

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Guilherme Oliveira Mussa Tavares

**GERAÇÃO PROCEDURAL DE TERRENOS COM SIMULAÇÃO
DE PROCESSOS EROSIVOS EM MÚLTIPLAS CAMADAS**

Campos dos Goytacazes
2024

Guilherme Oliveira Mussa Tavares

GERAÇÃO PROCEDURAL DE TERRENOS COM SIMULAÇÃO DE PROCESSOS EROSIVOS EM MÚLTIPLAS CAMADAS

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof Dr. Luis Antonio Rivera Escriba.

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Orientador: Prof. Dr. Luis Antonio Rivera Escriba

Campos dos Goytacazes

2024

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Cientista da Computação” e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação.

Campos dos Goytacazes, RJ, 28 de junho de 2024.

Banca Examinadora:

Prof. Dr. Luis Antonio Rivera Escriba
Orientador

Prof. Dr. Luis Alberto Rabanal Ramirez
Membro da Banca

Prof. Dr. João Luiz de Almeida Filho
Membro da Banca

Agradecimentos

A Deus, por me guiar pelo caminho certo e me ajudar a superar os desafios.

Ao meu irmão Bernardo e aos meus pais, Carla e Celso, por tanto amor, cuidado e incentivo em todas as etapas da minha vida. À minha família, especialmente às minhas avós, pelo imenso carinho que sempre tiveram comigo.

Ao meu orientador Rivera, pelos conselhos antes e durante a escrita desta monografia, bem como pelos conhecimentos transmitidos nas aulas.

Além dos aprendizados que adquiri de cada um, agradeço a Annabell pelo apoio quando eu mais precisava; ao Tang pelo excelente trabalho como coordenador do curso; e ao Ausberto, por me fazer apreciar a Engenharia de Software.

Aos colegas que pude conhecer e com quem compartilhei bons momentos, em especial, meus amigos Yuri, Maria, Luis e Javier. Foi sensacional contar com vocês durante essa trajetória!

Por fim, minha gratidão ao corpo docente, técnico e administrativo da UENF. Sem o empenho de todos, nada disso seria possível. Obrigado!

Resumo

A capacidade de simular terrenos naturais favorece diferentes aplicações, como jogos eletrônicos, simulações ambientais e estudos científicos. Um dos procedimentos mais adotados para a criação de terrenos se refere à geração procedural, ou seja, de forma automática através de algoritmos.

Neste trabalho, pretendeu-se investigar técnicas de geração de terrenos e simulação de erosão com base em modelos de múltiplas camadas de solo, um campo ainda pouco explorado na literatura. Em vista disso, foram implementados algoritmos de geração de terrenos utilizando ruído de Perlin com método fractal, que possibilitaram a composição de camadas. Além disso, desenvolveu-se um algoritmo de erosão hidráulica que simula a ação de gotas de água na remoção, transporte e deposição de sedimentos.

Os resultados mostram que a abordagem proposta gera terrenos visualmente satisfatórios e permite a simulação eficaz dos processos de erosão. A análise dos tempos de execução do algoritmo confirmou uma correlação próxima com a complexidade teórica. Conclui-se que a utilização de modelos de múltiplas camadas é viável para a simulação de processos geomorfológicos, sugerindo que futuros aprimoramentos podem focar na modelagem mais precisa da infiltração e saturação de água no solo e em otimizações para execução em ambientes de computação paralela.

Palavras-chaves: Geração de terrenos, Geração procedural, Simulação de erosão, Representação em camadas

Abstract

The ability to simulate natural terrains benefits various applications, such as video games, environmental simulations, and scientific studies. One of the most widely adopted procedures for creating terrains is procedural generation, which is automatic and algorithm-based.

This study aimed to investigate techniques for terrain generation and erosion simulation based on multi-layered soil models, an area still underexplored in the literature. To this end, terrain generation algorithms using Perlin noise with a fractal method were implemented, allowing for the composition of layers. Additionally, a hydraulic erosion algorithm was developed to simulate the action of water droplets in the removal, transport, and deposition of sediments.

The results show that the proposed approach generates visually satisfactory terrains and effectively simulates erosion processes. The analysis of the algorithm's execution times confirmed a close correlation with theoretical complexity. It is concluded that the use of multi-layered models is viable for simulating geomorphological processes, suggesting that future improvements could focus on more accurate modeling of soil water infiltration and saturation, as well as optimizations for execution in parallel computing environments.

Key-words: Terrain generation, Procedural generation, Erosion simulation, Layered representation

Listas de ilustrações

Figura 2.1 – Dois modos de representar dados de terreno: mapa de altura (à esquerda) e <i>voxels</i> (à direita)	17
Figura 2.2 – Exemplo de mapa de altura	18
Figura 2.3 – Elevação do terreno a partir de um mapa de altura.	18
Figura 2.4 – Representação esquemática de modelo em camadas	19
Figura 2.5 – Terreno fractal de neve	21
Figura 2.6 – Resultado final do algoritmo Perlin Noise	22
Figura 2.7 – Soma de duas oitavas de ruído	23
Figura 2.8 – Representação do processo erosivo	24
Figura 2.9 – Exemplo da iteração inicial do algoritmo de erosão térmica	25
Figura 2.10 – Representação do processo erosivo euleriano e lagrangiano.	27
Figura 2.11 – Exemplo de estrutura geológica em camadas.	28
Figura 2.12 – bordagem híbrida do framework <i>Arches</i>	29
Figura 2.13 – Estabilização simulada de camadas do terreno	29
Figura 2.14 – Estrutura de dados de terreno para fluxos virtuais de água.	31
Figura 2.15 – Representação de terreno em camadas com encadeamento de seções.	32
Figura 3.1 – Etapas do modelo de geração proposto.	34
Figura 3.2 – Modelo representativo de terreno em camadas.	35
Figura 3.3 – Estruturas de dados agregadas ao modelo em camadas.	36
Figura 3.4 – Organização interna do mapa de camada.	37
Figura 3.5 – Fluxograma do algoritmo de composição de camadas do terreno.	38
Figura 3.6 – Exemplo de preenchimento e interação de duas camadas.	39
Figura 3.7 – Diagrama do ciclo de vida da gota d’água na simulação erosiva.	40
Figura 3.8 – Posicionamento ilustrativo dos nós da célula no mapa.	42
Figura 4.1 – Módulos do sistema desenvolvido e sua interação com o motor gráfico. .	48
Figura 4.2 – Triangulação da grade de terreno	49
Figura 4.3 – Diagrama de classes do módulo de geração.	51
Figura 4.4 – Diagrama de sequência do processo integral da geração de terrenos. . .	52
Figura 4.5 – Diagrama de atividades do algoritmo de erosão hidráulica.	54
Figura 4.6 – Sequência evolutiva da formação de terreno em camadas.	60
Figura 5.1 – Influência do tamanho da grade; terreno (a) com tamanho de 128x128, (b) 256x256 e (c) 512x512.	67
Figura 5.2 – Exemplo de terreno gerado com 4 camadas.	69
Figura 5.3 – Gráfico com médias de tempo de execução para combinações entre diferentes tamanhos de grade e quantidade de camadas do terreno.	70

Figura 5.4 – Gráfico de dispersão e regressão polinomial de grau 2 dos valores de tempos de execução referentes aos tamanhos da grade.	71
Figura 5.5 – Gráfico de dispersão e regressão linear dos valores de tempo de execução referentes à quantidade de camadas.	72
Figura 5.6 – Terreno gerado anteriormente ao processo de erosão.	73
Figura 5.7 – Sucessão de impactos erosivos segundo a duração de simulação.	74
Figura 5.8 – Gráfico de dispersão e regressão linear dos valores de tempo de execução relativos à duração de simulação.	76
Figura 5.9 – Gráfico de dispersão e regressão linear dos valores de tempo de execução relativos ao tamanho da grade.	77
Figura 5.10–Exemplo de terreno com 256x256 e 4 camadas: (a) terreno original; (b) simulação erosiva de 100 mil gotas; (c) simulação erosiva de 200 mil gotas.	79
Figura 5.11–Exemplo de terreno com 364x364 e 4 camadas: (a) terreno original; (b) simulação erosiva de 120 mil gotas; (c) simulação erosiva de 240 mil gotas.	80
Figura 5.12–Exemplo de terreno com 512x512 e 4 camadas: (a) terreno original; (b) simulação erosiva de 150 mil gotas; (c) simulação erosiva de 300 mil gotas.	81

Lista de tabelas

Tabela 2.1 – Comparaço entre mapa de altura, grade de <i>voxel</i> e multicamada.	20
Tabela 5.1 – Tipos de solo definidos para os cenarios de teste	68
Tabela 5.2 – Medias do tempo de execuao (em milissegundos) do algoritmo de eroso correspondentes as combinaoes entre tamanho da grade e duraao de simulaao.	75

Lista de códigos

Código 4.1	Método de busca de seção de solo em uma posição (x, y)	56
Código 4.2	Método de adição de uma seção de solo a uma posição do mapa.	56
Código 4.3	Método de remoção da seção de solo de uma pilha.	57
Código 4.4	Lógica de repetição para gerar valores de ruído de uma oitava.	58
Código 4.5	Algoritmo de composição de camadas.	59
Código 4.6	Cálculo de pesos de erosão para cada posição do mapa.	61
Código 4.7	Inicialização do processo de erosão hidráulica.	61
Código 4.8	Ciclo de vida e movimentação da partícula.	62
Código 4.9	Determinação da inclinação e da capacidade máxima de sedimentos.	63
Código 4.10	Algoritmo de deposição de sedimentos.	64
Código 4.11	Algoritmo do processo de erosão.	65

Sumário

1	INTRODUÇÃO	11
1.1	Problemática	12
1.2	Hipótese	13
1.3	Objetivos	13
1.4	Justificativa	13
1.5	Método	14
1.6	Organização do Trabalho	15
2	GERAÇÃO PROCEDURAL DE TERRENOS	16
2.1	Representação de Terreno	17
2.2	Técnicas Fractais na Geração de Terreno	21
2.3	Simulação de Processos Erosivos	24
2.3.1	Erosão Térmica	25
2.3.2	Erosão Hidráulica	26
2.4	Trabalhos Relacionados	27
3	MODELAGEM CONCEITUAL PARA SIMULAÇÃO DE TERRENOS	33
3.1	Estrutura e Dinâmica das Camadas de Terreno	35
3.1.1	Mapa de Camada	35
3.1.2	Composição de Camadas	37
3.2	Simulação de Erosão Hidráulica	39
3.2.1	Dinâmica da Partícula	41
3.2.2	Erosão e Deposição	45
4	IMPLEMENTAÇÃO DO SISTEMA DE GERAÇÃO DE TERRENOS	48
4.1	Módulo de Geração	50
4.2	Módulo de Erosão	52
4.3	Desenvolvimento da Aplicação	55
4.3.1	Estruturas de Dados	55
4.4	Ruído e Geração de Camadas	57
4.4.1	Algoritmo de Erosão Hidráulica	60
5	RESULTADOS E DISCUSSÃO	66
6	CONSIDERAÇÕES FINAIS	83
	REFERÊNCIAS	84

1 Introdução

Com uma vasta aplicação em videogames, filmes e simulações, os terrenos virtuais desempenham relevante papel na criação de ambientes imersivos e visualmente atraentes. A sua elaboração envolve técnicas modernas de modelagem e texturização, necessárias para reproduzir com precisão as complexidades e os atributos das paisagens naturais e artificiais. Conforme declaram [Dey, Doig e Gatzidis \(2018\)](#), uma quantidade expressiva de pesquisas sobre o tema vem se utilizando de métodos totalmente automatizados ou semi-automatizados.

Uma solução para criação de terrenos é o conceito de geração procedural de conteúdo (GPC) como uma criação algorítmica de conteúdo de jogo com reduzido nível de interferência humana ([SHAKER; TOGELIUS; NELSON, 2016](#)). Sua aplicação tem se voltado preferencialmente para jogos digitais ao produzir múltiplos cenários, níveis, mapas, missões, texturas, personagens e outros elementos. Neste século, a GPC tem sido usada para gerar conteúdo em diferentes games como *Minecraft*, *Spelunky*, *Terraria*, *No Man's Sky* e outros. Desde o lançamento do *Minecraft* e do *Spelunky* em 2009, cresceu o interesse industrial e comercial pelas técnicas de geração de conteúdo voltadas para jogos.

Mecanismos procedurais não dispensam, no entanto, a contribuição humana: designers e artistas participam deste processo de criação, que é bastante dinâmico. Estes profissionais podem modelar a base de terrenos, que será modificada pelo método procedural para aperfeiçoar os detalhes ou, ao contrário, a base do terreno é gerada proceduralmente, cabendo ao artista transformá-la. Percebe-se, assim, que uma das vantagens na geração procedural de terrenos é a criação dinâmica de conteúdos, gerando superfícies dotadas de variados recursos. Além disso, a geração procedural oferece vantagens em termos de produtividade, ao automatizar o processo de criação de terrenos, reduzindo o esforço manual ([AZAD et al., 2016](#)).

Vale ressaltar que o adjetivo procedural não sugere a ideia de uma absoluta aleatoriedade, já que o uso de valores aleatórios se baseia em certos parâmetros, o que torna os resultados relativamente previsíveis ([RUBIO; RÍO; ARRIBAS, 2020](#)) Quanto ao vocábulo conteúdo, deve-se considerar que o mesmo corresponde aos objetos físicos do jogo e suas particularidades, tais como masmorras (*dungeons*) e até mesmo as suas regras.

Um desafio significativo na literatura reside na determinação de métodos, parâmetros e representações condizentes para gerar terrenos proceduralmente diversos. No entanto, existe uma necessidade crítica de estabelecer uma compreensão abrangente de como diferentes técnicas procedurais impactam sobre as características dos terrenos resultantes, tais como realismo, detalhe e eficiência computacional. Este desafio é acentuado pela

variedade de métodos disponíveis e pelas complexas relações entre estas técnicas e os seus parâmetros correspondentes. Além disso, como sustentam Valencia-Rosado e Starostenko (2019), percebe-se uma relativa dificuldade em avaliar os resultados qualitativos do terreno em virtude de não existir uma padronização de métricas.

Na geração de um terreno, existem algumas possibilidades de representá-lo além do aspecto visual, como, por exemplo, o modo como os dados são estruturados. Benes e Forsbach (2001) observam que as representações mais comuns são aquelas bidimensionais e tridimensionais, como mapa de altura e grade de *voxel*, respectivamente. Estes autores propuseram um novo arranjo de dados, especificamente terrenos em múltiplas camadas. O intuito principal deste formato foi favorecer a simulação de processos erosivos, uma vez que cada camada do terreno possui materiais com propriedades físicas distintas. Isto possibilita a modelagem de fenômenos naturais complexos, como o desgaste de rochas, o acúmulo de sedimentos e a formação de ravinhas ao longo do tempo.

A erosão é um processo que ocorre na natureza pelo qual materiais como areia e rochas são retirados de seu local original e depois transportados e depositados em outro local por meio da água, vento ou outras forças naturais. Os métodos utilizados na modelagem de terreno baseada na erosão têm se diversificado bastante nas últimas décadas. Musgrave, Kolb e Mace (1989) realizaram as primeiras tentativas de gerar terrenos proceduralmente através da geometria fractal, a fim de simular os processos de erosão que os modificam. Os processos de erosão podem ser subdivididos em três categorias principais: térmica, hidráulica e eólica. Estes tipos são, do ponto de vista geomorfológico, afetados por uma série de variáveis que podem influenciar a formação e manipulação do relevo ao longo do tempo, tais como os níveis de temperatura, a frequência e intensidade das precipitações, além da composição do solo, cobertura vegetal e topografia. Diante da complexidade inerente à criação de modelos computacionais de erosão, muitos dos modelos existentes tendem a priorizar a simulação de apenas uma ou duas destas variáveis.

1.1 Problemática

Um dos principais problemas abordados neste trabalho científico é a investigação de métodos de erosão aplicados em terrenos procedurais gerados em múltiplas camadas, uma área que ainda é escassa na literatura existente. A maioria dos estudos sobre geração de terrenos focam em abordagens baseadas em fractais para a criação de terrenos superficiais, sem considerar a complexidade adicional introduzida pela representação em camadas.

A aplicação de algoritmos de erosão ao modelo multicamada demanda uma compreensão profunda de como diferentes tipos de solo interagem sob condições variadas de saturação e fluxo de água, além de como tais interações influenciam a topografia resultante. Este trabalho busca preencher modestamente esta lacuna, propondo um modelo que integra

estas variáveis de forma a produzir resultados que refletem com maior precisão os processos naturais de erosão e deposição.

1.2 Hipótese

A geração em multicamada mostra-se capaz de aperfeiçoar potencialmente a adaptabilidade do terreno para simulações geomorfológicas, mesclando as vantagens da representação em mapa de altura e *voxels*. Supõe-se que este modelo proporciona estabilidade performática e expressividade visual, desde que sejam aplicadas estratégias eficazes de otimização.

1.3 Objetivos

O presente trabalho tem como objetivo geral desenvolver uma aplicação de geração de terrenos baseada na modelagem em multicamada e simulação erosiva. A partir deste modelo, fundamentado na contribuição de Benes e Forsbach (2001), a pesquisa em foco pretende encontrar um equilíbrio entre desempenho computacional e fidelidade visual, contribuindo, em última análise, para a exposição de cenários virtuais em vários domínios de aplicação.

O objetivo principal se desdobra nos seguintes objetivos específicos:

- Propor um método consistente voltado para a geração procedural de terrenos dispostos em multicamada;
- Implementar e adaptar algoritmos de interpolação fractal para geração de múltiplas camadas do terreno;
- Aplicar técnicas de simulação erosiva com o intuito de agregar realismo ao terreno.

1.4 Justificativa

A escolha de técnicas fractais é fundamentada em sua capacidade intrínseca de capturar a complexidade e a variação natural encontradas em ambientes geográficos reais. Os fractais permitem a criação de detalhes iterativos e autossemelhantes em diferentes escalas (ARCHER, 2011), o que se alinha com a necessidade de representar terrenos em múltiplas camadas. Além disso, a natureza estocástica das técnicas fractais permite a introdução de variabilidade realista nos terrenos gerados, o que é essencial para cenários virtuais que buscam autenticidade e diversidade topográfica.

Da mesma forma, a adoção de métodos de simulação para a geração de terrenos multicamada é justificada pela variabilidade de materiais e atributos geomorfológicos

e geológicos que moldam a superfície terrestre. Modelos de terrenos em multicamada oferecem a flexibilidade necessária para incorporar fenômenos como erosão, permitindo uma representação mais dinâmica. Isto é particularmente relevante quando se busca um equilíbrio entre a visualização fidedigna do terreno e a otimização de desempenho, já que as técnicas de simulação podem ser ajustadas para atender a requisitos específicos de aplicação.

1.5 Método

Este trabalho sintetiza, em caráter preliminar, o estado da arte referente à geração procedural de terrenos a partir de produções textuais como: livros, artigos científicos, sites e monografias.

O levantamento eletrônico da produção acadêmica sobre o tema desta monografia realizou-se por meio dos mecanismos de busca de quatro bibliotecas digitais: Google Acadêmico, IEEE Xplore, ACM e Scopus. A cadeia de busca considerada foi: (*“procedural terrain generation” OR “terrain generation” OR (“terrain generation” AND (layered OR “stack-based” OR volumetric OR heightmap OR “height-map”)) OR (“terrain generation” AND fractal) OR (terrain AND (erosion OR “erosion simulation” OR “hydraulic erosion”))*).

Neste trabalho, o algoritmo de geração de terrenos foi implementado com base no ruído Perlin (PERLIN, 1985) a fim de realizar a composição em camadas. O algoritmo gera uma altura para cada célula do terreno, levando em consideração as propriedades dos diferentes tipos de solo definidos na simulação.

Por sua vez, o algoritmo de erosão aqui empregado simula o impacto de partículas de água no terreno, movimentando-as de acordo com o gradiente local e ajustando a quantidade de sedimentos transportados. A presente pesquisa incluiu ainda a absorção de água pelo solo e a remoção de camadas superiores quando necessário.

A aplicação de geração de terrenos multicamada para simulação erosiva - objetivo deste trabalho - foi implementada a partir do motor de desenvolvimento de jogos *Unity* e da linguagem C#. Este motor oferece uma ferramenta apropriada para a coleta de dados de tempo de execução. Com base nestes dados, realizou-se uma plotagem de gráficos de regressão por meio da ferramenta *LibreOffice Calc* para avaliar a performance dos algoritmos e sua complexidade. Além disso, foram geradas representações visuais dos terrenos antes e depois da erosão para validar visualmente os efeitos dos algoritmos implementados.

