

REFERÊNCIAS

- [1] Michael F. Barnsley. *Fractals everywhere*, 1993.
- [2] BLIZZARD ENTERTAINMENT. World of warcraft, 2007. Disponível em: <<http://www.blizzard.com>>.
- [3] Neils L. Clark. Addiction and the structural characteristics of massively multiplayer online games. Master's thesis, University of Hawai, Hawai, 2006.
- [4] M. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *Siggraph 03 Conference proceedings*, 2003.
- [5] Steven C. Dollins. *Modeling for the Plausible Emulation of Large Worlds*. PhD thesis, Brown University, Estados Unidos da América, 2002.
- [6] 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.
- [7] Hugo Elias. Perlin noise, 2009. Disponível em: <<http://freespace.virgin.net/hugo.elias/models/mperlin.htm>>.
- [8] Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Realtime procedural generation of 'pseudo infinite' cities, 2005.
- [9] Mark D. Griffiths, Mark N.O. Davies, and Darren Chappell. Breaking the stereotype the case of online gaming, 2003.
- [10] Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proceedings of Graphics Interface 94*, 1994.

- [11] Hans Häggström. Real-time generation and rendering of realistic landscapes. Master's thesis, University of Helsinki, Finlândia, 2006.
- [12] Hans Häggström. Skycastle - free multiplayer game engine focusing on player creativity and world simulation, 2009. Disponível em: <<http://www.skycastle.org/>>.
- [13] Jason Weber and Joseph Penn. Creation and rendering of realistic trees, 1995.
- [14] George Kelly and Hugh McCabe. A survey of procedural techniques for city generation, 2004.
- [15] Sylvain Lefebvre and Fabrice Neyret. Pattern based procedural textures. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, 2003.
- [16] J. P. Lewis. Generalized stochastic subdivision. In *ACM Transactions on Graphics*, 1987.
- [17] Ondrej Linda. Generation of planetary models by means of fractal algorithms. Technical report, Czech Technical University, 2007.
- [18] Bernd Lintemann and Oliver Deussen. A modelling method and user interface for creating plants, 1998. Computer Graphics Forum.
- [19] Torben Æ. Mogensen. Instant planet generator, 2009. Disponível em: <<http://www.eldritch.org/erskin/roleplaying/planet.php>>.
- [20] 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.
- [21] Jacob Olsen. Realtime synthesis of eroded fractal terrain for use in computer games, 2004.
- [22] Cesar Tadeu Pozzer, Marcelo Dreux, and Bruno Feijó. A real-time single-pass continuous lod algorithm for regular height maps, 2004.
- [23] Prusinkiewicz Przemyslaw and Aristid Lindenmayer. The algorithmic beauty of plants, 1990. Springer-Verlag.

- [24] SONY ENTERTAINMENT. Everquest, 2007. Disponível em:
<http://everquest2.station.sony.com/>.
- [25] 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).
- [26] Thomas Wang. Integer hash function, 2000. Available at:
<http://www.concentric.net/Ttwang/tech/inthash.htm>.