1.6 Organização do Trabalho

Além desta introdução, o presente trabalho está organizado em outros cinco capítulos que abordam de forma estruturada os conceitos, metodologias, implementações e resultados obtidos na pesquisa sobre geração de terrenos procedurais com simulação de processos erosivos. A seguir, é apresentada uma visão geral dos capítulos a seguir:

- **Capítulo 2:** Neste capítulo, é realizada uma revisão da literatura que explora os principais conceitos e técnicas relacionados à geração procedural de terrenos e simulação de erosão. São discutidos trabalhos anteriores, com foco em métodos baseados em ruído e fractal, algoritmos de erosão, e representações de terreno em múltiplas camadas. Esta revisão fornece a base teórica necessária para o desenvolvimento do trabalho.
- **Capítulo 3:** Este capítulo apresenta a modelagem conceitual do sistema, incluindo as estruturas de dados e algoritmos utilizados para a geração de terrenos e simulação de erosão. Uma definição formal é elaborada a respeito dos algoritmos implementados.
- **Capítulo 4:** No quarto capítulo, são discutidos os aspectos práticos da implementação do sistema. Detalham-se os passos de desenvolvimento, desde a definição das estruturas de dados e os parâmetros envolvidos até a implementação dos algoritmos de geração e erosão. São apresentados trechos de código e explicações das principais funções presentes nos algoritmos.
- **Capítulo 5:** Este capítulo é dedicado à apresentação e análise dos resultados obtidos com a aplicação dos algoritmos de geração e erosão. São exibidos gráficos e figuras ilustrando os terrenos gerados e as transformações resultantes dos processos erosivos. A análise dos resultados é realizada com base em métricas como o tempo de execução, e variáveis como o tamanho da grade do terreno, relacionando os dados obtidos com a complexidade teórica dos algoritmos.
- **Capítulo 6:** O último capítulo apresenta as conclusões do trabalho, resumindo os principais procedimentos utilizados ao longo desta pesquisa. São apresentadas as limitações deste trabalho e formuladas sugestões para trabalhos futuros que possam aprimorar os métodos desenvolvidos e explorar novas áreas de aplicação.

2 Geração Procedural de Terrenos

A geração procedural de terrenos consiste na criação de superfícies por meio de algoritmos. Os terrenos gerados proceduralmente podem abranger altura, texturas, construções, elementos da natureza como água e vegetação. Diante desta variedade de componentes da paisagem em um mundo virtual, muitas vezes a modelagem manual revela-se onerosa e repetitiva se comparada com as técnicas procedurais de construção de cenários.

Inicialmente, a geração algorítmica de terrenos limitava-se à elevação de suas superfícies e também ao crescimento de vegetação, expandindo sua aplicabilidade para ambientes urbanos ([SMELIK et al., 2009](#)). A variedade de terrenos e texturas pode ser calculada a partir de metodologias fractais. Fractal consiste em “um objeto geometricamente complexo, cuja complexidade surge através da repetição de forma em algum intervalo de escala” ([EBERT, 2003](#)).

[Valencia-Rosado e Starostenko \(2019\)](#) discorreram sobre as diferentes técnicas empregadas na geração procedural de terrenos, sejam elas simulações ou objetos de entretenimento, como vales, rios, montanhas, nos quais se introduz certo grau de realismo e controle do usuário. Abordagens diversas têm sugerido uma reprodução realista dos padrões da natureza, seja por meio da imitação de processos geológicos, da réplica de paisagens ou das interações do usuário para moldar os terrenos. Conforme demonstra os autores supracitados, os métodos de geração compreendem quatro famílias, a saber:

- **métodos estocásticos:** Utilizam regras e parâmetros para “imitar” a aleatoriedade do mundo real. São os mais antigos e muito rápidos, porém com recursos gerados em posições aleatórias. Inclui as técnicas fractais (estes são objetos geométricos em que cada uma de suas partes se assemelha ao todo), técnicas baseadas em gramática, técnicas de revestimento e técnicas de parametrização. São empregados para gerar terrenos de base, que são modificados através de outros métodos.
- **métodos de simulação:** são aqueles que emulam eventos naturais modeladores de terrenos, como simulações geomorfológicas, simulações de ecossistemas e simulações baseadas em agentes. São mais lentos, porém mais precisos. A erosão, sobretudo a erosão hídrica, é o evento mais simulado neste método.
- **métodos de aprendizagem:** estes usam dados do mundo real, principalmente na forma de imagens e modelos digitais de elevação (DEM) e tentam imitar a aparência de terrenos reais.

- **métodos baseados em esboço:** tais métodos aperfeiçoam o controle sobre terrenos gerados, propiciando ao usuário a capacidade de inserir recursos de terreno. Pelo fato de permitirem controlar o posicionamento dos recursos do terreno, são ideais para videogames, já que em jogos onde existem corpos d'água e vegetação é comum o jogador atravessar os terrenos. O traço negativo é que o grau de realismo é menor que nos outros métodos.

Cada família possui características e aplicações distintas, sendo possível combiná-las. Por fim, é indispensável considerar os objetivos específicos de cada projeto para selecionar a abordagem mais adequada.

2.1 Representação de Terreno

Existe uma ampla pesquisa acerca das estruturas de dados para armazenar o conjunto de dados do terreno. A memória ocupada, a eficiência de renderização e a velocidade de acesso à estrutura de dados podem ser consideradas fatores relevantes na representação do terreno ([NADIG, 2022](#)).

Os modelos de representação mais comuns de terrenos são mapas de altura (do inglês, *heightmaps*) e *voxels*. Os primeiros definem a superfície do terreno com uma matriz bidimensional de valores que fornecem, em cada célula, os níveis de altitude do terreno ([KOCA, 2012](#)). Nesta matriz, cada valor representa a altura do terreno na exata posição deste valor. Uma grade de *voxels* é composta por um conjunto de elementos de volume (*voxels*) em um espaço tridimensional discreto ([WANG; LIU; JUN, 2015](#)).

A [Figura 2.1](#) mostra as duas representações de terreno referidas acima.

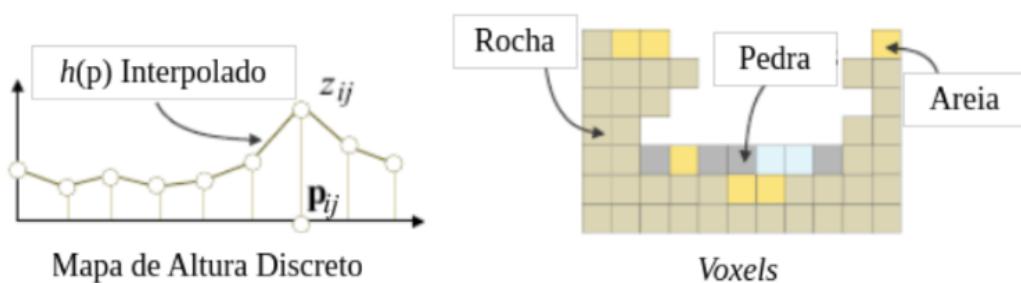


Figura 2.1 – Dois modos de representar dados de terreno: mapa de altura (à esquerda) e *voxels* (à direita). Fonte: [Galin et al. \(2019, tradução nossa\)](#)

Os mapas de altura podem ser visualizados como uma imagem em escala de cinza onde a tonalidade de cada pixel representa a altitude naquela localização (ver [Figura 2.2](#)). Tipicamente, a cor preta significa menor altitude e a branca, a maior, ao passo que as tonalidades de cinza compreendem as altitudes intermediárias.

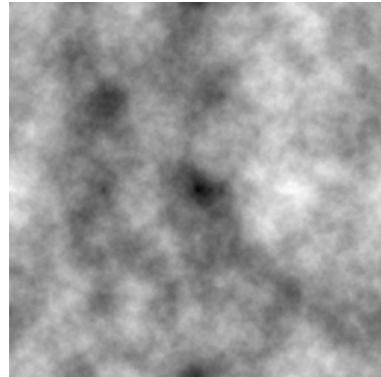


Figura 2.2 – Exemplo de mapa de altura. Fonte: [Willemse e Hawick \(2013\)](#)

A Figura 2.3 ilustra como os valores de altitude codificados no mapa de altura são usados para deslocar vértices em uma grade 3D, formando, assim, uma malha poligonal que representa um terreno. No entanto, pelo fato de a representação se limitar apenas aos dados de elevação do nível da superfície, os mapas de altura são restritos e, portanto, não podem criar estruturas complexas como cavernas ou penhascos encontrados em terrenos do mundo real ([KOCA, 2012](#)).

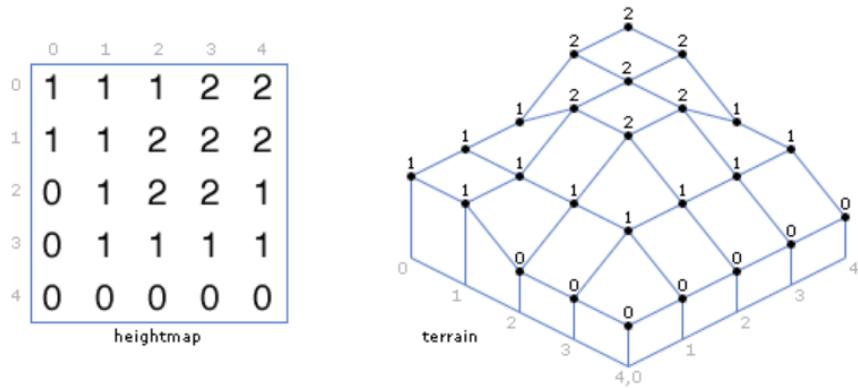


Figura 2.3 – Elevação do terreno a partir de um mapa de altura.

Fonte: [Alonso, Arinyo e Albajés \(2008\)](#)

Na técnica de discretização em uma grade de *voxels* não há estas restrições topológicas que os mapas de altura possuem, sendo amplamente utilizadas formações rochosas complexas, que incluem elementos como cavernas e saliências ([BOGGUS; CRAWFIS, 2009](#)). Entretanto, segundo [Löffler, Müller e Schumann \(2011\)](#), uma representação tridimensional excede rapidamente os recursos de memória disponíveis. Algoritmos de isosuperfície, como o Marching Cubes ([LORENSEN; CLINE, 1998](#)), são empregados para extrair superfícies contínuas de campos escalares discretos representados pelos *voxels*, otimizando a visualização de estruturas volumétricas. Estes algoritmos ajudam a mitigar o volume de dados ao

converter a representação volumétrica em malhas geométricas, que podem ser processadas e renderizadas mais eficientemente. Outras medidas de otimização incluem a utilização de estruturas de dados esparsas (LAINE; KARRAS, 2010) ou representações *voxel* em camadas (PEYTAVIE et al., 2009) (FURTADO, 2019).

Portanto, é importante considerar que, pela diferença de dimensionalidade do mapa de altura em relação à grade de *voxel*, o primeiro é mais eficiente em termos de memória, enquanto o modelo de *voxels* favorece a construção de estruturas paisagísticas mais salientes.

Devido às limitações referidas acima acerca das representações já mencionadas, foi idealizado e implementado por Benes e Forsbach (2001) um modelo em múltiplas camadas ou multicamada. Esta estrutura de dados possibilitou a simulação de erosão de uma forma mais consistente, de vez que permitiu a distinção de materiais em ordem predefinida empilhados, tais como rocha, pedra, cascalho, areia ou água, na configuração geomorfológica, como se vê na Figura 2.4.

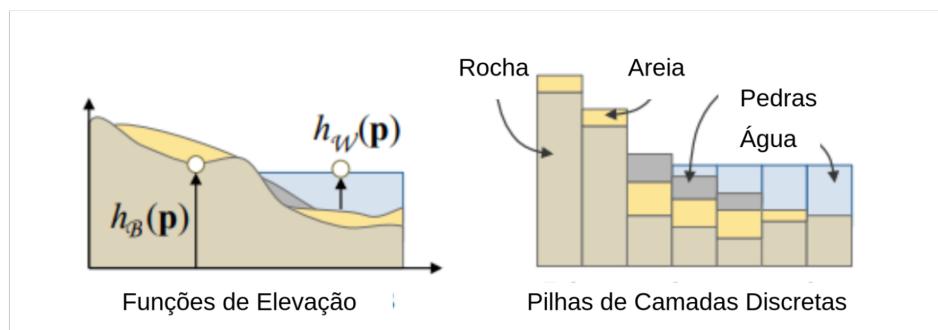


Figura 2.4 – Representação esquemática de modelo em camadas. Fonte: Galin et al. (2019)

As representações em camadas de mapas topográficos também apresentam uma vantagem distintiva ao simular características de terreno que não são facilmente replicadas por representações mais simples de mapas de elevação. Neste paradigma de representação, a altitude atribuída a cada célula é composta pelas altitudes associadas a camadas distintas de terreno, caracterizadas por diferentes materiais, tais como rochas, areia, solo ou até mesmo o ar (PINTO, 2017).

Este tipo de descrição do terreno contribui significativamente para a melhoria do cenário na aplicação de algoritmos de erosão, permitindo uma modelagem mais precisa e detalhada das mudanças topográficas ao longo do tempo.

Entretanto, é importante destacar que a maior riqueza de informação mantida por essa representação implica em uma utilização mais substancial de memória, representando assim uma desvantagem comparativa em relação ao uso dos mapas de altura. Contudo,

essa desvantagem se torna justificável quando se busca alcançar resultados que demandam mais detalhes em uma escala reduzida.

A Tabela 2.1 compara as três representações em foco a partir de uma seleção de critérios.

Aspecto	Mapa de Altura	Grade de Voxel	Multicamada
Estrutura de Dados	Grade 2D com valores de elevação	Grade 3D com dados volumétricos	Múltiplas camadas 2D empilhadas verticalmente
Flexibilidade do Terreno	Limitada às características da superfície	Eficaz para estruturas complexas, como cavernas e saliências	Pode representar camadas como solo, rocha, água, mas características complexas como cavernas requerem tratamento especial
Consumo de Memória	Relativamente baixo	Intensivo por conta da grade tridimensional	Moderadamente eficiente dependendo do número de camadas
Deformação do Terreno	Limitada, devido à natureza bidimensional	Consegue lidar com mudanças dinâmicas de <i>voxels</i>	Flexível, dependendo de como as camadas interagem e são modificadas
Desempenho de Renderização em Tempo Real	Eficiente para renderização de superfície	Mais lento devido aos dados volumétricos	Desempenho variável, podendo ser otimizado com base na interação das camadas

Tabela 2.1 – Comparaçāo entre mapa de altura, grade de *voxel* e multicamada.

A análise dos métodos de representação de terrenos revela que cada abordagem oferece vantagens e desafios específicos, adequando-se a diferentes requisitos de simulação e renderização. O mapa de altura, com seu consumo de memória relativamente baixo, é ideal para representações bidimensionais eficientes, enquanto a grade de *voxel* mostra aptidão para estruturas tridimensionais complexas, embora com maior demanda de recursos. Por outro lado, o método multicamada equilibra a flexibilidade na representação de diferentes tipos de solo com a eficiência no consumo de memória, tornando-se uma opção viável para

simulações erosivas, que requerem múltiplas camadas interativas.

2.2 Técnicas Fractais na Geração de Terreno

Através dos terrenos fractais é possível criar paisagens texturizadas, cenários da natureza como regiões montanhosas, linhas costeiras e até mesmo paisagens lunares. [Rose e Bakaoukas \(2016\)](#) destacam duas conhecidas propriedades dos fractais: são autossimilares (os objetos apresentam a mesma forma em diferentes escalas) e caóticos (comportamento recursivo, variando conforme a quantidade infinita de escalas). A própria natureza fornece exemplos do que se denomina geometria fractal, através de árvores, galhos, plantas, folhas, nuvens, cristais e outros elementos. Os algoritmos usados na geração de paisagens fractais apresentam certas limitações, já que não podem replicar transições abruptas, tais como cavernas, encostas íngremes etc.

A [Figura 2.5](#) mostra um exemplo de terreno fractal.



Figura 2.5 – Terreno fractal de neve. Fonte: [Valdivia, García e Lastra \(2020\)](#)

Além dos métodos baseados em fractais, outra abordagem eficaz na geração de terrenos envolve o uso de ruídos, como o *Perlin Noise* (ruído de Perlin) e o *Simplex Noise*. Ruídos são algoritmos matemáticos que geram valores pseudo-aleatórios a fim de simular ou modelar o ruído presente em fenômenos naturais como texturas, paisagens e animações. Tais funções são frequentemente utilizadas em gráficos por computador, jogos, computação gráfica e outras aplicações para adicionar variedade aos elementos gerados digitalmente.

Formalmente, a função de ruído é dada por $h : R^n \mapsto R$, onde a entrada corresponde a coordenadas reais n-dimensionais e a função retorna um valor real ([DEMBOGURSKI, 2009](#)).

[Ebert \(2003\)](#) expõe as propriedades fundamentais em uma função de ruído ideal. São elas:

- É uma função repetitiva pseudo-aleatória a partir de suas entradas;
- Retorna valores contidos no intervalo [-1, 1];
- Não apresenta periodicidade óbvia ou padrões regulares;
- Em relação às transformações geométricas como translação ou rotação, as funções de ruído são invariáveis.

A composição de texturas para objetos tridimensionais foi introduzida primeiramente por [Perlin \(1985\)](#). Por outro lado, o algoritmo *Perlin Noise* é capaz de implementar uma função de ruído com uma, duas, três ou mais dimensões espaciais, embora a formulação inicial do *Perlin Noise* seja usada principalmente em duas ou três dimensões. Em linhas gerais, o algoritmo *Perlin Noise* em duas dimensões requer os seguintes passos:

1. Definir uma grade de células com pontos em coordenadas inteiras
2. Atribuir um vetor gradiente pseudo-aleatório nos vértices da grade
3. Para um ponto específico P da grade, localizar a célula que contém este ponto. Determinar os vetores de deslocamento, ou seja, subtrair a posição de P com os quatro vértices da célula
4. Realizar o produto escalar entre o vetor de deslocamento e o vetor gradiente correspondente para cada vértice
5. Interpolar linearmente os produtos escalares calculados obtendo, assim, o valor para o ponto na grade

A [Figura 2.6](#) representa a grade com os valores interpolados de cada ponto.

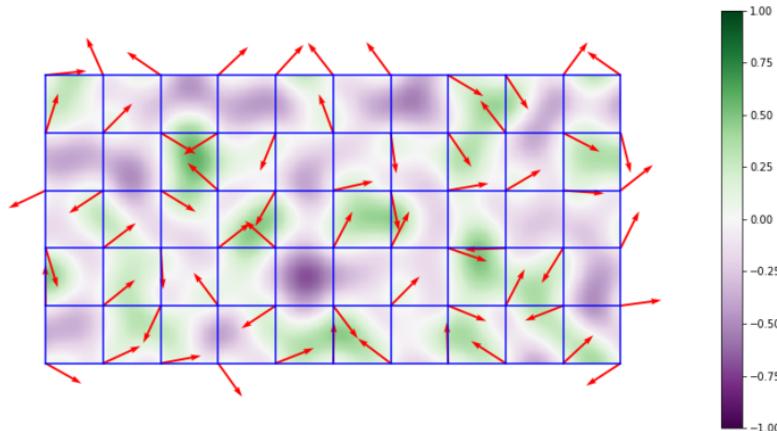


Figura 2.6 – Resultado final do algoritmo Perlin Noise. Fonte: [Perlin... \(2023\)](#)

Como esclarecem Rose e Bakaoukas (2016) muitas funções de ruído não são inherentemente fractais. Para produzir imagens fractais, os algoritmos de ruído são usados em conjunto com o modelo estocástico *Fractional Brownian Motion* (fBm). Este não é propriamente um ruído, mas uma técnica que combina diferentes camadas de ruído, que contém, cada uma delas, variações de amplitude e frequência. A isto se acrescenta algumas propriedades como ganho, que controla a amplitude, lacunaridade, que controla a frequência, e oitavas, que são as camadas de ruído, nas quais cada camada se soma à(s) anterior(es). Na Figura 2.7, os gráficos (a), (b) e (c) exemplificam a soma de oitavas de ruídos unidimensionais.

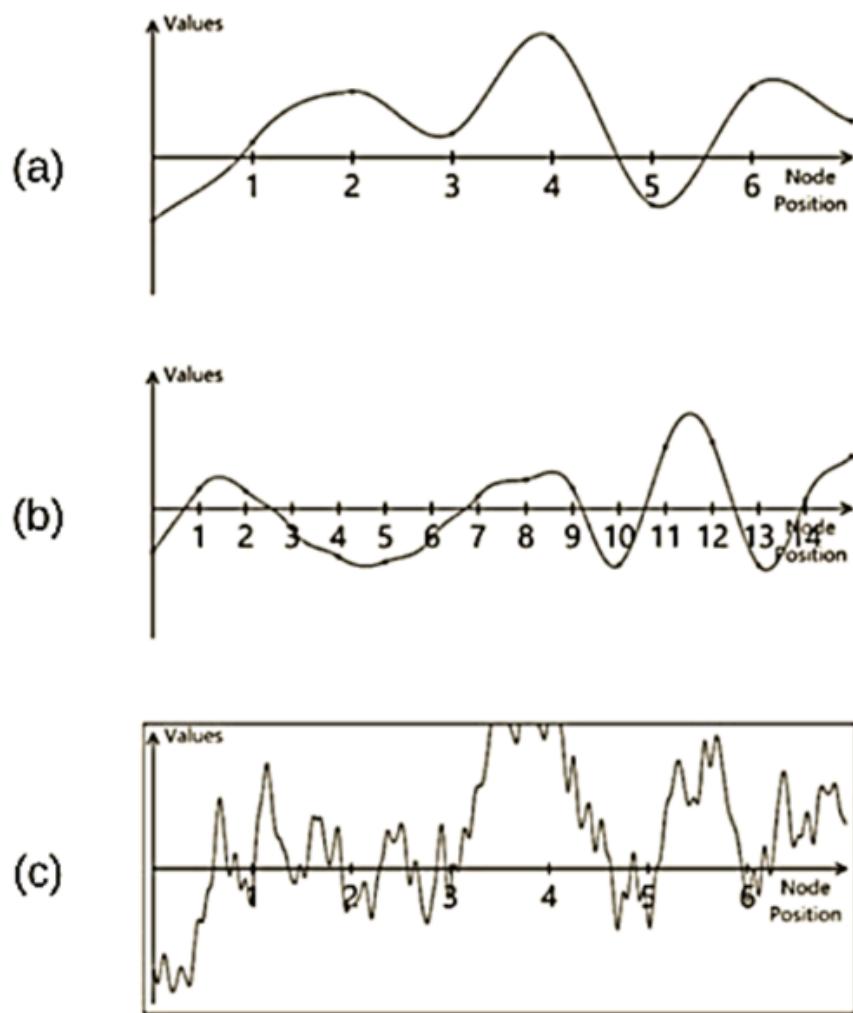


Figura 2.7 – Soma de duas oitavas de ruído. Fonte: [Schaal \(2017\)](#)

No gráfico (a), vemos uma onda suave de baixa frequência representando uma oitava de ruído. O gráfico (b) mostra uma segunda oitava de ruído onde uma frequência mais alta foi adicionada à anterior, resultando em uma forma diferenciada. Finalmente, o gráfico (c) apresenta o resultado da soma de (a) e (b), cada qual com uma frequência e

amplitude distintas, culminando em um padrão complexo. Esta técnica é usada para criar mapas de altura em modelos de terreno, simulando a variação encontrada na natureza.

2.3 Simulação de Processos Erosivos

Na modelagem de terrenos, os métodos de simulação visam recriar os fenômenos de erosão e alterações morfológicas verificados na dinâmica da natureza. O vocábulo “erosão” diz respeito aos processos de desgaste da superfície do terreno seguidos da remoção e do transporte de sedimentos por um ou mais agentes até sua deposição em terrenos de menor altitude. De acordo com Galin et. al. (2019), a erosão pode ser descrita em três etapas: a separação do sedimento da superfície do terreno; o transporte do sedimento pelo agente erosivo e a deposição do mesmo numa área rebaixada, como ilustrado na Figura 2.8.

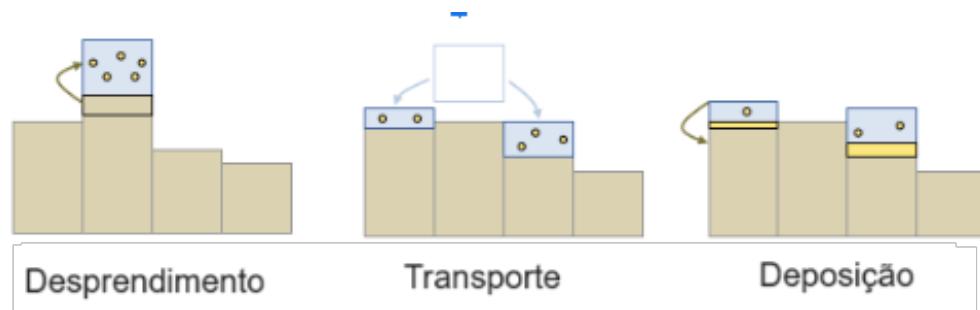


Figura 2.8 – Representação do processo erosivo. Fonte: [Galin et al. \(2019\)](#), tradução nossa)

Na perspectiva da computação gráfica, Musgrave, Kolb e Mace (1989), com o objetivo de introduzir mais realismo às paisagens naturais, propuseram algoritmos para simulação dos processos de erosão hidráulica e térmica em mapas de altura regulares. Tal procedimento, por outro lado, demandava um elevado custo computacional, uma vez que métodos baseados em simulação exigem um grande número de cálculos e implementação bastante complexa.

Benes e Forsbach (2001) introduziram uma representação de dados estruturados em camadas conjugada à aplicação do algoritmo de erosão térmica. A estrutura por camadas possibilitou a construção de uma grade bidimensional como um mapa de elevação, sendo que cada espaço da grade contém um array com elementos de informações das camadas (BENES; FORSBACH, 2001). Isto favorece a representação de solos de tipos distintos, incluindo rebaixos e cavernas. Esta contribuição inspirou outros autores tais como Št'ava et al. (2008) e Warszawski e Nikiel (2014) em seus modelos de erosão. Estes apresentaram uma ampla variedade de efeitos de modelagem baseados em erosão dinâmica, como nascentes, formação de rios, chuva, evaporação de áreas inundadas, precipitação etc.

2.3.1 Erosão Térmica

O material originado da erosão térmica consiste no desprendimento de sedimentos devido às diferenças de temperatura que causam expansão e contração nas rochas, levando à formação de fendas e suas subsequentes fraturas. A erosão termal busca simular a erosão de materiais do topo de uma superfície elevada, seu deslizamento pela encosta e, por fim, sua deposição num plano mais baixo.

O método descrito por [Olsen \(2004\)](#) se baseia em um critério de inclinação comparado a um ângulo crítico denominado ângulo *talus T*, que determina se o material em determinado ponto será movido ou estará em repouso. Se a inclinação de um ponto com altura h (como exemplificado na [Figura 2.9](#)) em relação ao seu vizinho excede T , o material nesse ponto é considerado instável e sujeito a movimentação. Neste caso, uma parcela da referida altura é transferida para o ponto adjacente, ou seja, o ponto com altura h_2 .

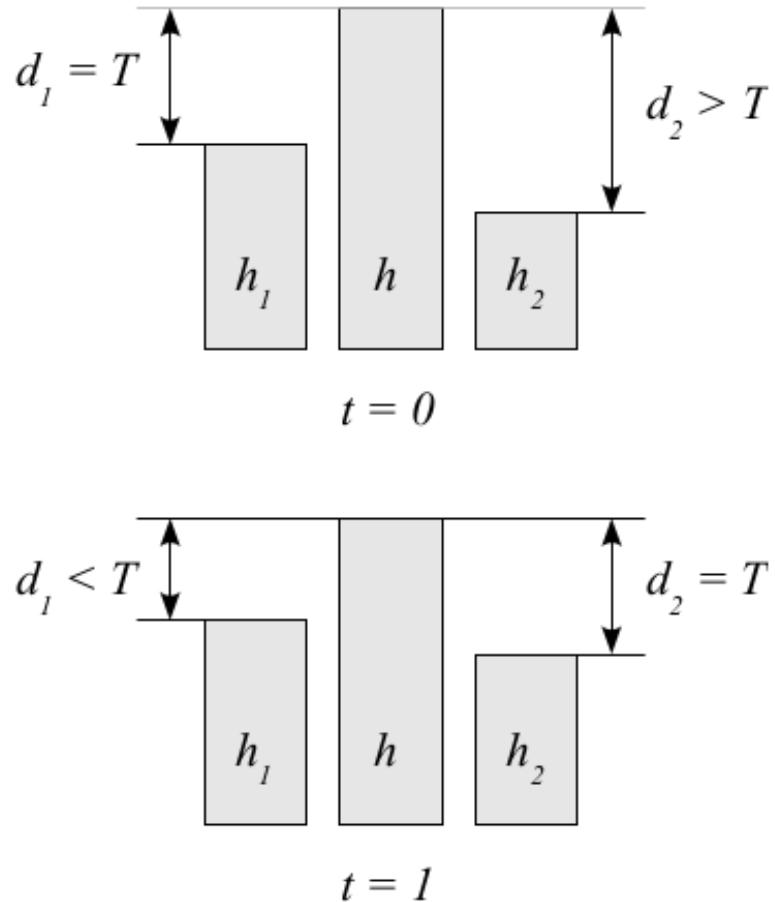


Figura 2.9 – Exemplo da iteração inicial do algoritmo de erosão térmica.

Fonte: [Olsen \(2004\)](#)

Este procedimento deve ocorrer repetidas vezes para se produzir efeito de erosão. Por fim, a erosão térmica tende a aplinar as encostas das colinas ou montanhas, de modo que, quanto menor for o valor do ângulo *talus*, maior será o impacto erosivo, dado que ocorrerá em declives não muito grandes.

2.3.2 Erosão Hidráulica

A erosão hidráulica constitui uma simulação que reproduz o processo de deposição de sedimentos transportados pelas águas pluviais e fluviais. A deposição destes sedimentos ocorre em conformidade com uma constante de saturação da água, que, por sua vez, é suscetível à influência direta da evaporação e da precipitação. Segundo [Santos, Griebeler e Oliveira \(2010\)](#), o poder das chuvas em deslocar partículas ou conjuntos delas está correlacionado à sua intensidade e à energia cinética das gotas ao atingirem o solo.

[Musgrave, Kolb e Mace \(1989\)](#) descreveram um modelo capaz de simular um processo de erosão hidráulica. Este modelo começa com a simulação da precipitação de água nos pontos mais altos de um terreno, permitindo que a água e os sedimentos, por ela carregados, se movam para pontos próximos mais baixos. A erosão causada pela água depende do seu volume e da quantidade de sedimentos que ela já contém.

Em cada ponto do terreno, denominado vértice, são considerados três elementos: a altura do terreno, a quantidade de água acumulada e os sedimentos que a água comporta. Com o passar do tempo, a água e os sedimentos em excesso são transferidos do vértice atual para os vértices vizinhos situados em alturas inferiores.

A transferência de água entre os vértices depende da quantidade de chuva, do volume de água acumulado e das diferenças de altura entre os vértices. Quando a transferência de água é nula ou negativa, significa que uma parte dos sedimentos se deposita no vértice atual. Se a transferência é positiva, ocorre a redistribuição da água e dos sedimentos.

A erosão hidráulica pode ser compreendida usando duas abordagens mostradas na [Figura 2.10](#): a Euleriana e a Lagrangiana.

A técnica euleriana aborda a erosão por meio de uma representação em grade, na qual o fluido contido em uma célula erode o material e o deposita nas células inferiores do terreno, levando em consideração a capacidade erosiva do fluido ([GALIN et al., 2019](#)). Na técnica lagrangiana, ocorre a simulação do movimento das partículas de água, onde são registradas informações como velocidade, capacidade de carga e posição da gota. Com base nesses dados, a partícula é deslocada ([BEYER, 2015](#)). Apesar de ser menos precisa que a abordagem euleriana, essa técnica é mais eficiente computacionalmente e mais adequada para a geração procedural de terrenos.

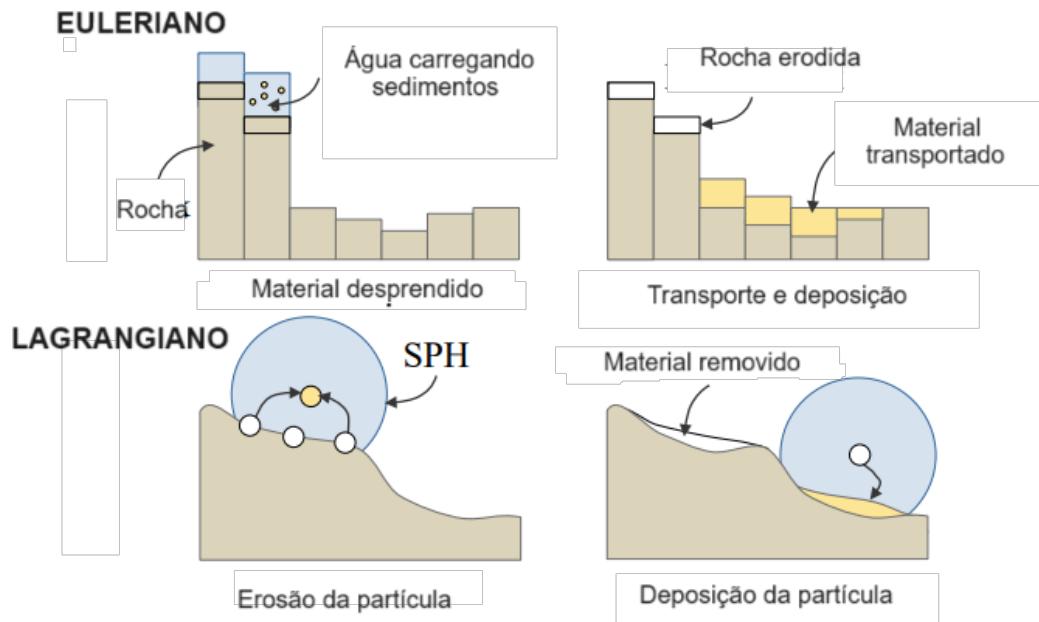


Figura 2.10 – Representação do processo erosivo euleriano e lagrangiano.

Fonte: [Galin et al. \(2019\)](#), tradução nossa)

[Rose e Bakaoukas \(2016\)](#) observam que modelos de erosão baseados em princípios físicos são uma ferramenta eficaz para conferir ao terreno de um jogo uma aparência mais realista e naturalmente desgastada. Segundo tais autores, embora a erosão hidráulica opere de forma consideravelmente mais lenta do que a erosão térmica e requeira aproximadamente três vezes mais memória, ela gera resultados substancialmente superiores. A simulação da erosão hidráulica utilizando a GPU pode ajudar a superar as limitações de velocidade inerentes ao algoritmo, oferecendo uma solução mais eficiente.

2.4 Trabalhos Relacionados

Um dos métodos mais antigos na geração de terrenos erodidos é baseado em fractais e foi iniciado por [Perlin \(1985\)](#), com contribuições subsequentes de [Musgrave, Kolb e Mace \(1989\)](#). Esse método opera por meio de um algoritmo que utiliza uma semente para gerar terreno em duas dimensões, cujo resultado pode ser então representado em um mapa de altura. Embora seja caracterizado por sua rapidez e baixo consumo de recursos de armazenamento, esse método não é capaz de produzir terreno com precisão, aparência natural ou conformidade matemática, conforme apontado por [Musgrave, Kolb e Mace \(1989\)](#).

A pesquisa realizada por [Benes e Forsbach \(2001\)](#) mostra esquematicamente uma estrutura de camadas que reúne materiais que possuem altura e atributos específicos

como, por exemplo, densidade e água acumulada. Esta disposição resulta em camadas compostas por diferentes materiais quando distribuídas sob o solo e sobre a superfície, tal como mostra a [Figura 2.11](#), possibilitando ainda a introdução de camadas de ar, e, consequentemente, estruturas subterrâneas.

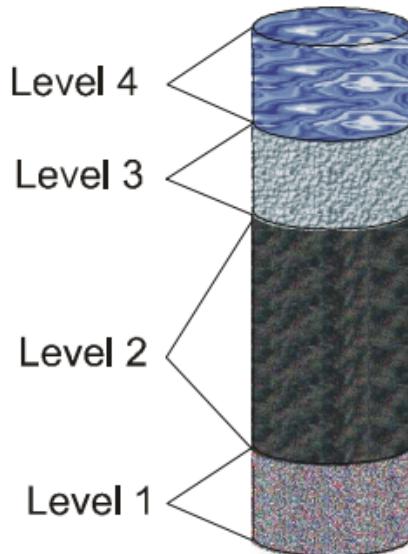


Figura 2.11 – Exemplo de estrutura geológica em camadas.

Fonte: [Benes e Forsbach \(2001\)](#)

Os autores implementaram um algoritmo de erosão térmica descrito por [Musgrave, Kolb e Mace \(1989\)](#) na estrutura de dados proposta, garantindo que a erosão não fosse efetuada somente sobre a superfície do terreno, mas também em todas as cavidades subterrâneas. Materiais precipitados dos tetos das cavernas também são capturados nesta implementação. [Benes e Forsbach \(2001\)](#) não utilizam dados volumétricos na sua representação em multicamada, alegando ser um desperdício de dados já que as camadas geralmente são muito espessas.

[Peytavie et al. \(2009\)](#) introduziram um *framework* de geração de terrenos que utiliza um modelo de representação híbrido conforme ilustrado na [Figura 2.12](#). O modelo associa uma estrutura de dados volumétrica e discreta para armazenar as camadas de materiais do terreno e uma representação implícita destinada a esculpir e criar a superfície do terreno. Com isso, fornece uma estrutura mais elaborada com camadas distintas de materiais capazes de reproduzir saliências rochosas e cavernas.

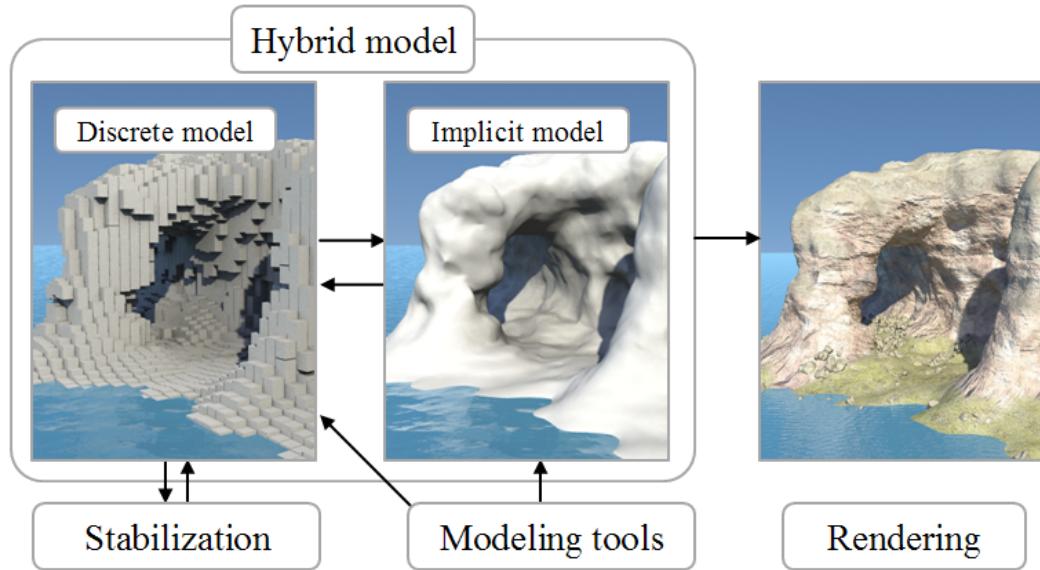


Figura 2.12 – Abordagem híbrida do framework *Arches*.

Fonte: [Peytavie et al. \(2009\)](#)

Em termos mais simples, é possível editar através de uma abordagem clara ou por um método mais indireto, escolhendo-se aquele que melhor se adapta à situação. O método indireto envolve moldar a fundação e criar uma malha de superfície usando algoritmos implícitos de poligonização de superfície. Por outro lado, o método direto envolve um modelo de camadas de material discretas. Após a edição, uma simulação de estabilização é aplicada para garantir que as camadas de areia e rocha se assentem em um estado estável, como mostrado na [Figura 2.13](#)

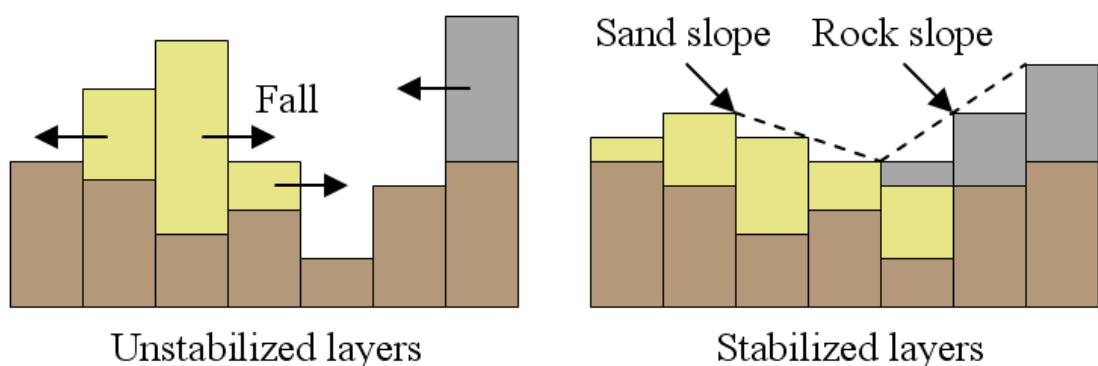


Figura 2.13 – Estabilização simulada de camadas do terreno.

Fonte: [Peytavie et al. \(2009\)](#)

O modelo de geração de terrenos em pilhas descrito por Löffler, Müller e Schumann (2011) apresenta uma nova abordagem para renderização em tempo real de terrenos complexos. Este modelo utiliza uma representação em camadas onde diferentes materiais são empilhados em cada ponto de uma grade, proporcionando uma maneira compacta de armazenar terrenos tridimensionais ricos em detalhes. A técnica proposta introduz um método para construir uma hierarquia de níveis de detalhe (*Level of Detail*) para essas pilhas de materiais. Tal procedimento é realizado por meio da decomposição dos dados em *octree*, estrutura de dados hierárquica na forma de árvore, onde cada nó possui oito filhos usados para dividir o espaço tridimensional em partes menores e mais gerenciáveis. Desta forma, foi possível melhorar o desempenho da renderização em aplicações interativas e de simulação em tempo real.

Quanto aos processos erosivos, Furtado (2019) sustenta que a estratégia inicial proposta para criar camadas de material erodido dependeu de um modelo de simulação centrado na estabilização, conforme delineado por Peytavie et al. (2009). Esse modelo compartilha semelhanças com simulações de erosão, mas a distinção principal está na existência de camadas de material estáveis. Ao contrário dos métodos de erosão, que geralmente tratam todos os materiais de maneira uniforme, a simulação de estabilização identifica determinados materiais como estáveis, permitindo que apenas os materiais soltos sofram movimento e assentamento. No âmbito deste método, consideram-se os materiais estratificados como estáveis, enquanto os materiais erodidos são vistos como soltos.

O algoritmo de erosão hidráulica implementado por Beyer (2015) baseou-se no movimento de partículas que transportam sedimentos de acordo com sua posição, capacidade de carga e velocidade dentro de um mapa de altura. Cada partícula ou gota possui um ciclo de vida dentro do qual realiza os processos de erosão e deposição, detendo informações a cada momento sobre a posição, a direção, a velocidade e a quantidade de água e de sedimentos.

Mei, Decaudin e Hu (2007) descrevem um modelo de erosão hidráulica no qual cada etapa lida com diferentes aspectos da interação entre a água e o terreno. O processo de erosão e deposição é calculado com base no campo de velocidade, determinando como o sedimento é removido e depositado ao longo do terreno. A água é reduzida devido à evaporação, completando o ciclo de simulação que é repetido para obter resultados progressivos. Cada uma das etapas utiliza os dados atualizados das etapas anteriores, e os valores são armazenados em uma estrutura de dados em camadas ilustrada na Figura 2.14.

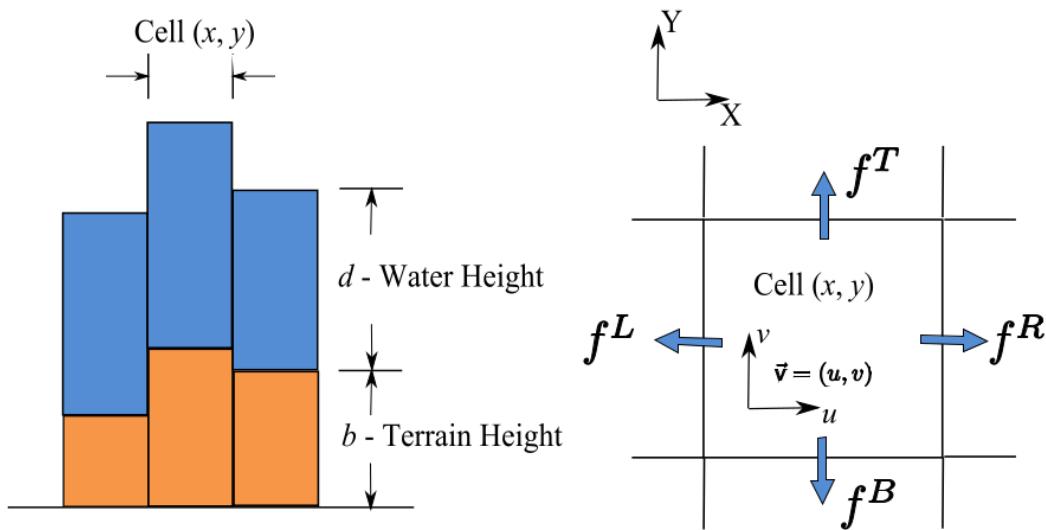


Figura 2.14 – Estrutura de dados de terreno para fluxos virtuais de água.

Fonte: [Mei, Decaudin e Hu \(2007\)](#)

A estrutura do terreno é representada por várias camadas de *arrays* 2D, onde cada célula armazena a altura do terreno, a altura da água, a quantidade de sedimento suspenso, o fluxo de saída e o vetor de velocidade. Este modelo é uma adaptação e extensão do modelo de tubo virtual proposto por [O'Brien e Hodgins \(1995\)](#), que transporta água através de tubos virtuais com base na diferença de pressão hidrostática entre células vizinhas. A implementação paralela em GPU deste modelo permite lidar eficientemente com simulações em terrenos de grande escala, abordando as limitações de métodos anteriores que requeriam muitas iterações sobre a grade 2D a cada passo de tempo.

A estrutura de dados multicamada descrita na pesquisa de [McDonald \(2022\)](#) permite a representação de terrenos com diferentes tipos de solo e camadas de sedimentos, como é mostrado na [Figura 2.15](#).

Cada seção do terreno contém informações específicas como tipo de material, densidade e resistência à erosão, sendo organizada em uma lista duplamente encadeada em um mapa de camada, o que facilita a manipulação e atualização das propriedades do terreno. O modelo também dispõe de uma estrutura denominada *section pool* que torna eficiente a gestão de alocação de memória das seções do terreno.

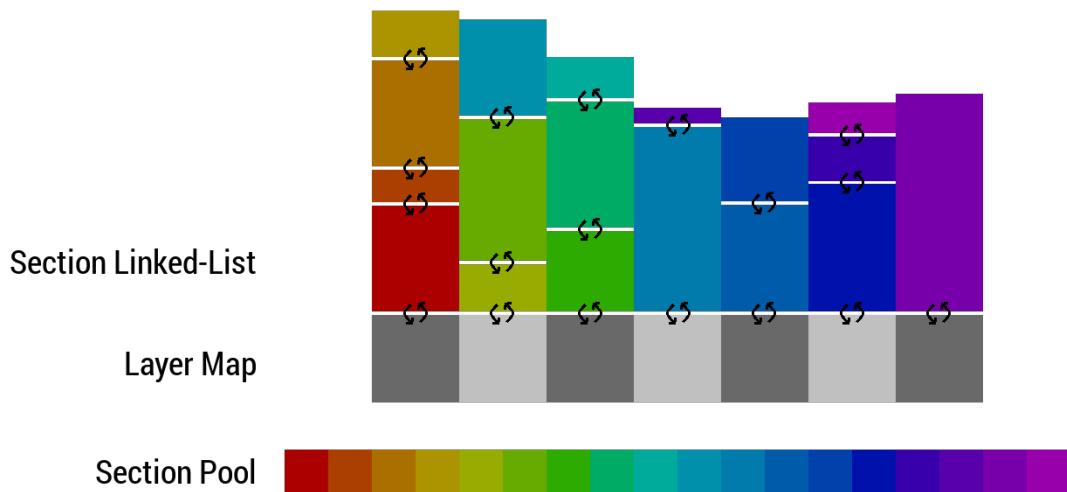


Figura 2.15 – Representação de terreno em camadas com encadeamento de seções.

Fonte: [McDonald \(2022\)](#)

O modelo de erosão de [McDonald \(2022\)](#) integra múltiplos processos: erosão hidráulica, na qual a água transporta partículas de solo; erosão térmica, afetando a estabilidade do solo; e erosão eólica, em que o vento remove partículas de superfícies expostas. Estas partículas são modeladas como agentes que interagem dinamicamente com o terreno, modificando suas propriedades conforme se movem e depositam sedimentos. Esse mecanismo permite simulações mais fidedignas das mudanças no terreno ao longo do tempo.

3 Modelagem Conceitual para Simulação de Terrenos

A geração em multicamada difere do modelo tradicional de mapa de altura por exibir uma estrutura de pilhas em toda extensão da grade, não se limitando somente à definição de alturas para cada posição do mesmo. Neste sentido, ao serem introduzidas etapas específicas, o processo de geração é alterado consideravelmente.

A [Figura 3.1](#) constitui uma visualização conceitual do desdobramento da geração de terrenos em múltiplas camadas proposto no presente trabalho.

O processo de geração de terreno que inicia com a entrada do usuário, onde são definidos os tipos de solo e parâmetros de geração e erosão. Na fase de preparação, esses parâmetros são utilizados para configurar a simulação em uma estrutura de dados discreta e bidimensional específica para a representação em camadas. Primeiramente, as colunas desta estrutura são dispostas de forma homogênea, não havendo nenhuma distinção entre elas. Em seguida, na etapa de composição de camadas, o terreno é estruturado com múltiplas camadas de diferentes tipos de solo empilhadas verticalmente.

Após a composição, o terreno é renderizado para visualização, mostrando a distribuição inicial das camadas de solo. A fase de simulação de erosão aplica os parâmetros de erosão definidos pelo usuário, modificando a superfície conforme a simulação progride. Este processo é iterativo, onde a malha do terreno é atualizada continuamente para refletir as mudanças causadas pela erosão e deposição de sedimentos. O resultado final é um perfil diferenciado de terreno posteriormente a consecutivas iterações da simulação erosiva, refletindo o desgaste dos solos em terrenos reais.

Este processo leva em conta a interação entre as camadas para determinar a ordem dos materiais nas colunas. Após o estabelecimento das variações de altitude, os dados são convertidos em malhas poligonais e, então, o terreno é renderizado. Paralelamente, ocorre o processo erosivo, no qual os materiais da superfície podem receber sedimentos ou sofrer erosão, além de interagirem com aqueles da camada inferior. Deve-se destacar que cada material possui características únicas que respondem de maneira diferente aos agentes erosivos.

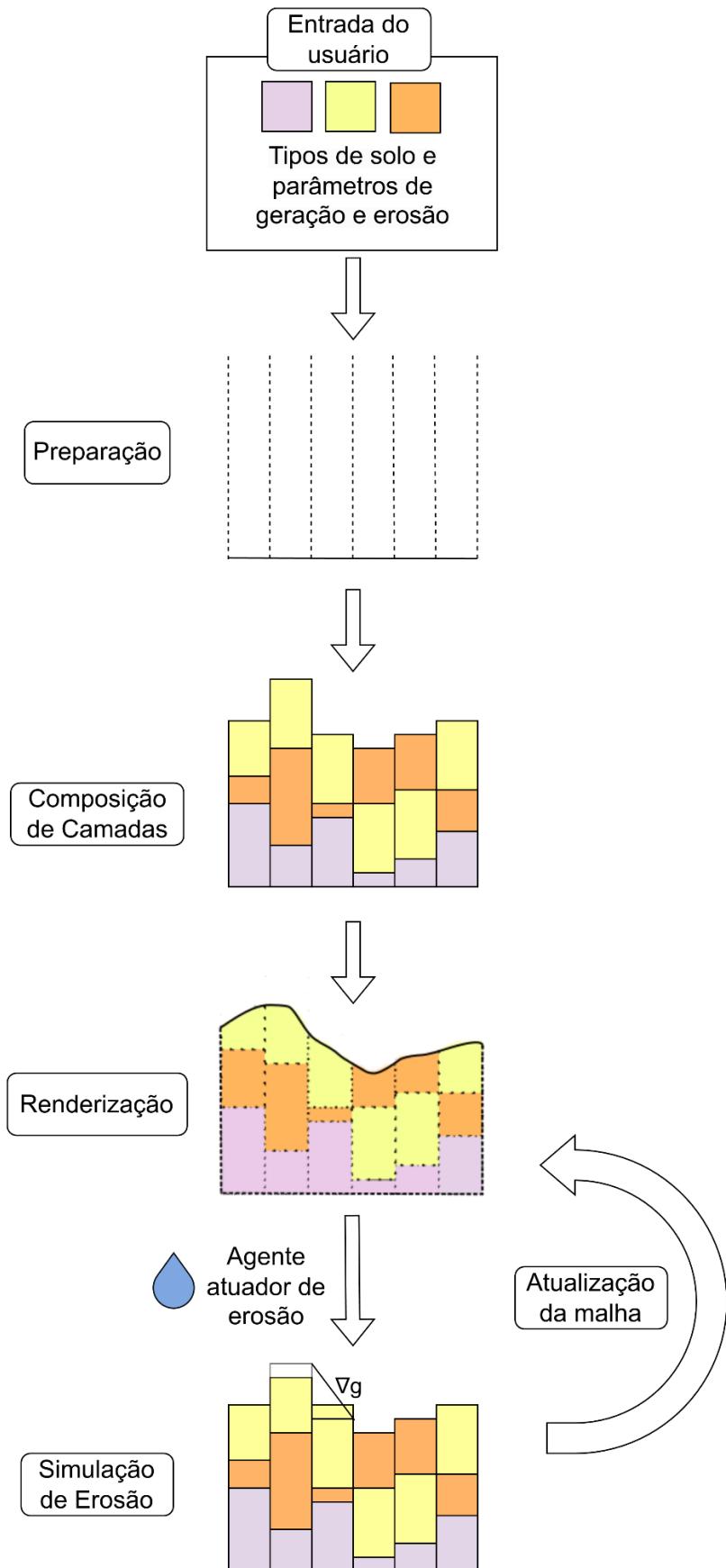


Figura 3.1 – Etapas do modelo de geração proposto.

3.1 Estrutura e Dinâmica das Camadas de Terreno

A representação estrutural e a dinâmica das camadas de terreno são fundamentais para a simulação realista de processos geológicos em ambientes virtuais. A ideia do mapa de camada (ou *layermap*), conforme proposto por McDonald (2022), oferece uma base sólida para representar variações de solo e erosão em um modelo de terreno procedural. Este conceito permite uma simulação detalhada bem como a dinâmica da erosão, onde diferentes tipos de solo reagem de maneira singular aos processos erosivos.

A Figura 3.2 exemplifica de forma abstrata a representação em camadas em uma grade discreta de 4×4 com posições de tamanho h^2 . A cada posição é associada uma pilha de camadas, comportando diferentes tipos de solo.

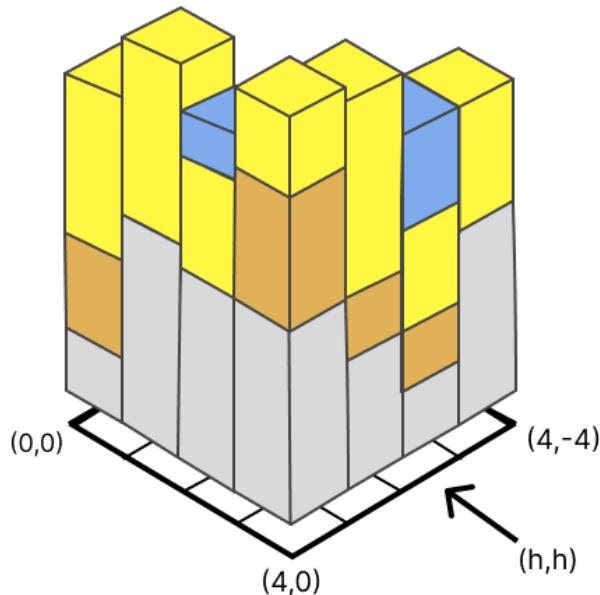


Figura 3.2 – Modelo representativo de terreno em camadas.

Verifica-se, através desta representação genérica, que existe nas paisagens naturais uma ampla variedade de composição de solos, mesmo com ausência de um deles, além das diferenças de profundidade entre os mesmos. No presente trabalho, no entanto, preferiu-se não omitir nenhuma deles, a fim de garantir a integralidade do conjunto.

3.1.1 Mapa de Camada

O mapa de camada constitui uma estrutura de dados discreta que armazena, em uma grade bidimensional, seções de solo empilhadas em cada posição do mapa. Tais seções correspondem a uma estrutura unitária contendo as informações como altitude, tipo de solo e nível de saturação da água. Observa-se na Figura 3.3 a organização de dados na estruturação em camadas:

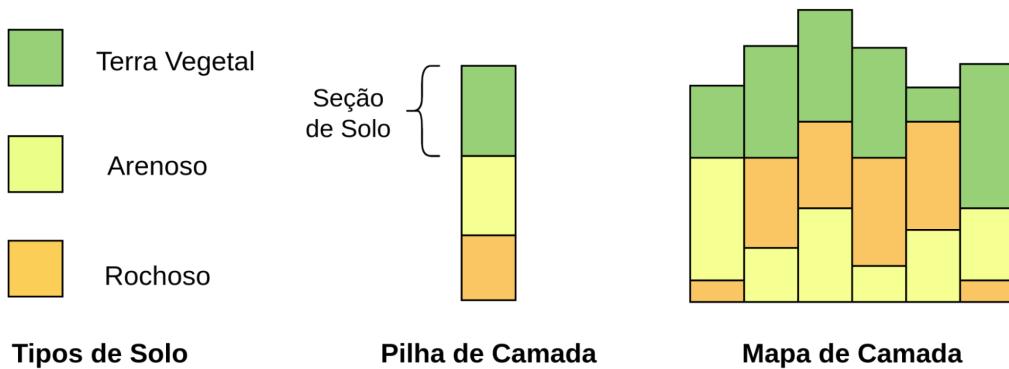


Figura 3.3 – Estruturas de dados agregadas ao modelo em camadas.

Os tipos de solo exemplificados na [Figura 3.3](#) são: solo rochoso, cuja composição é constituída principalmente por fragmentos de rochas e minerais; solo arenoso, de textura leve caracterizado pelo alto teor de areia, sendo normalmente granuloso; e a terra vegetal ou solo orgânico, formado pela decomposição de matéria animal, vegetal e de microorganismos. Neste sentido, de forma a considerar todas estas características individuais dos tipos de solo, foram selecionados os seguintes parâmetros, avaliados nos estudos de [McDonald \(2022\)](#) e [Santamaría-Ibirika et al. \(2013\)](#):

- Profundidade;
- Permeabilidade;
- Taxa de absorção da água.

A profundidade diz respeito à distância da superfície do solo até sua camada mais profunda, sendo possível especificar o intervalo de profundidade do tipo de solo. A permeabilidade refere-se à capacidade do solo de dar passagem à água. A taxa de absorção, por sua vez, indica a quantidade relativa de água que infiltra no solo em determinado tempo.

Em cada posição, o mapa de camada armazena uma pilha de seções de solos duplamente encadeadas. Verifica-se na [Figura 3.4](#) que o mapa possui uma referência à seção do topo de cada pilha de camada. Neste caso, na consulta de uma posição arbitrária, obtém-se a seção de solo localizada na superfície dessa posição.

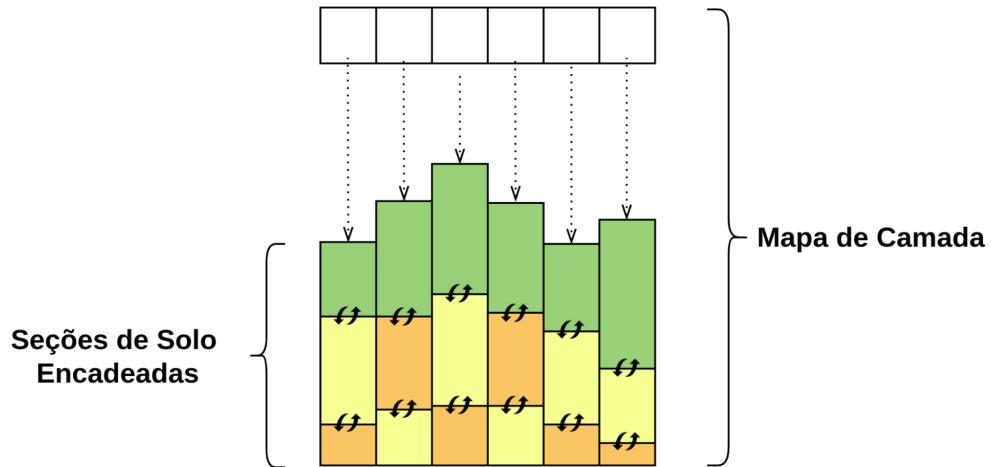


Figura 3.4 – Organização interna do mapa de camada.

Observa-se ainda na [Figura 3.4](#) que cada seção possui ponteiros de encadeamento com a anterior (subjacente) e a próxima (superior), formando as camadas de maneira discreta. A interação interna entre camadas no modelo apresentado é estruturada de tal forma que o acesso às seções de solo abaixo da superfície depende exclusivamente da utilização de ponteiros, que estão organizados de maneira a conectar cada seção de solo diretamente àquelas imediatamente acima e abaixo dela.

3.1.2 Composição de Camadas

A composição de camadas refere-se ao processo de empilhamento de diferentes materiais em cada posição no mapa de altura para criar uma representação em múltiplas camadas. Esta etapa envolve essencialmente a atribuição de materiais ou propriedades específicas a diferentes faixas de altitude.

Inicialmente, o preenchimento das camadas depende da definição, por parte do usuário, dos parâmetros e dos tipos de solo, os quais serão armazenados em um *array*. Os índices atribuídos a cada tipo de solo seguem a mesma sequência determinada na escolha do usuário.

A [Figura 3.5](#) mostra o fluxograma referente a esta etapa de composição. Uma vez definido os parâmetros pelo usuário, é instanciado o mapa de camada com tamanho da grade n^2 sendo $n \in \mathbb{Z} \mid n \geq 2$. O algoritmo inicia uma repetição para cada tipo de solo presente no array supracitado. Um mapa de ruído é gerado utilizando-se Ruído de Perlin e *Fractional Brownian Motion* para produzir padrões fractais que imitam fenômenos naturais.

Itera-se o mapa de ruído em cada posição (x, y) . Considera-se um tipo de solo t com intervalos de profundidade específicos, denotados por n_t e m_t que representam os

extremos de profundidade para o tipo de solo, tal que $0 \geq n_t \leq m_t \leq 1$. Assim, a altura $h(x, y)$ com uma função de ruído $\text{noise}(x, y)$ para uma dada seção de solo do tipo t é calculada pela seguinte equação:

$$h(x, y) = n_t + (m_t - n_t) \cdot \text{noise}(x, y) \quad (3.1)$$

O cálculo de 3.1 é realizado através da interpolação linear do valor do ruído entre os extremos de profundidade n_t e m_t do tipo de solo t .

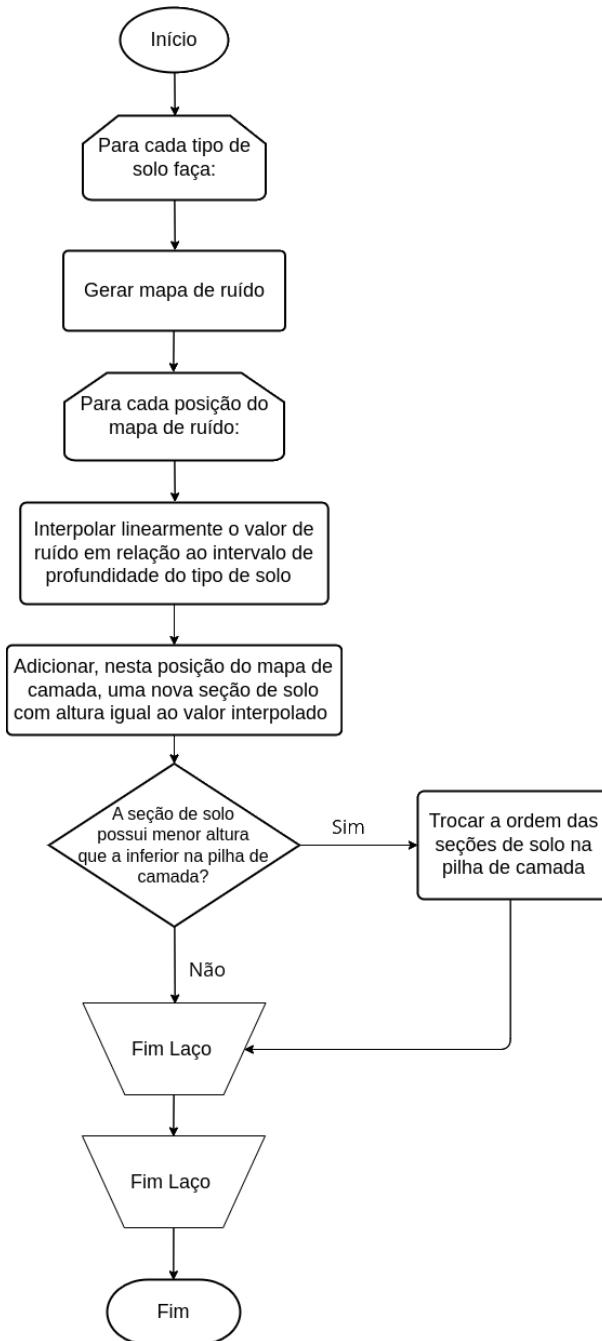


Figura 3.5 – Fluxograma do algoritmo de composição de camadas do terreno.

Após o cálculo da altura de uma seção de solo, ocorre a inserção da mesma no mapa de camada. Neste momento, acontecem, dinamicamente, mudanças na sequência das seções nas pilhas de camadas. A Figura 3.6 ilustra esta alteração de acordo com a variação de altura entre elas.

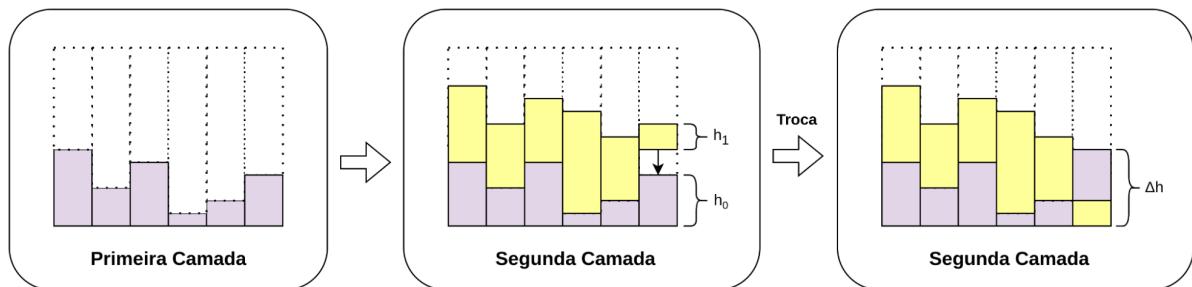


Figura 3.6 – Exemplo de preenchimento e interação de duas camadas.

O preenchimento da primeira camada sucede normalmente. É necessário destacar que as seções de solo adicionadas em camadas superiores são suscetíveis à mudança na ordem de sua pilha. Na Figura 3.6, supõe-se que a diferença $\Delta h = h_1 - h_0 \leq 0$ acarreta a troca das seções na pilha de camada, uma vez que o resultado da referida subtração é negativo.

3.2 Simulação de Erosão Hidráulica

No contexto da geração de terreno baseada em simulação, o modelo em duas dimensões enfrenta limitações substanciais. A utilização de um mapa de altura simples durante a simulação da erosão para a geração de terreno pressupõe a existência de propriedades de erosão homogêneas em todo o mapa (MCDONALD, 2022). A presente pesquisa se beneficia do método de erosão exclusivo para mapas de altura baseado em partículas exposto por Beyer (2015) e valeu-se da contribuição de McDonald (2022) para modelagem em mapas de camada.

O método empregado levou em consideração a movimentação de partículas (ou gotas d'água) no solo. A Figura 3.7 mostra o processo cíclico da simulação de erosão hidráulica



Figura 3.7 – Diagrama do ciclo de vida da gota d’água na simulação erosiva.

O processo é composto por 5 etapas, podendo algumas delas ser divididas em subprocessos, a saber:

- **Incremento da Fonte de Água:** onde cada gota d’água é inicializada com volume de água.
- **Fluxo da Gota:** descreve o processo de movimento da gota d’água baseado na direção do gradiente do terreno.
- **Absorção de Água:** nesta etapa, o solo absorve uma parte da água da gota, baseado na taxa de absorção específica do tipo de solo, além da transferência de água do solo superficial para camadas mais profundas, simulando o processo de infiltração.
- **Erosão e Deposição:** definem o processo de erosão (quando a gota remove material do solo) ou deposição (quando a gota deposita sedimentos no solo), dependendo da capacidade de transporte de sedimento pela gota e da topografia do terreno.
- **Evaporação da Água:** engloba a perda de volume da água da gota e do solo.

O ciclo comprehende duas condições de parada: quando a gota ultrapassa uma constante de máximo alcance K_a previamente definida ou quando se move para uma posição fora do mapa.

3.2.1 Dinâmica da Partícula

Na dinâmica de erosão hidráulica do gerador de terrenos proposto, gotas de água são simuladas ao percorrerem a superfície do terreno, interagindo com o solo de modo a imitar processos naturais. À medida que as gotas se movem, acarretam erosão no solo, transportando sedimentos dos pontos mais elevados para os mais baixos, onde se depositam de acordo com o acúmulo de sedimentos contidos na partícula. Este processo iterativo resulta na formação de características terrestres, como vales e canais, modificando progressivamente a paisagem de acordo com padrões de fluxo de água.

Esta seção compreende as etapas anteriores e posteriores dos eventos erosão e deposição. Cada partícula de água detém as seguintes propriedades:

- O vetor de posição $p(x, y)$, gerado aleatoriamente.
- O volume de água w com valor inicial igual a 1.
- O vetor de direção D , normalizado no intervalo $[0,1]$.
- A velocidade da partícula V com valor inicial igual a 1.
- A quantidade de sedimento carregada q e sua capacidade máxima de acúmulo de sedimentos Q , ambas inicializadas com valor igual a 0.

Quando uma partícula é gerada, sua posição em um ponto de coordenadas x e y é dada por:

$$p = (x + \alpha, y + \beta) \quad (3.2)$$

onde α e β são, respectivamente, os desvios horizontal e vertical na posição da gota d'água dentro da célula. Entende-se por células os espaços compreendidos por quatro vértices no mapa de camada. O fluxo da água consiste na interpolação bilinear dos gradientes dos vértices em uma célula arbitrária (ver [Figura 3.8](#)).

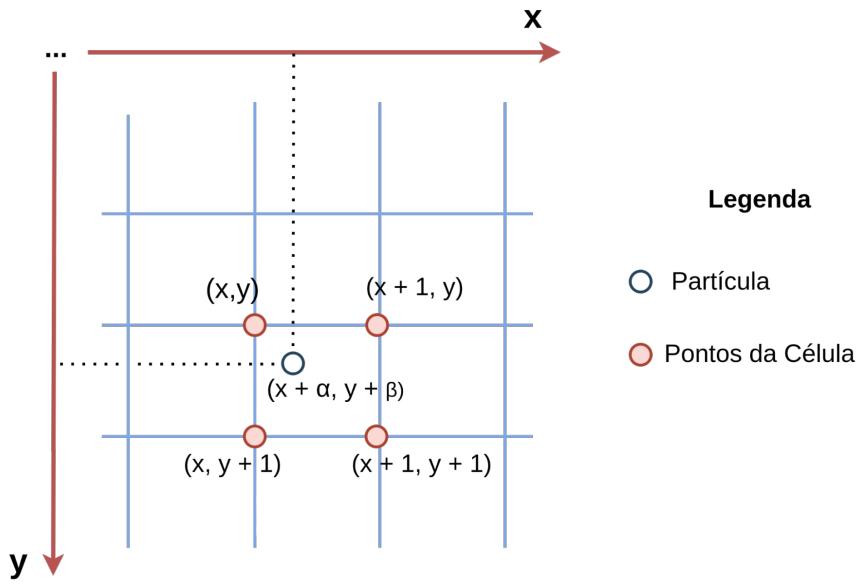


Figura 3.8 – Posicionamento ilustrativo dos nós da célula no mapa.

Considerem-se as posições representativas das seções de solo nos cantos da célula $P_1(x, y)$, $P_2(x+1, y)$, $P_3(x, y+1)$ e $P_4(x+1, y+1)$ com alturas h_1, h_2, h_3, h_4 , respectivamente. O gradiente de cada uma das posições é dado por:

$$\begin{cases} g(P_1) = (h_2 - h_1, h_3 - h_1) \\ g(P_2) = (h_2 - h_1, h_4 - h_2) \\ g(P_3) = (h_4 - h_3, h_3 - h_1) \\ g(P_4) = (h_4 - h_3, h_4 - h_2) \end{cases} \quad (3.3)$$

Ao se interpolar bilinearmente os gradientes dos pontos em questão, temos que o gradiente na posição da partícula p é definido como:

$$g(p) = \begin{pmatrix} (h_2 - h_1) \cdot (1 - \beta) + (h_4 - h_3) \cdot \beta \\ (h_3 - h_1) \cdot (1 - \alpha) + (h_4 - h_2) \cdot \alpha \end{pmatrix} \quad (3.4)$$

O próximo passo consiste em atualizar a direção da partícula a partir do vetor gradiente resultante. Essa nova direção D' é calculada como uma combinação ponderada entre g e a direção anterior, denominada D . Uma constante K_n , que varia de 0 a 1, é utilizada para simular a inércia das partículas, controlando a influência de cada vetor na direção final (BEYER, 2015). A equação a seguir descreve este evento:

$$D' = D \cdot K_n - g \cdot (1 - K_n) \quad (3.5)$$

Quando o coeficiente K_n assume o valor 0, a partícula vai mudar imediatamente de direção, fluindo encosta abaixo, uma vez que a direção anterior D será ignorada. Por outro lado, quando a constante K_n é igual a 1, a partícula não mudará de direção, tendo em vista que o gradiente g será completamente desconsiderado.

A nova direção de movimento D' é normalizada. Utilizando-se o mesmo princípio explicado na Figura 3.8, realiza-se uma interpolação bilinear das alturas dos pontos da célula na posição da partícula p , obtendo h_p . Em seguida, a posição da partícula é atualizada com a adição da posição atual p à nova direção D' , isto é, $p' = p + D'$. Efetua-se a mesma interpolação na nova posição p' e obtém-se a variação de altura Δh entre as posições final e inicial da partícula, como se segue:

$$\Delta h = h_{p'} - h_p \quad (3.6)$$

A diferença de altura (Δh) é empregada para determinar se a gota d'água moveu-se em declive ou em ascensão, conforme demonstrada por Beyer (2015). Esta mesma diferença influenciará no processo de erosão e deposição, cuja dinâmica será discutida na Subseção 3.2.2.

Anterior ao processo erosivo, a partícula transfere uma parcela do volume d'água para as quatro seções da célula com base na taxa de absorção do respectivo tipo de solo e na quantidade de sedimentos carregados. Tal distribuição ocorre a partir de uma interpolação bilinear ajustada à posição da partícula. O volume absorvido pelo solo VL é calculado através da seguinte expressão:

$$VL = w \cdot A \cdot K_t \quad (3.7)$$

Onde:

- w é a quantidade de volume d'água da partícula
- A é a taxa de absorção do tipo de solo correspondente
- K_t é a constante de escoamento (proporção da água da gota absorvida pelo solo em cada interação)

Nota-se que o acúmulo maior de sedimentos na partícula impede uma transferência significativa de água para o solo. Com isto o volume de água presente na seção de solo no topo da posição $P(x, y)$ e na partícula são atualizados, como indicam as equações a seguir:

$$\begin{aligned} w_P &= w_P + VL \\ w &= w - VL \end{aligned} \quad (3.8)$$

Caso a saturação da seção de solo ultrapasse o valor-limite, será verificado o processo de infiltração, ou seja, de absorção da água pela camada subjacente. Inicialmente, calcula-se a diferença de altura ΔH entre a seção de solo superficial com posição P e a seção diretamente subjacente a ela PS , ambas na mesma pilha de camada. Em seguida, obtém-se a quantidade de água que será transferida de P para PS através das seguintes equações:

$$\begin{aligned} VL_T &= w_P \cdot \rho_P \cdot (1 - \Delta H) \\ S_P &= S_P - VL_T \\ S_{PS} &= S_{PS} + VL_T \cdot A_{PS} \end{aligned} \tag{3.9}$$

Onde:

- VL_T é a quantidade de água de transferência
- S_P e S_{PS} são, respectivamente, as saturações de água da seção P e PS
- ρ_P é a permeabilidade do tipo de solo da seção P
- A_{PS} é a taxa de absorção do tipo de solo da seção PS

Considera-se ΔH no cálculo da transferência, visto que, caso a seção superior detenha uma altura consideravelmente elevada em relação à subjacente, a infiltração será afetada. Além disso, uma parcela de água pode ser desperdiçada durante a transferência conforme a capacidade de absorção da seção subjacente.

Após o processo erosivo e de deposição, a iteração da partícula finaliza com a mudança de sua velocidade e a decorrente evaporação. As equações a seguir, definidas por [Beyer \(2015\)](#), demonstram esta etapa, sendo V' e w' a velocidade e o volume de água atualizados da partícula, respectivamente.

$$\begin{aligned} V' &= \sqrt{V^2 + \Delta h \cdot K_g} \\ w' &= w \cdot (1 - K_v) \end{aligned} \tag{3.10}$$

A velocidade é atualizada com mais ênfase na velocidade atual da partícula V em comparação com a diferença de altura ou inclinação h , calculada na etapa de movimentação da partícula. A constante K_g é usada para simular o efeito gravitacional e o volume de água w evapora gradualmente conforme uma constante de evaporação K_v ao longo do seu fluxo.

3.2.2 Erosão e Deposição

Os modelos de erosão fundamentados em princípios físicos constituem uma ferramenta proveitosa para conferir ao terreno uma estética mais natural e desgastada. Erosão e deposição de sedimentos, intercalados pelo transporte dos mesmos, constituem operações fundamentais em algoritmos de modelagem de terreno, nos quais são simulados processos de movimentação e redistribuição de partículas em ambientes virtuais.

O modelo adotado neste trabalho apoia-se na verificação de que uma partícula de água capta sedimentos se estiver supersaturada, descarta-os sobre o solo (deposição), ou, do contrário, os remove (erosão). Em seguida à movimentação da partícula e a absorção de água pelo solo, a capacidade máxima de sedimentos Q é avaliada. Em cada iteração do ciclo de vida da partícula, a referida capacidade é atualizada para um novo valor Q' , calculado de acordo com a seguinte expressão:

$$Q' = \max(-\Delta h, K_s) \cdot V \cdot w \quad (3.11)$$

Onde:

- Δh é a diferença de altura obtida com a movimentação da partícula.
- K_s é a constante de inclinação mínima.
- V e w são, respectivamente, a velocidade e o volume d'água da partícula.

A capacidade máxima de sedimentos Q indica quanta carga uma partícula de água pode transportar. Se a partícula desce uma encosta íngreme (Δh negativo), poderá carregar mais sedimentos devido à maior energia adquirida. Por exemplo, uma partícula transitando por uma descida íngreme com alto volume e velocidade aumenta significativamente Q , permitindo transportar mais sedimentos. Em contrapartida, em terrenos planos ou com pouca declividade, a capacidade é limitada pela constante K_s , refletindo a menor energia disponível para erosão e transporte de material. É importante ressaltar que K_s garante que a capacidade Q não possua valor negativo.

Se a quantidade de sedimentos s da partícula for menor que a sua capacidade máxima Q , seções de solo sofrerão a erosão dentro de uma constante de raio K_r que determina a área afetada. Para determinar a quantidade de sedimento que cada ponto do mapa perde durante o processo erosivo, valores de peso de impacto são calculados e atribuídos para cada posição do mapa. Neste caso, itera-se o mapa em cada posição com coordenadas C_x e C_y , levando-se em conta os n pontos cuja distância entre as mesmas seja, no máximo, o raio K_r . Para qualquer ponto localizado neste raio com coordenadas x_i e y_i , tem-se que seu peso W_i é dado por:

$$W_i = \left(1 - \frac{\sqrt{(x_i - C_x)^2 + (y_i - C_y)^2}}{K_r} \right) \cdot \frac{1}{W_k} \quad (3.12)$$

Onde W_k é a soma de todos os pesos dos n pontos (x_k, y_k) contidos no raio K_r , isto é:

$$W_k = \sum_{k=0}^n 1 - \frac{\sqrt{(x_k - C_x)^2 + (y_k - C_y)^2}}{K_r} \quad (3.13)$$

A equação de peso W_i calcula o impacto relativo da erosão em um ponto (x_i, y_i) com base em sua distância radial do centro C . Esta distância é dividida pelo raio de influência K_r e então é subtraída de 1 para inverter a seguinte relação: pontos mais próximos do centro terão pesos maiores (pois a subtração resulta em um número menor), enquanto pontos mais distantes terão pesos menores. A normalização pela divisão de W_k garante que os pesos diminuam linearmente de 1 a 0, conforme a distância até o raio de influência.

A quantidade de sedimentos erodidos E pode ser representada pela equação a seguir:

$$E = \min((Q - q) \cdot K_e, -\Delta h) \cdot W \cdot S \quad (3.14)$$

Onde:

- Q e q são, respectivamente, a capacidade máxima e quantidade atual de sedimentos da partícula.
- Δh é a diferença de altura obtida com a movimentação da partícula.
- K_e é a taxa de velocidade de erosão .
- W e S são, respectivamente, o peso de impacto e a saturação da seção de solo afetada.

A equação em 3.14 descreve a quantidade de sedimentos erodidos de um ponto específico. O termo $\min((Q - q) \cdot K_e, -\Delta h)$ determina a quantidade de erosão baseada na diferença de capacidade máxima e quantidade atual de sedimento da partícula $(Q - q)$, ajustada pela taxa de velocidade de erosão K_e . Deve-se ressaltar que a gota nunca acumula mais sedimentos que a diferença de altura Δh , caso contrário ocasionaria fissuras no terreno. Este valor é então modulado pelo peso de impacto W elucidado anteriormente, assim como pela saturação S , que representa o quanto saturado de água o solo está no ponto de impacto. Valores altos de W e S intensificam a erosão, traduzindo um maior impacto da gota carregada com sedimentos e a suscetibilidade do solo à erosão, respectivamente.

Na condição de a partícula extrapolar sua capacidade máxima de armazenar sedimentos, observa-se a deposição dos mesmos. A quantidade de deposição T é obtida com base na equação que se segue:

$$T = (q - Q) \cdot K_d \quad (3.15)$$

Caso a gota de água exceda sua capacidade de transporte de sedimentos, uma fração do seu excedente será depositada no solo, conforme determinado pela taxa de velocidade de deposição K_d . Em seguida, interpola-se bilinearmente T entre os quatro cantos da célula na qual a gota se encontra.

4 Implementação do Sistema de Geração de Terrenos

Com o propósito de atestar que o modelo de erosão funcione de maneira eficiente, definir uma estrutura arquitetural do sistema é um requisito indispensável. A concepção desta arquitetura requer uma análise cuidadosa dos componentes do sistema, como a geração de terreno, da simulação erosiva e da interação de camadas e como eles se integram para formar um todo coeso. Diagramas como os de classes e de sequência oferecem a oportunidade de visualizar e estruturar as relações entre os componentes, facilitando o entendimento do fluxo de dados e do processamento dentro do sistema.

A Figura 4.1 exibe a decomposição dos dois subsistemas formulados para a sua implementação, os quais constituem o módulo de geração (fundo verde) e o módulo de erosão ou simulação (fundo azul), assim como a interação com componentes do motor gráfico que compõem as APIs (*Application Programming Interface*) que dão suporte à visualização do sistema.

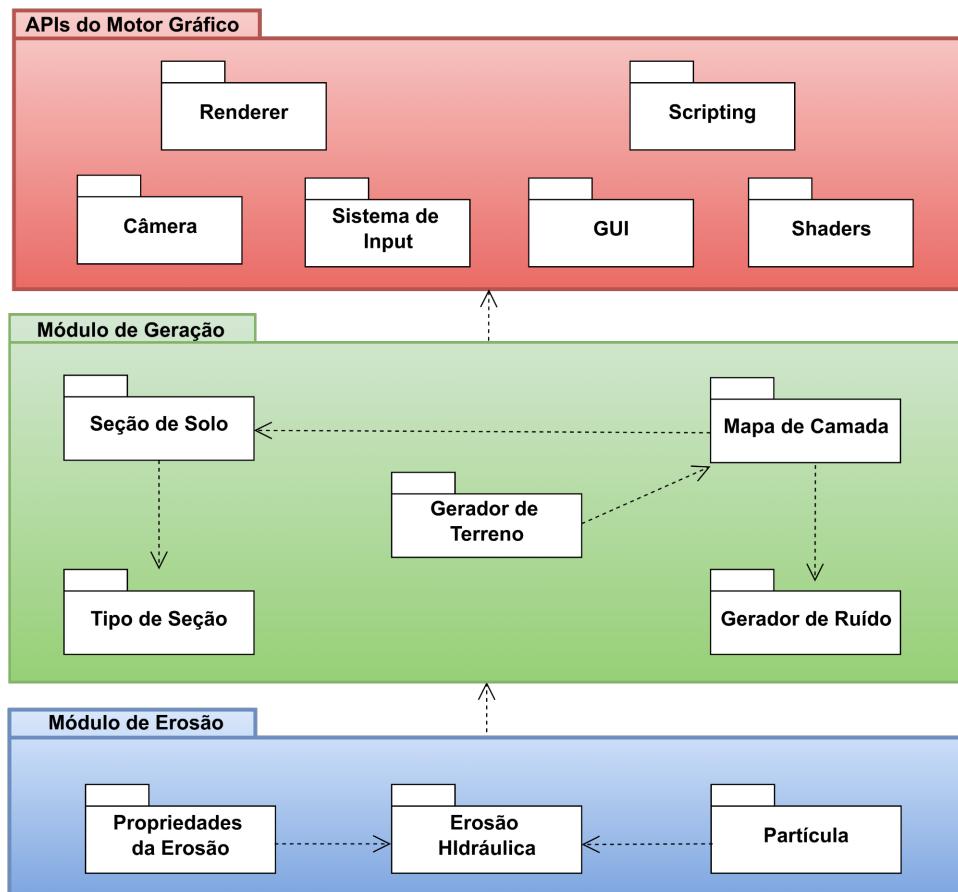


Figura 4.1 – Módulos do sistema desenvolvido e sua interação com o motor gráfico.

A arquitetura da aplicação, aqui implementada, é concebida como uma série de componentes interconectados que interagem com as do motor gráfico. Estas disponibilizam recursos como renderização, câmera, entrada de usuário, interface gráfica (GUI) e *shaders* necessários para a geração, visualização e interação do usuário com o terreno gerado.

O Módulo de Geração, com fundo verde, é responsável pela criação da estrutura física do terreno. Componentes como seção de solo e tipo de seção definem as características e a diversidade do terreno. É no mapa de camada onde a complexidade do terreno é construída a partir do gerador de ruído, sobrepondo as camadas geradas para formar a topografia. O gerador de terreno, por sua vez, realiza a transformação dos dados estruturais do mapa de camada em uma representação tridimensional, mapeando os vértices e triângulos das malhas poligonais que compõem a superfície do terreno.

Finalmente, o Módulo de Erosão, com fundo azul, incorpora as propriedades da erosão e a dinâmica das partículas para aplicar a simulação de erosão hidráulica. Este módulo utiliza as propriedades do solo e a movimentação das partículas de água para modelar a erosão, o que resulta em um desgaste do terreno, demonstrando a integração direta entre os dados de simulação e a representação visual do terreno.

A técnica de renderização do terreno escolhida neste trabalho foi aquela de malhas triangulares de superfície. Uma amostra desta técnica pode ser observada na [Figura 4.2](#).

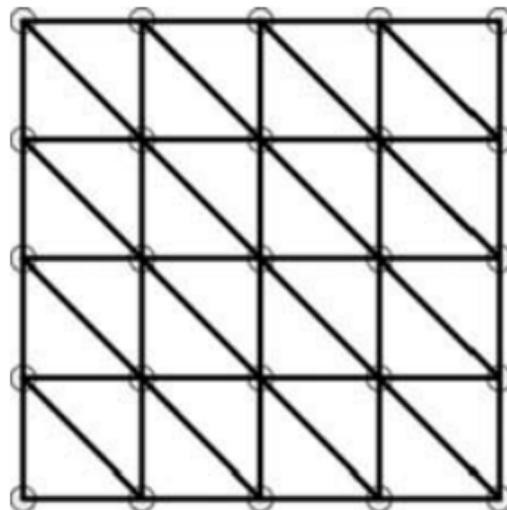


Figura 4.2 – Triangulação da grade de terreno. Fonte: [Koca e Güdükbay \(2014\)](#)

Cada ponto no mapa é tratado como um vértice em uma grade regular, e os triângulos são formados conectando esses vértices. Após os triângulos serem definidos, eles são renderizados pela *pipeline* gráfica da GPU, convertendo-os em pixels na tela com base nas propriedades dos vértices (posição, cor, normais etc.).

Apesar de ser um método simples e destinado majoritariamente a mapas de altura, o modelo em camadas ou de pilhas permite a renderização de volume, como visto no modelo de [Peytavie et al. \(2009\)](#).

4.1 Módulo de Geração

O módulo de geração representa uma camada crítica, responsável pela criação de estruturas de dados que irão definir a morfologia do ambiente virtual. Este módulo é encarregado de compilar informações diversas sobre o terreno, transformando parâmetros abstratos em uma representação poligonal concreta do terreno. Ele desempenha funções como a interpretação de ruído para a modelagem de camadas do terreno e a integração de diferentes tipos de solo. A [Figura 4.3](#) ilustra, em termos de desenvolvimento da aplicação, o diagrama de classes UML (*Unified Modelling Language*) específico para este módulo, destacando as entidades-chave e suas relações.

Analizando o diagrama de classe fornecido para a aplicação de geração de terreno, buscou-se compor uma organização lógica e uma separação clara de responsabilidades entre as entidades. A classe `TerrainGenerator` contém uma instância de `Layermap` (mapa de camada). Ela utiliza o `Layermap` para estabelecer os parâmetros iniciais do terreno, como tamanho da grade e lista de tipos de solo representado pela classe `SoilType`. Além disso, esta classe é responsável por inicializar o terreno através do método `Generate` e pela criação e atualização da malha gráfica correspondente.

Na classe `Layermap`, a propriedade `data` constitui-se de um vetor de seções de solo do tipo `Section`, traduzindo a grade do terreno. O vetor é unidimensional, no entanto, é capaz de mapear o índice de acesso comportando-se como uma matriz (a ser discutido na Seção 4.2). A classe referente ao mapa de camada ainda gerencia acesso e manipulação espaciais das seções com métodos como `AddAt`, `GetAt` e `PopAt`, e interage com o gerador de ruído da classe `Noise` para criar mapas de ruído. Esta última cumpre o papel de definir características como altura e inclinação do terreno.

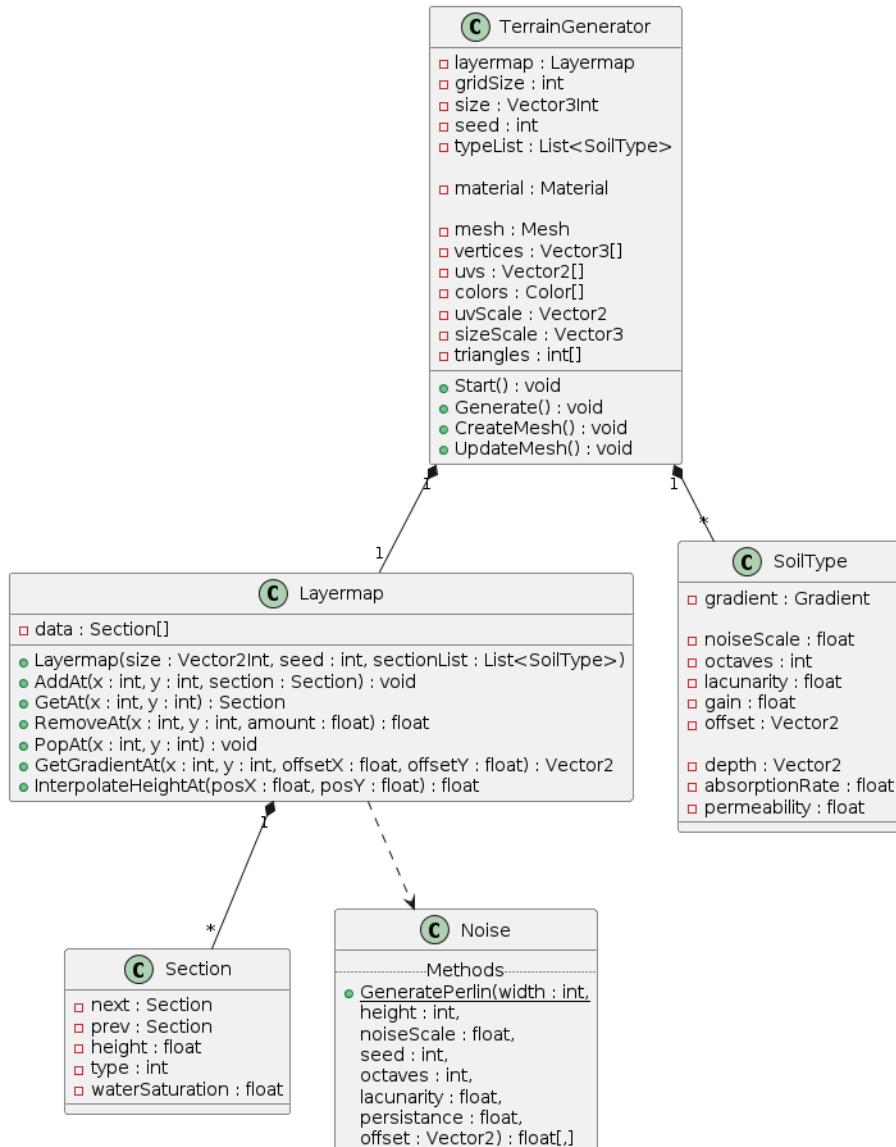


Figura 4.3 – Diagrama de classes do módulo de geração.

As classes **Section** e **SoilType** são designadas para a representação dos dados do terreno. A primeira encapsula uma seção individual de solo com atributos que mantêm o rastreamento das seções conectadas (**next**, **prev**), a altura (**height**), o índice do tipo de solo (**type**) e o nível de saturação de água (**waterSaturation**). Já a classe **SoilType** contém propriedades específicas de cada tipo de solo, como a taxa de absorção (**absorptionRate**) e permeabilidade (**permeability**), que definem como cada tipo de solo responde aos processos de erosão e absorção.

A [Figura 4.4](#), com seu diagrama de sequência, fornece uma visão estruturada dos processos interativos dentro da aplicação de geração de terreno elaborada. Esta disposição exibe um fluxo de mensagens desde o momento em que o usuário interage com a interface gráfica de usuário (*GUI*) até a renderização final do terreno, passando pela geração de camadas e a associação de ruídos para formar a topografia. Percebem-se ainda

que os diferentes componentes do sistema se comunicam e colaboram para realizar a tarefa de gerar e visualizar os terrenos.

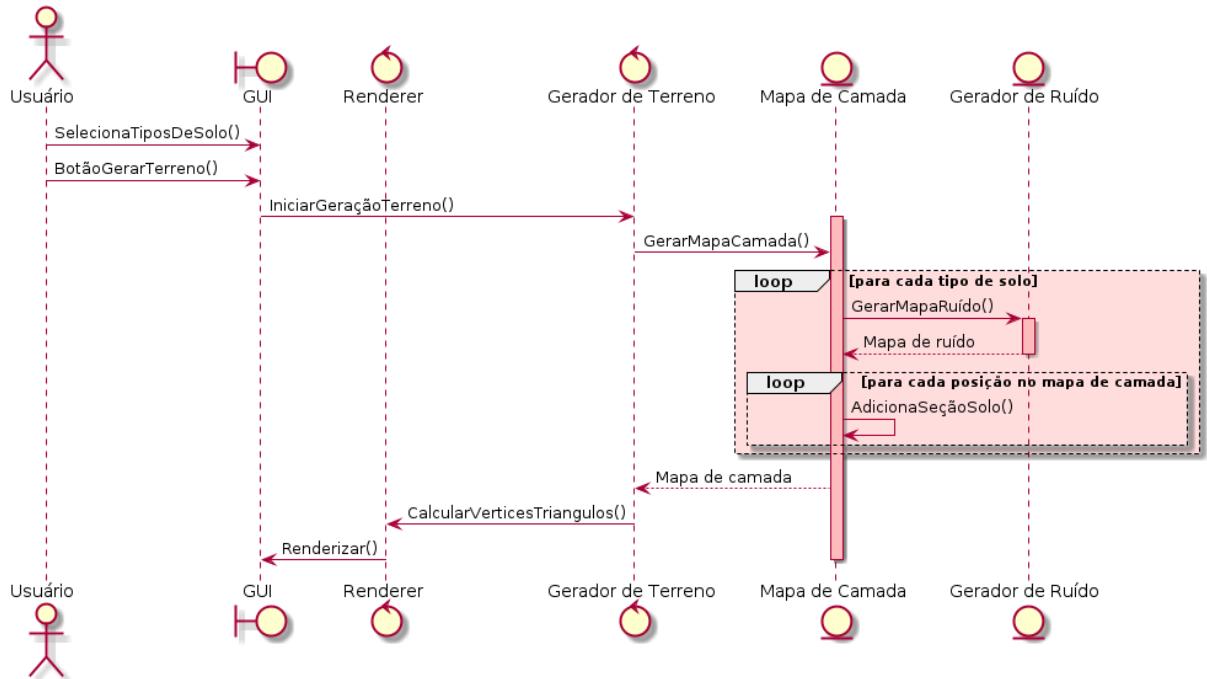


Figura 4.4 – Diagrama de sequência do processo integral da geração de terrenos.

A interação começa com o usuário fornecendo *input* por meio da interface gráfica de usuário, o que desencadeia a geração de terreno. Este componente, por sua vez, inicia a geração do mapa de camada, baseada na lista de tipos de solos obtida na interação com o usuário.

Dentro do processo de geração do mapa de camada, existem loops aninhados: o primeiro percorre cada tipo de solo para criar mapas de ruído específicos, enquanto o segundo e terceiro iteram sobre cada posição no mapa de camada para adicionar seções de solo. Essas seções são alocadas com base nos dados gerados pelo gerador de ruído. Uma vez que o mapa de camada é criado, o gerador de terreno calcula os vértices e triângulos necessários para a sua renderização. Este fluxo reflete uma comunicação estreita entre os componentes da geração de terreno e a visualização, destacando a necessidade da estruturação de dados.

4.2 Módulo de Erosão

O módulo de simulação corresponde, no contexto da aplicação, aos agentes e algoritmos que simulam processos naturais que influenciam a formação e a evolução do relevo. O método de erosão hidráulica proposto no Capítulo 3 foi implementado levando-se em conta a extensibilidade para outras abordagens. As funções incluídas abrangem a

inicialização do processo erosivo, a criação e movimentação de partículas sobre o terreno, a avaliação de quando e onde ocorrerá a deposição ou erosão de sedimentos, e a atualização contínua das características do terreno, como altura e inclinação, além da evaporação da água, que culmina na alteração da paisagem virtual.

O módulo de erosão, como visto na [Figura 4.1](#), é composto por três principais componentes: **Propriedades da Erosão**, **Erosão Hidráulica** e **Partícula**. Estes componentes interagem entre si e com o módulo de geração para simular o processo de erosão do terreno.

O componente **Propriedades da Erosão** define os parâmetros e características que influenciam a erosão do solo, como a capacidade de carga de sedimentos, a taxa de erosão e os fatores que afetam a dispersão de partículas. Estas propriedades são imprescindíveis para configurar a simulação de erosão de acordo com o tipo de solo e as condições ambientais especificadas no módulo de geração.

A **Erosão Hidráulica** é o objeto central do processo simulado de movimentação de sedimentos realizado pela água. Este componente utiliza as **Propriedades da Erosão** para determinar como a água interage com o solo, removendo e depositando material ao longo do tempo.

O componente **Partícula** representa cada unidade de água que interage com o terreno. Cada partícula tem atributos como posição, velocidade, volume de água e quantidade de sedimentos carregados. As partículas são geradas e manipuladas pela **Erosão Hidráulica**, e suas interações com o terreno são baseadas nas propriedades da erosão.

O módulo de erosão interage externamente com o módulo de geração através do mapa de camada, que fornece a estrutura do terreno a ser erodido. O gerador de terreno cria o mapa de camada utilizando o gerador de ruído, e a erosão hidráulica modifica este mapa ao longo do tempo, simulando o impacto contínuo da erosão hídrica no terreno.

A simulação começa com a inicialização do processo de erosão, onde cada iteração constitui a geração de uma nova gota d'água e contribui para a evolução progressiva do terreno. A [Figura 4.5](#) exibe o sequenciamento lógico das ações da gota, esquematizando seu ciclo de vida na simulação de erosão.



Figura 4.5 – Diagrama de atividades do algoritmo de erosão hidráulica.

Inicialmente, uma gota é gerada em uma posição aleatória sobre o mapa de camada. Para cada ciclo de vida dessa gota, o algoritmo calcula sua posição na célula. O vetor gradiente da posição é usado para atualizar a direção da gota e calcular a altura interpolada em sua nova posição. Caso a gota permaneça dentro do mapa, o processo segue para calcular a diferença de altura que, por sua vez, influencia se a gota irá depositar ou erodir sedimentos.

Adicionalmente, após a verificação de limite do mapa, ocorre uma etapa de absorção de água pelo solo, onde a saturação hídrica da seção de solo em contato com a gota é ajustada. Esta etapa é relevante para simular adequadamente o impacto da água no

terreno e na erosão subsequente. O processo de erosão ou deposição é decidido com base na capacidade de sedimentos da gota. Se a gota estiver sobrecarregada de sedimentos ou estiver se elevando, depositará parte dos sedimentos. Caso contrário, a gota irá erodir o terreno no raio de alcance, retirando material do solo e aumentando sua carga de sedimentos. A quantidade de material erodido é ponderada pela saturação hídrica da seção de solo e pelo peso predefinido.

Durante a atualização da velocidade e do volume de água da gota, o algoritmo considera a velocidade atual da gota e a evaporação ocorrida. A velocidade é reduzida à medida que a gota perde volume de água, simulando a evaporação natural. Este ciclo continua até que o tempo de vida da gota seja alcançado ou saia dos limites do mapa, encerrando, assim, o processo de erosão para aquela gota específica. O processo é repetido para cada nova gota gerada, acumulando o efeito da erosão e deposição no terreno ao longo do tempo.

4.3 Desenvolvimento da Aplicação

Um simulador geomorfológico para terrenos procedurais foi elaborado com o propósito de representar os processos de erosão pluvial e consequente deposição de sedimentos. Uma de suas vantagens reside na estrutura de dados em camadas ([MCDONALD, 2022](#)), que possibilita a modelagem de eventos erosivos sobre o solo e camadas subjacentes. Além disso, a estrutura comporta múltiplos tipos de solo, traduzindo a heterogeneidade das paisagens naturais.

A aplicação foi desenvolvida com o motor gráfico de desenvolvimento de jogos *Unity* e escrita na linguagem C#. O motor fornece uma base sólida para construção de aplicações gráficas, ajustando-se aos propósitos deste trabalho.

4.3.1 Estruturas de Dados

No presente projeto, o objetivo das estruturas de dados é modelar a complexidade dos ambientes naturais. As estruturas de seção de solo (**Section**), tipo de solo (**SoilType**) e mapa de camada (**Layermap**) constituem a base de dados sobre as quais os algoritmos de erosão operam. Essas estruturas contribuem para a precisão do modelo de terreno, atuando como fundamentos para a simulação de processos erosivos. Elas permitem a execução de algoritmos que consideram as características distintas de cada tipo de solo, facilitando a representação da dinâmica do terreno e o impacto da erosão ao longo do tempo.

A classe de mapa de camada expõe uma interface simples com métodos para acessar, adicionar e remover as seções de solo. No [Código 4.1](#) pode-se verificar o método para obter uma seção de solo situada na superfície de uma posição arbitrária do mapa.

```

1  public Section GetAt(int x, int y) {
2      return data[x + size.x * y];
3  }

```

Código 4.1: Método de busca de seção de solo em uma posição (x, y)

A expressão $[x + size.x * y]$ é uma fórmula matemática para mapear coordenadas 2D (x, y) para um índice 1D dentro de um vetor, permitindo que o vetor `data`, que é unidimensional, simule o comportamento de uma matriz. Para adicionar uma seção de solo na estrutura de dados em camadas, adota-se o método exemplificado no [Código 4.2](#).

```

1  public void AddAt(int x, int y, Section section) {
2      int index = x + size.x * y;
3
4      if (data[index] == null) {
5          data[index] = section;
6          return;
7      }
8
9      var currentSection = data[index];
10
11     if (currentSection != null && currentSection.height > section.height) {
12         section.next = currentSection;
13
14         if (currentSection.prev != null) {
15             section.prev = currentSection.prev;
16             section.prev.next = section;
17         }
18
19         currentSection.prev = section;
20         return;
21     }
22
23     currentSection.next = section;
24     section.prev = currentSection;
25
26     data[index] = section;
27 }

```

Código 4.2: Método de adição de uma seção de solo a uma posição do mapa.

O método se inicia com uma avaliação da posição destinada à nova seção. Se esta posição estiver vazia, a nova seção é diretamente inserida. Em casos onde já existe uma seção, a comparação de alturas determina a inserção da nova seção acima ou abaixo da atual, reajustando as referências de encadeamento para manter a ordem. A altura superior

permanece no topo, facilitando o acesso e mantendo a organização das camadas. Se a nova seção não estiver em uma posição mais baixa, ela é posta acima da atual, e as referências de ligação são atualizadas. Esta inserção mantém a coesão da estrutura do terreno e, portanto, novas seções se integram de forma ordenada.

O [Código 4.3](#) apresenta o método que é utilizado para remover a seção de solo mais superficial de uma posição específica no mapa de camada.

```

1  public void PopAt(int x, int y) {
2      var removeSection = GetAt(x, y);
3
4      if (removeSection.prev != null) {
5          removeSection.prev.next = removeSection.next;
6          data[x + size.x * y] = removeSection.prev;
7      }
8  }
```

Código 4.3: Método de remoção da seção de solo de uma pilha.

Este procedimento remove uma seção de solo superficial do terreno como parte do processo de simulação de erosão. Ao ser invocado, o método identifica a seção de solo na posição especificada e, se houver uma seção anteriormente ligada, ajusta as ligações da pilha, reposicionando a referência de topo para a próxima seção abaixo. Com isto, a seção de solo é removida da pilha e a estrutura de dados permanece íntegra.

A classe de mapa de camada também conta com métodos avançados para a análise do terreno, tais como os de interpolação de gradiente e altura. Estes métodos favorecem o movimento da partícula e a inclinação em posições específicas do terreno e são utilizados na simulação de erosão.

4.4 Ruído e Geração de Camadas

O ruído e o algoritmo de composição de camadas estão intrinsecamente interligados, uma vez que cada camada é formada através de um mapa de ruído. Este mapa é utilizado como base para a construção detalhada do terreno, determinando a elevação e a disposição do solo. Ao sobrepor essas camadas, o algoritmo de composição se encarrega da interseção de características, propiciando uma transição natural entre os diferentes tipos de solo. Cada camada sucessiva introduz características topográficas próprias que contribuem para a diversidade da paisagem.

O cálculo do mapa de ruído é uma parte integrante da geração de terreno, onde para cada ponto do mapa é atribuída uma altura. A iteração é feita através de dois laços aninhados que percorrem o mapa nas coordenadas (x, y). O [Código 4.4](#) demonstra o cálculo

para uma oitava de ruído. Esta abordagem de uma oitava, aplicada consistentemente, já é capaz de criar variação topográfica no terreno, sendo que o algoritmo completo envolve iterações adicionais para incorporar o efeito acumulativo de múltiplas oitavas, conferindo ao terreno características fractais. Cada contribuição é incorporada para formar a altura final do ponto, considerando as variações de amplitude e frequência determinadas, respectivamente, pelos parâmetros de persistência e lacunaridade.

```

1  for (int x = 0; x < width; x++) {
2      for (int y = 0; y < height; y++) {
3          float sample = 0;
4
5          float freq = 1f;
6          float amp = 1f;
7          float noiseHeight = 0f;
8
9          float xCoord = (float)x / scale * freq;
10         float yCoord = (float)y / scale * freq;
11
12         for (int k = 0; k < octaves; k++) {
13             sample = Mathf.PerlinNoise(xCoord + offset.x + seed, yCoord +
14                                         ↵ offset.y + seed);
15
16             noiseHeight += sample * amp;
17
18             freq *= lacunarity;
19             amp *= gain;
20         }
21
22         noisemap[x, y] = noiseHeight;
23     }
}

```

Código 4.4: Lógica de repetição para gerar valores de ruído de uma oitava.

O procedimento para compor camadas é detalhado no [Código 4.5](#), onde uma função da classe de mapa de camada é inicializada com uma série de tipos de solo. Cada tipo na série possui parâmetros únicos para a geração de ruído que influenciam a aparência da camada resultante. Durante o processo, um mapa de ruído é criado para cada tipo de solo, utilizando estes parâmetros para estabelecer a altura da camada. Com base nesse mapa de ruído, e através da interpolação linear, valores de altura são definidos para cada ponto do mapa conforme os limites de profundidade do tipo de solo em questão.

```

1  for (int k = 0; k < soilList.Count; k++) {
2      var fraction = (float)k / soilList.Count;
3
4      var noisemap = Noise.GeneratePerlin(
5          size.x,
6          size.y,
7          soilList[k].noiseScale,
8          seed + (int)(fraction * MAX_SEED),
9          soilList[k].octaves,
10         soilList[k].lacunarity,
11         soilList[k].gain,
12         soilList[k].offset);
13
14     for (int x = 0; x < size.x; x++) {
15         for (int y = 0; y < size.y; y++) {
16             var noiseValue = noisemap[x, y];
17             float height = Mathf.Lerp(soilList[k].depth.x, soilList[k].depth.y,
18                                         noiseValue);
19
20             var section = new Section
21             {
22                 height = height,
23                 type = k,
24             };
25
26             AddAt(x, y, section);
27         }
28     }

```

Código 4.5: Algoritmo de composição de camadas.

A função de adicionar uma seção de solo, discutida na Subseção 4.3.1, é responsável por posicionar cada seção no mapa de acordo com a altura determinada. Este método permite a organização das camadas de solo de maneira que as seções mais elevadas fiquem visíveis, enquanto as inferiores são dispostas abaixo. A estrutura de dados é projetada para ajustar-se automaticamente com o uso de ponteiros que conectam cada seção, permitindo que a simulação reflita mudanças topográficas devido a eventos de erosão e deposição.

A [Figura 4.6](#) representa, de maneira exemplar, a composição de três tipos de solo em camadas. Observa-se o processo dinâmico e sequencial de construção de um terreno tridimensional por meio da técnica de sobreposição de camadas. As setas coloridas representam as diferentes fases sobre a estrutura do terreno.

A seta azul simboliza a adição progressiva de novas camadas de solo da lista predefinida de tipos de solos, indicando como cada nova camada é construída com base nas características específicas de seu tipo de solo correspondente. Por sua vez, a seta

alaranjada horizontal demonstra a iteração ao longo do mapa de camada, um processo no qual as seções de solo são adicionadas em cada posição específica do mapa. Este processo é repetido para cada tipo de solo na lista, criando uma composição de camadas que se sobrepõem e interagem para formar o relevo final do terreno. A seta alaranjada, portanto, destaca a aplicação repetida deste método através de toda a extensão do mapa, ou seja, cada ponto é analisado e a topografia resultante reflete a integração das propriedades de cada camada de solo adicionada.

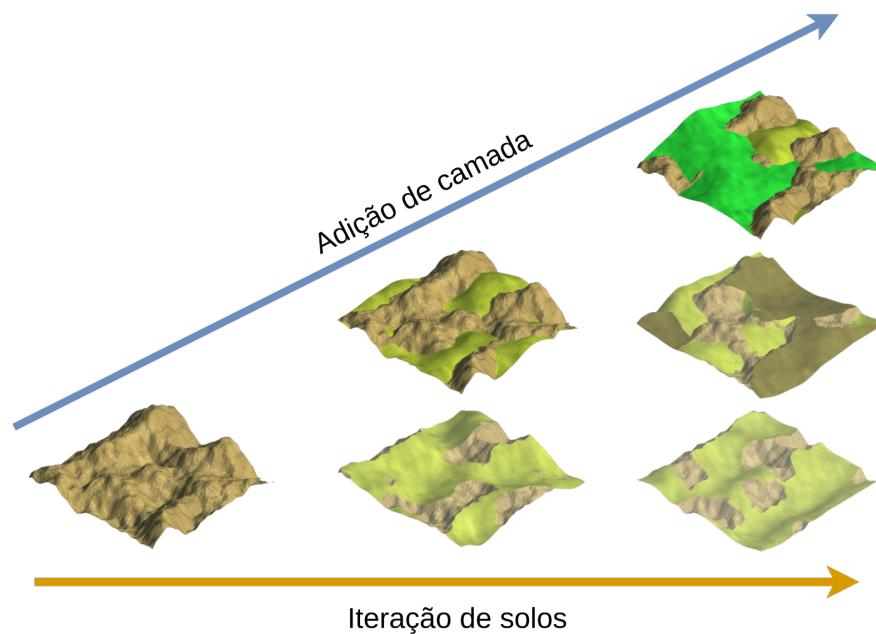


Figura 4.6 – Sequência evolutiva da formação de terreno em camadas.

4.4.1 Algoritmo de Erosão Hidráulica

Propõe-se, nesta Subseção, estabelecer as principais etapas que compõem o algoritmo em foco.

Inicialmente, os pesos de impacto de erosão são computados. No trecho visto no [Código 4.6](#) o algoritmo itera sobre cada ponto do mapa de camada para obter as distâncias quadráticas de um ponto central dentro do raio definido. Com base nessas distâncias, pesos são atribuídos de maneira que pontos mais próximos têm maior influência – uma abordagem que reflete a forma como a água provocaria o desgaste do solo mais fortemente nas proximidades do ponto de impacto.

```

1  for (int y = -radius; y <= radius; y++) {
2      for (int x = -radius; x <= radius; x++) {
3          float sqrDst = Mathf.Sqrt(x * x + y * y);
4
5          if (sqrDst < radius * radius) {
6              int coordX = centreX + x;
7              int coordY = centreY + y;
8
9              if (coordX >= 0 && coordX < gridSize && coordY >= 0 && coordY < gridSize)
10             → {
11                  float weight = 1 - sqrDst / radius;
12                  weightSum += weight;
13                  weights[addIndex] = weight;
14                  xOffsets[addIndex] = x;
15                  yOffsets[addIndex] = y;
16                  addIndex++;
17             }
18         }
19     }

```

Código 4.6: Cálculo de pesos de erosão para cada posição do mapa.

Em seguida à inicialização de pesos para o processo de erosão, o algoritmo de simulação hidráulica inicia a criação e o ciclo de vida das gotas d' água, como se pode verificar no [Código 4.7](#). Para cada uma das gotas, definidas por um número preestabelecido, o sistema gera uma posição aleatória dentro dos limites da grade de simulação. A posição da gota determina o percurso durante a simulação de erosão e, por consequência, as áreas que sofrerão o impacto de seu deslocamento.

```

1  for (int i = 0; i < droplets; ++i) {
2      var droplet = new Droplet
3      {
4          position = new Vector2(UnityEngine.Random.Range(0, gridSize - 1),
5                           UnityEngine.Random.Range(0, gridSize - 1)),
6          StartLifecycle(droplet)
7      }
8  }

```

Código 4.7: Inicialização do processo de erosão hidráulica.

Conforme demonstrado no [Código 4.8](#), o ciclo perdura por um tempo de vida variado para cada partícula de água. Neste estágio, a posição da partícula é definida por valores reais correspondentes às suas coordenadas no eixo horizontal e vertical de uma

célula da grade. A partir destes valores, são calculados os deslocamentos em relação à sua posição atual, que serão empregados para determinar o gradiente do terreno onde a partícula se encontra, assim como a altura interpolada do solo em sua localização exata.

Após o cálculo do gradiente e da altura, a direção da partícula é atualizada. Esta atualização é uma combinação da direção anterior e do gradiente recentemente calculado, ponderada por um coeficiente que define a tendência da partícula em continuar seu curso ou seguir o declive do terreno. A direção resultante é então normalizada para garantir que a partícula se desloque com um vetor de unidade. Finalmente, a nova posição é determinada pela adição do vetor de direção à posição atual, movendo a partícula para o próximo ponto no ciclo de seu trajeto pelo terreno.

```

1  for (int lifetime = 0; lifetime < maxDropletLifetime; lifetime++) {
2      int x = (int)Mathf.Floor(droplet.position.x);
3      int y = (int)Mathf.Floor(droplet.position.y);
4
5      float offsetX = droplet.position.x - Mathf.Floor(droplet.position.x);
6      float offsetY = droplet.position.y - Mathf.Floor(droplet.position.y);
7
8      Vector2 gradient = layermap.GetGradientAt(x, y, offsetX, offsetY);
9      float interpolatedHeight = layermap.InterpolateHeightAt(x, y, offsetX,
10      → offsetY);
11
12      droplet.direction = droplet.direction * inertia - gradient * (1f -
13      → inertia);
14      droplet.direction.Normalize();
15  }
```

Código 4.8: Ciclo de vida e movimentação da partícula.

Após a movimentação, interpola-se a altura da nova posição e calcula-se a variação da mesma de acordo com a posição anterior. A variação de altura é positiva quando a partícula se move para cima e negativa quando se move para um ponto mais baixo. A partir deste cálculo de inclinação, determina-se a capacidade de carga de sedimentos da partícula. Esta capacidade é diretamente proporcional à magnitude da variação de altura negativa, à velocidade da partícula e ao volume de água que carrega, com um limite mínimo estabelecido a fim de que apresente potencial para carregar sedimentos.

O [Código 4.9](#) exprime esta dinâmica:

```

1 float newInterpolatedHeight =
  ↵ layermap.InterpolateHeightAt(droplet.position.x,
  ↵ droplet.position.y);

2
3 float deltaHeight = newInterpolatedHeight - interpolatedHeight;

4
5 bool isMovingUphill = deltaHeight > 0;

6
7 droplet.sedimentCapacity = Mathf.Max(-deltaHeight * droplet.speed *
  ↵ droplet.waterVolume, minSedimentCapacity);

8
9 if (droplet.sediment > droplet.sedimentCapacity || isMovingUphill) {
10   // Depositar sedimentos
11 } else {
12   // Erodir o solo
13 }
```

Código 4.9: Determinação da inclinação e da capacidade máxima de sedimentos.

A [Código 4.9](#) expõe ainda o segmento de código que determina se haverá deposição ou erosão, com base na capacidade calculada e na quantidade atual de sedimentos transportados pela partícula. Se estiver transportando mais sedimentos do que sua capacidade ou se estiver transitando por uma elevação, ocorrerá, então, a deposição dos sedimentos nas seções da célula atual. Caso contrário, se a capacidade de carga de sedimentos não for excedida, a partícula atuará na erosão do terreno, removendo sedimentos da superfície, adicionando-os à sua própria carga. Este processo é cíclico e continua até que a vida útil da partícula termine ou ela saia dos limites do mapa.

O processo de deposição de sedimentos é influenciado pela posição da partícula em relação à célula da grade e à sua capacidade de transportar sedimentos. Conforme se vê no [Código 4.10](#), a quantidade de deposição é determinada pelo menor valor entre a inclinação e a quantidade de sedimento que a partícula está transportando, caso ela esteja se deslocando para cima. Do contrário, a deposição é baseada no excedente de sedimentos, multiplicado pelo fator de velocidade de deposição.

```

1 float amountToDeposit = isMovingUphill ?
2     Mathf.Min(deltaHeight, droplet.sediment) :
3     (droplet.sediment - droplet.sedimentCapacity) * depositSpeed;
4
5 Deposit(x, y, offsetX, offsetY, amountToDeposit);
6 droplet.sediment -= amountToDeposit;
7
8 // Restante da implementação...
9 private void Deposit(int x, int y, float offsetX, float offsetY, float
10    ↪ amountToDeposit)
11 {
12     var topLeftSection = layermap.GetAt(x, y);
13     var topRightSection = layermap.GetAt(x + 1, y);
14     var bottomLeftSection = layermap.GetAt(x, y + 1);
15     var bottomRightSection = layermap.GetAt(x + 1, y + 1);
16
17     topLeftSection.height += amountToDeposit * (1f - offsetX) * (1f - offsetY);
18     topRightSection.height += amountToDeposit * offsetX * (1f - offsetY);
19     bottomLeftSection.height += amountToDeposit * (1f - offsetX) * offsetY;
20     bottomRightSection.height += amountToDeposit * offsetX * offsetY;
21 }

```

Código 4.10: Algoritmo de deposição de sedimentos.

A fim de proporcionar que os sedimentos sejam distribuídos naturalmente, utiliza-se a interpolação bilinear baseada na posição fracionária da partícula dentro de uma célula da grade. Os sedimentos são adicionados aos quatro pontos da célula, ponderados pelos desvios horizontal e vertical da partícula dentro da mesma. Este procedimento simula o espalhamento natural de sedimentos em ambientes reais, onde materiais como areia e lama se acumulam e se nivelam gradativamente sobre o terreno.

O algoritmo de erosão, tal como é observado no [Código 4.11](#), simula a remoção de material do solo com base na capacidade de carga de sedimentos de uma partícula de água e na dinâmica do terreno. A quantidade de material erodido é calculada como o menor valor entre a diferença necessária para preencher a capacidade de sedimentos da partícula, ajustada pela taxa de velocidade de erosão, e a mudança negativa de altura do terreno, que indica um movimento descendente da partícula. Este cálculo garante que a erosão não exceda o volume que a partícula pode transportar nem distorça a aparência real do terreno.

```

1  float amountToErode = Mathf.Min((droplet.sedimentCapacity - droplet.sediment) *
2    ↵ erosionSpeed, -deltaHeight);
3
4  for (int brushPointIndex = 0; brushPointIndex <
5    ↵ erosionIndices[dropletIndex].Length; brushPointIndex++) {
6    int index = erosionIndices[dropletIndex][brushPointIndex];
7    var section = layermap.GetAt(index % gridSize, index / gridSize);
8    float weighedErodeAmount = amountToErode * section.waterSaturation *
9      ↵ erosionWeights[dropletIndex][brushPointIndex];
10
11   float deltaSediment = (section.height < weighedErodeAmount) ?
12     ↵ section.height : weighedErodeAmount;
13   section.height -= deltaSediment;
14   droplet.sediment += deltaSediment;
15
16 }

```

Código 4.11: Algoritmo do processo de erosão.

Durante a execução do processo de erosão, esta é aplicada de forma distribuída dentro de um raio predefinido ao redor da posição da partícula. Para cada ponto dentro deste raio, o volume de solo a ser erodido é regulado pelo peso calculado, que é baseado na distância do ponto ao centro onde a partícula se encontra. Este peso é também modificado pela saturação de água do solo, indicando que solos mais úmidos podem ter uma resistência diferente à erosão.

Durante o processo de erosão, se a altura de uma seção de solo for reduzida para um nível inferior ao de uma seção adjacente na pilha, esta seção superior será removida. A gestão das camadas de solo através deste mecanismo permite que as camadas inferiores possam emergir, o que favorece a dinâmica de erosão e deposição em ambientes simulados.

5 Resultados e Discussão

Este capítulo apresenta as implicações do método proposto para a geração de terrenos com simulação de erosão. Seu objetivo é analisar os dados gerados a partir das simulações realizadas, nas quais as características dos terrenos são exploradas sob a influência de diferentes parâmetros de erosão e composição de camadas. Através da avaliação desses resultados, busca-se não apenas verificar a conformidade do modelo com os fenômenos físicos que ele pretende simular, mas também identificar possíveis melhorias e ajustes no algoritmo.

A discussão aqui apresentada visa correlacionar os resultados obtidos com a hipótese inicialmente proposta, avaliando a aplicabilidade do modelo em cenários práticos de modelagem de terrenos. Configurações de simulação são comparadas, demonstrando como variações nos tipos de solo, na intensidade e no tipo de erosão na morfologia do terreno simulado. Esta análise é fundamental para entender as limitações do modelo proposto e para sugerir direções futuras de pesquisa que possam otimizar a precisão e a eficiência das simulações realizadas.

Os cenários de teste, relativos aos resultados expostos neste capítulo, foram executados em um computador com processador AMD Ryzen 5-5500U de até 4 GHz, placa gráfica integrada AMD Radeon Graphics e 20GB de memória RAM, enquanto os algoritmos de geração e erosão foram processados na CPU. Utilizou-se a ferramenta denominada *Profiler*, integrada ao motor do *Unity*, para coletar tempos de execução nas chamadas de função dos algoritmos.

Um dos aspectos pertinentes à geração do terreno diz respeito ao tamanho da grade, que se refere à resolução espacial dos pontos de dados (no caso deste trabalho, as seções de solo) usados para modelar a superfície do terreno. Conforme ilustrado na [Figura 5.1](#) é evidente como variações no tamanho da grade impactam o nível de detalhe (LOD, do inglês *Level of Detail*) e a precisão da topografia simulada.

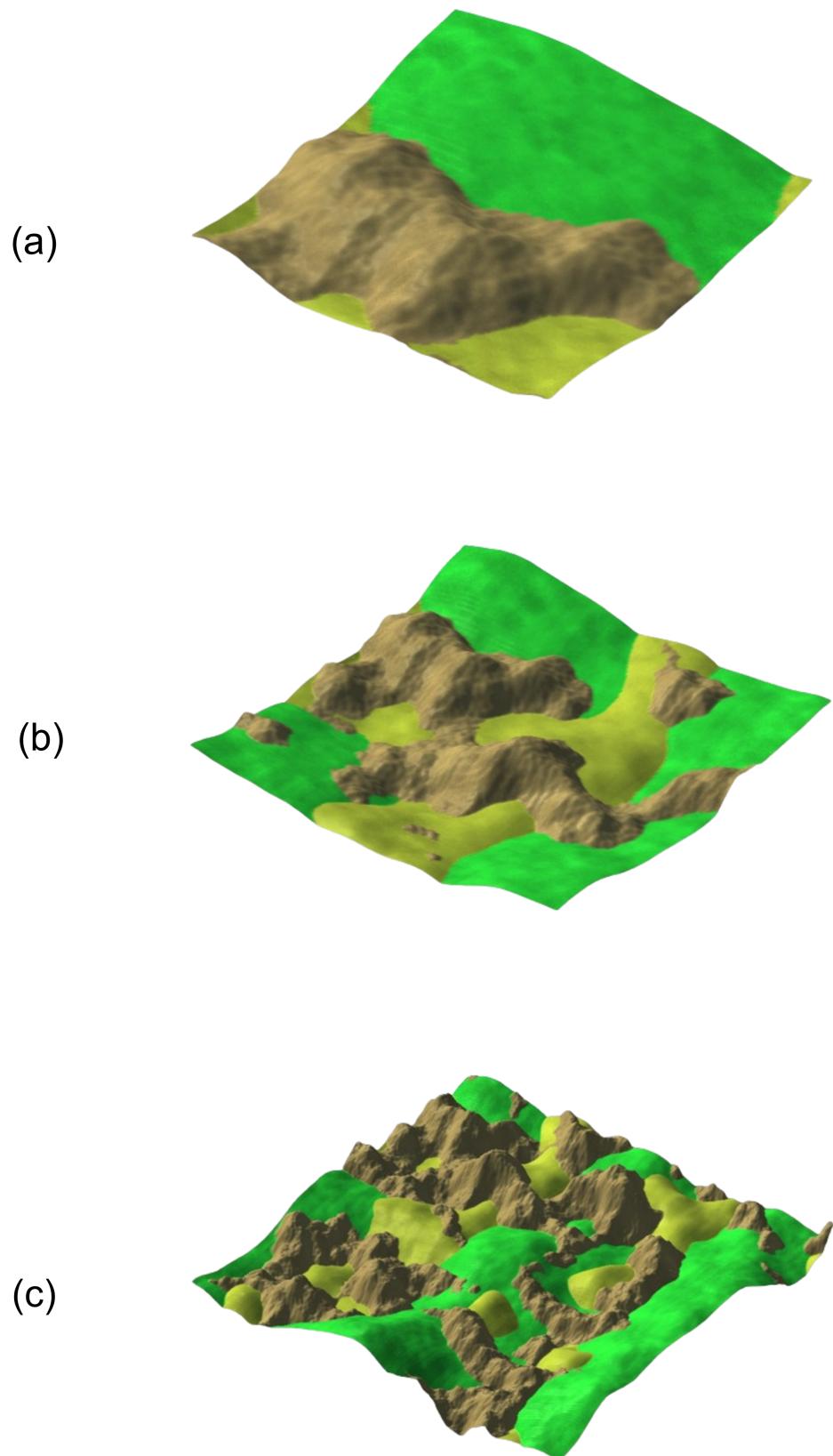


Figura 5.1 – Influência do tamanho da grade; terreno (a) com tamanho de 128x128, (b) 256x256 e (c) 512x512.

Uma grade com maior resolução implica em um maior número de pontos a serem processados, o que resulta em detalhes mais finos no terreno. Por outro lado, isto também aumenta a carga computacional, especialmente na etapa de renderização, onde a extensão da grade pode aumentar significativamente o consumo de recursos computacionais, dependendo da escala deste aumento.

A [Tabela 5.1](#) apresenta uma sistematização dos parâmetros utilizados para definir diferentes tipos de solo em cenários de teste na simulação de terrenos. Cada tipo de solo — Pedregoso, Argiloso, Arenoso e Terra Vegetal — possui características únicas em termos de profundidade, permeabilidade e taxa de absorção. Estes parâmetros são utilizados para modelar as propriedades físicas do solo que influenciam tanto a geração visual do terreno quanto a dinâmica de interações como erosão e deposição. Os tipos de solo referidos também possuem parâmetros de ruído fractal distintos. Os valores de lacunaridade, persistência e a escala de ruído ajustam a granularidade da distribuição espacial das características do solo, enquanto as oitavas de ruído determinam a complexidade da textura de cada tipo de solo, o que contribui para a fidelidade visual do terreno.

Parâmetros Tipos de Solo	Intervalo de Profundidade	Permeabilidade	Taxa de absorção	Oitavas de ruído	Lacunaridade	Persistência	Escala de ruído
Pedregoso	[0; 0,6]	0,4	0,3	6	4	0,15	60
Argiloso	[0; 0,6]	0,1	0,2	6	2	0,5	70
Arenoso	[0; 0,7]	1	0,7	5	2	0,2	100
Terra Vegetal	[0; 1]	0,2	0,9	2	4	0,1	200

Tabela 5.1 – Tipos de solo definidos para os cenários de teste

As definições de cada solo refletem suas propriedades naturais. O solo pedregoso, por exemplo, possui uma permeabilidade moderada e uma baixa taxa de absorção, indicando uma superfície mais resistente à erosão e mudanças de forma, condizentes com a representação de áreas rochosas. Por outro lado, a terra vegetal, com alta taxa de absorção, cria um terreno que responde mais visivelmente aos agentes ambientais como água e vento, simulando camadas de solo mais suscetíveis à erosão. Observa-se, na [Figura 5.2](#), uma ilustração de um terreno gerado com os quatro tipos de solo.

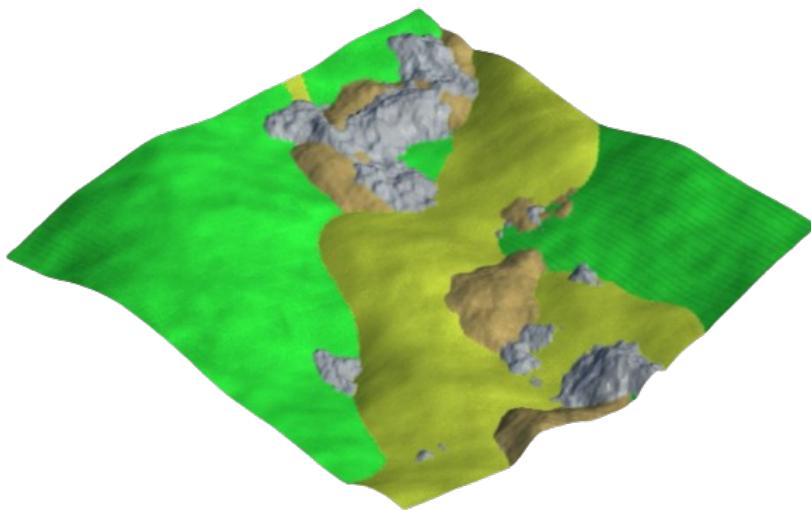


Figura 5.2 – Exemplo de terreno gerado com 4 camadas.

Nos testes realizados, estabeleceram-se cenários para avaliar a influência combinada entre o tamanho da grade e a quantidade de camadas na simulação de terreno. Os tamanhos de grade testados foram 128x128, 256x256 e 512x512, referidos, a partir de então, como G128, G256 e G512, respectivamente. Para a quantidade de camadas, utilizaram-se 2, 3 e 4 camadas, designadas como C2, C3 e C4. Estes cenários permitiram verificar como diferentes resoluções de grade e complexidades de camadas impactam a geração e modificação do terreno durante o processo de erosão.

Na configuração C2, os tipos de solo escolhidos foram o solo pedregoso e a terra vegetal. Esta combinação foi escolhida para observar como solos com características extremas de permeabilidade e absorção interagem no processo erosivo. Na simulação com C3, incluíram-se os tipos de solo pedregoso, arenoso e terra vegetal, aumentando a diversidade de propriedades do solo e, consequentemente, a complexidade da simulação. Este cenário visa analisar a transição e interação entre camadas de solo com diferentes características físicas e suas respostas frente à erosão. Ao adicionar o tipo argiloso como C4, aumentou-se ainda mais a complexidade do cenário.

A comparação entre estes diferentes cenários permitiu a identificação de padrões e comportamentos específicos na dinâmica de erosão e deposição, dependendo do tipo e da quantidade de solo, além de destacar como a resolução da grade afeta a precisão e o detalhamento do terreno simulado. Tal abordagem possibilitou uma análise dos fatores que influenciam a modelagem de terrenos e a eficácia do algoritmo desenvolvido.

O gráfico, representado na [Figura 5.3](#), apresenta os resultados de tempo médio de 10 execuções do algoritmo de geração de terreno para as nove combinações entre tamanho da grade (G128, G256 e G512) e quantidade de camadas (C2, C3 e C4). A análise dos tempos de execução permite verificar a influência dessas variáveis na performance do

algoritmo.

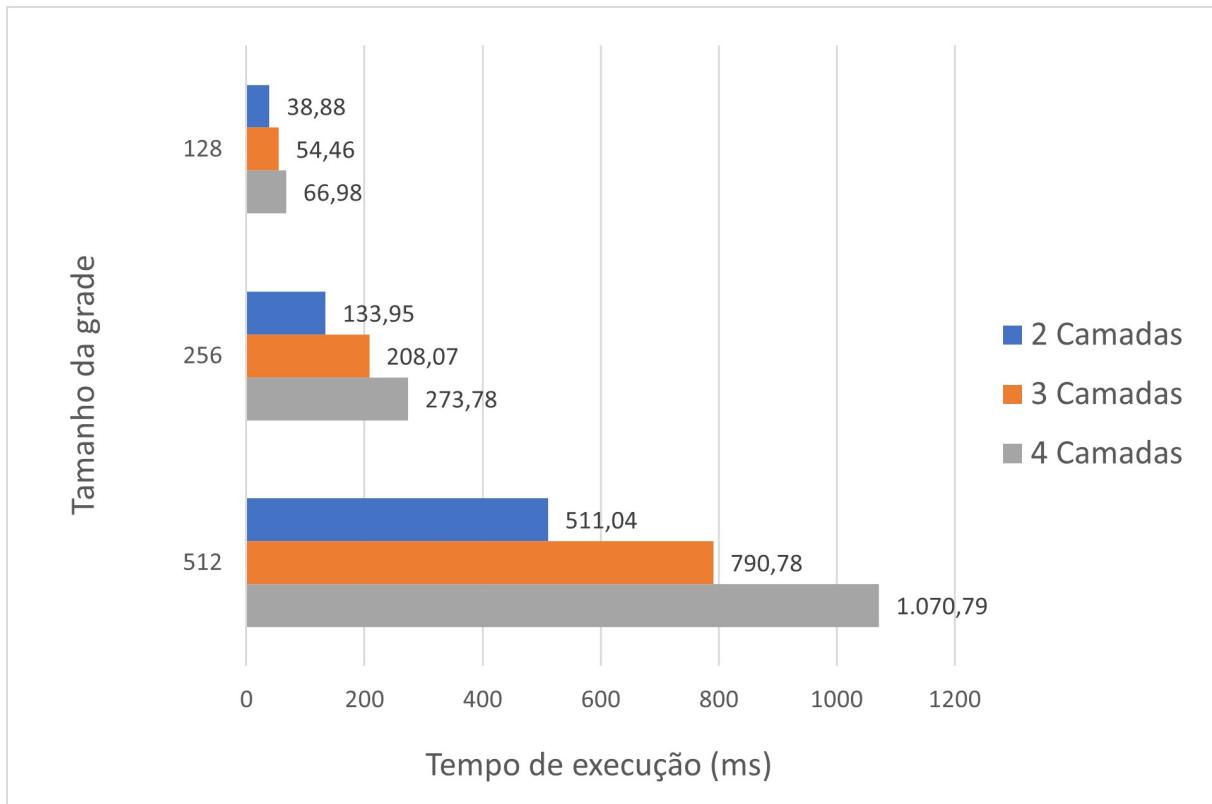


Figura 5.3 – Gráfico com médias de tempo de execução para combinações entre diferentes tamanhos de grade e quantidade de camadas do terreno.

Os resultados do gráfico (Figura 5.3) ilustram como o tempo de execução do algoritmo de geração de terreno aumenta com o tamanho da grade e a quantidade de camadas. De acordo com Benes e Forsbach (2001), a complexidade teórica do algoritmo pode ser expressa como $O(kn^2)$, onde k é a quantidade de camadas e n é o tamanho da grade. Isto implica que ao dobrar o tamanho da grade, o tempo de execução tende a quadruplicar, enquanto o aumento no número de camadas adiciona uma sobrecarga linear ao tempo de execução. A Figura 5.4 mostra uma visualização gráfica dos tempos de execução do algoritmo de geração de terreno em função do tamanho da grade, variando entre 128 e 512, para diferentes números de camadas (2, 3 e 4). Adotou-se o ajuste de regressão polinomial de grau 2 sobre os dados de tempo de execução ao se analisar a proximidade dos valores obtidos em relação à complexidade teórica do algoritmo.

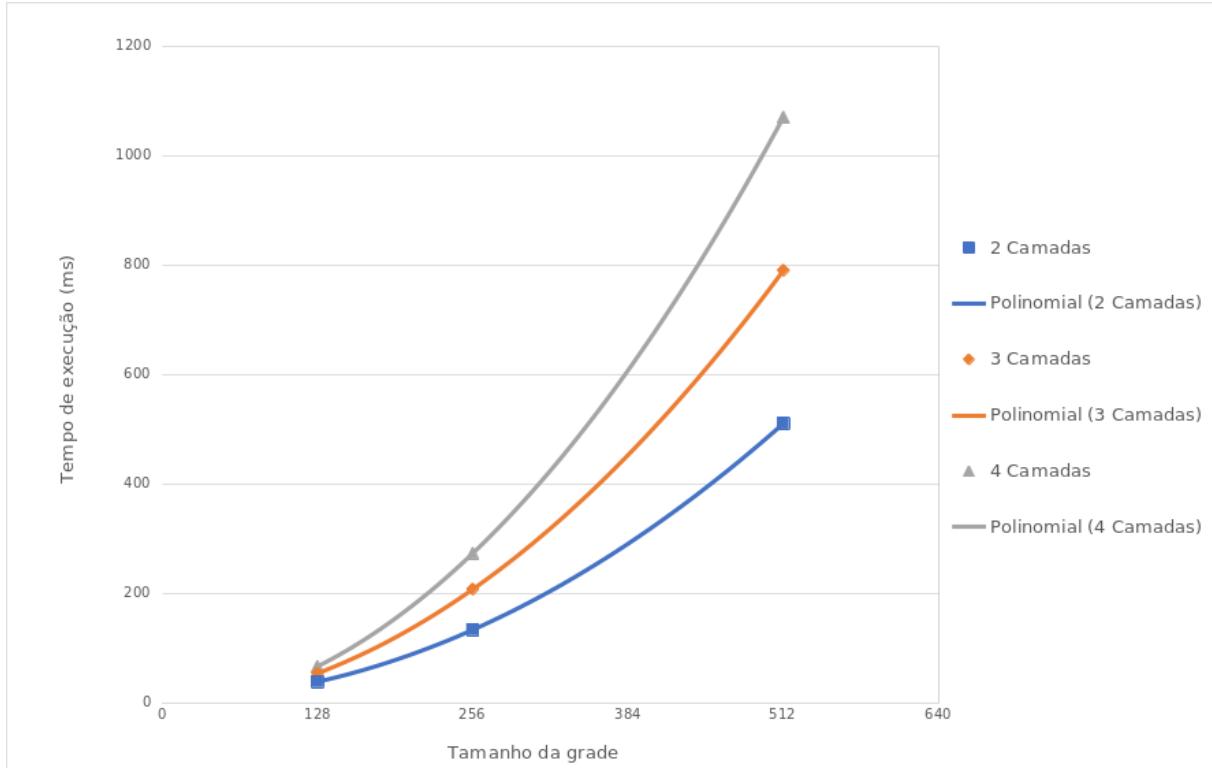


Figura 5.4 – Gráfico de dispersão e regressão polinomial de grau 2 dos valores de tempos de execução referentes aos tamanhos da grade.

Tomando como exemplo o cenário de 2 camadas, realiza-se a seguir a regressão polinomial de grau 2 utilizando os valores experimentais. O processo implica em ajustar os dados ao modelo da forma $y = ax + bx + c$ onde y representa o tempo de execução e x é o tamanho da grade. Os pontos de dados são: (128, 38.881), (256, 133.952), (512, 511.037). Aplicando o método de mínimos quadrados através da ferramenta *LibreOffice Calc*, obtém-se a equação de regressão dada por $0.0019x + 0.0125x + 6.1233$ e o coeficiente de determinação $R^2 = 1$. Este coeficiente indica a proporção da variabilidade nos dados de tempo de execução que é explicada pela equação de regressão ajustada, sugerindo que o modelo de regressão se conforma plenamente com os dados observados. Tais resultados indicam que, para o caso de 2 camadas, o tempo de execução aumenta aproximadamente com o quadrado do tamanho da grade, conforme previsto pela complexidade teórica. Ao estender esta análise para os casos de 3 e 4 camadas, o coeficiente R^2 também atinge o índice 1, corroborando o resultado anterior.

O gráfico da Figura 5.5 ilustra o tempo que o algoritmo de criação de terreno leva para ser executado em relação ao número de camadas. A análise é realizada utilizando regressão linear para cada um dos tamanhos de grade testados, com o objetivo de investigar a relação linear entre a quantidade de camadas e o tempo de execução.

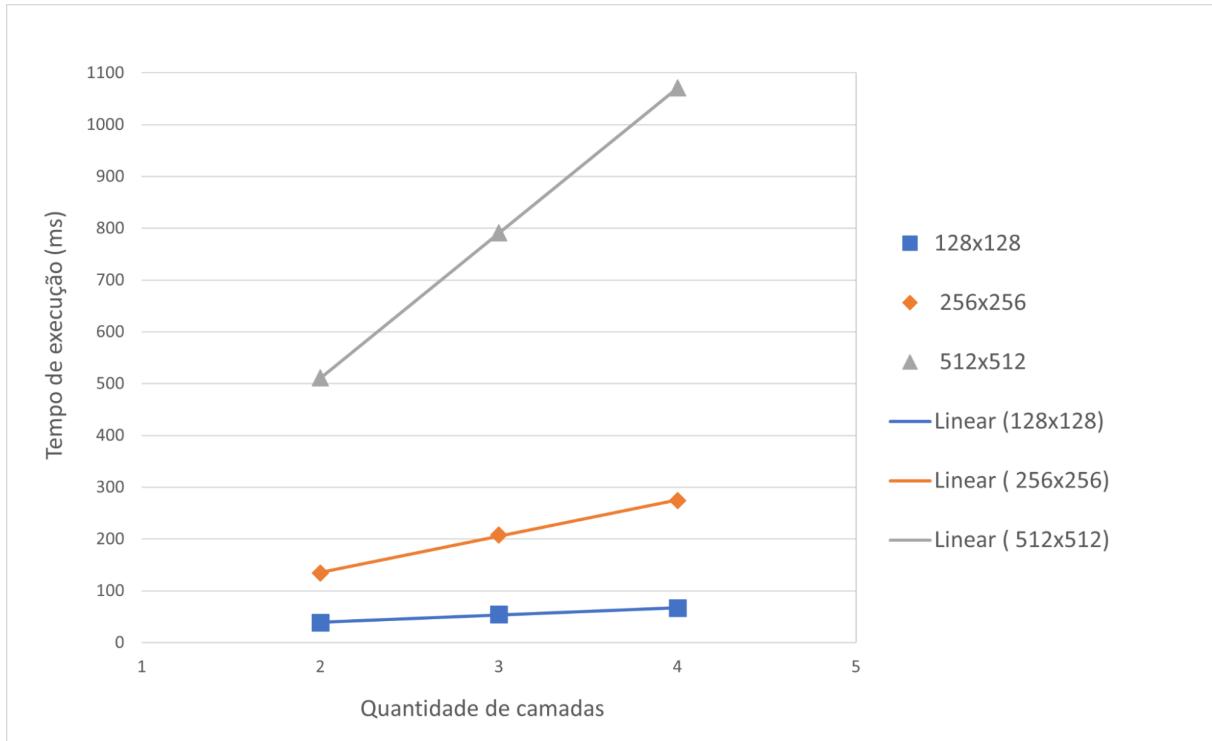


Figura 5.5 – Gráfico de dispersão e regressão linear dos valores de tempo de execução referentes à quantidade de camadas.

Observando os resultados, verifica-se que para o tamanho de grade 128x128, a relação linear entre a quantidade de camadas e o tempo de execução é perceptível, com um coeficiente de determinação $R^2 = 0,996$. Para o tamanho de grade 256x256, essa relação também se mantém forte, com $R^2 = 0,998$; já para a grade 512x512, o coeficiente de determinação obtido é $R^2 = 1$, apontando novamente uma relação linear significativa.

Portanto, a análise dos dados indica que o comportamento do tempo de execução alinha-se à complexidade teórica. O aumento do tamanho da grade resulta em um crescimento significativo no tempo de execução, ao passo que a quantidade de camadas revela um crescimento modesto. Além disso, operações de interpolação e cálculo de oitavas de ruído, para cada ponto da grade, impactam o tempo de execução. Em aplicações práticas, estes resultados destacam a importância de considerar a resolução do terreno e a complexidade da composição do solo para manter a eficiência computacional.

Para o algoritmo de erosão implementado neste trabalho, correlacionou-se o tamanho da grade e a duração de simulação como variáveis de análise. A escolha destas variáveis se justifica pela possibilidade de se avaliar a eficácia do algoritmo em diferentes condições de simulação. O tamanho da grade, conforme verificado na análise do algoritmo de geração, influencia diretamente a complexidade computacional. A duração de simulação, medido em iterações de partículas de água, afeta a dinâmica do processo de erosão, determinando a quantidade de sedimentos transportados e depositados ao longo do terreno. A paisagem

da [Figura 5.6](#) representa o terreno ainda não impactado pela erosão, enquanto as três paisagens mostradas na [Figura 5.7](#) traduzem, respectivamente:

- 50k - precipitação de curta duração, com 50 mil gotas durante a simulação;
- 100k - precipitação de duração moderada, com 100 mil gotas durante a simulação;
- 200k - precipitação de longa duração, com 200 mil gotas durante a simulação.

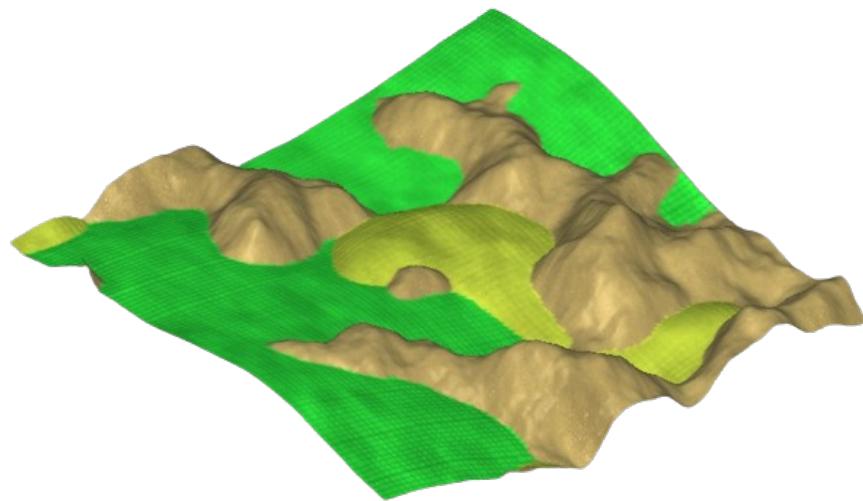


Figura 5.6 – Terreno gerado anteriormente ao processo de erosão.

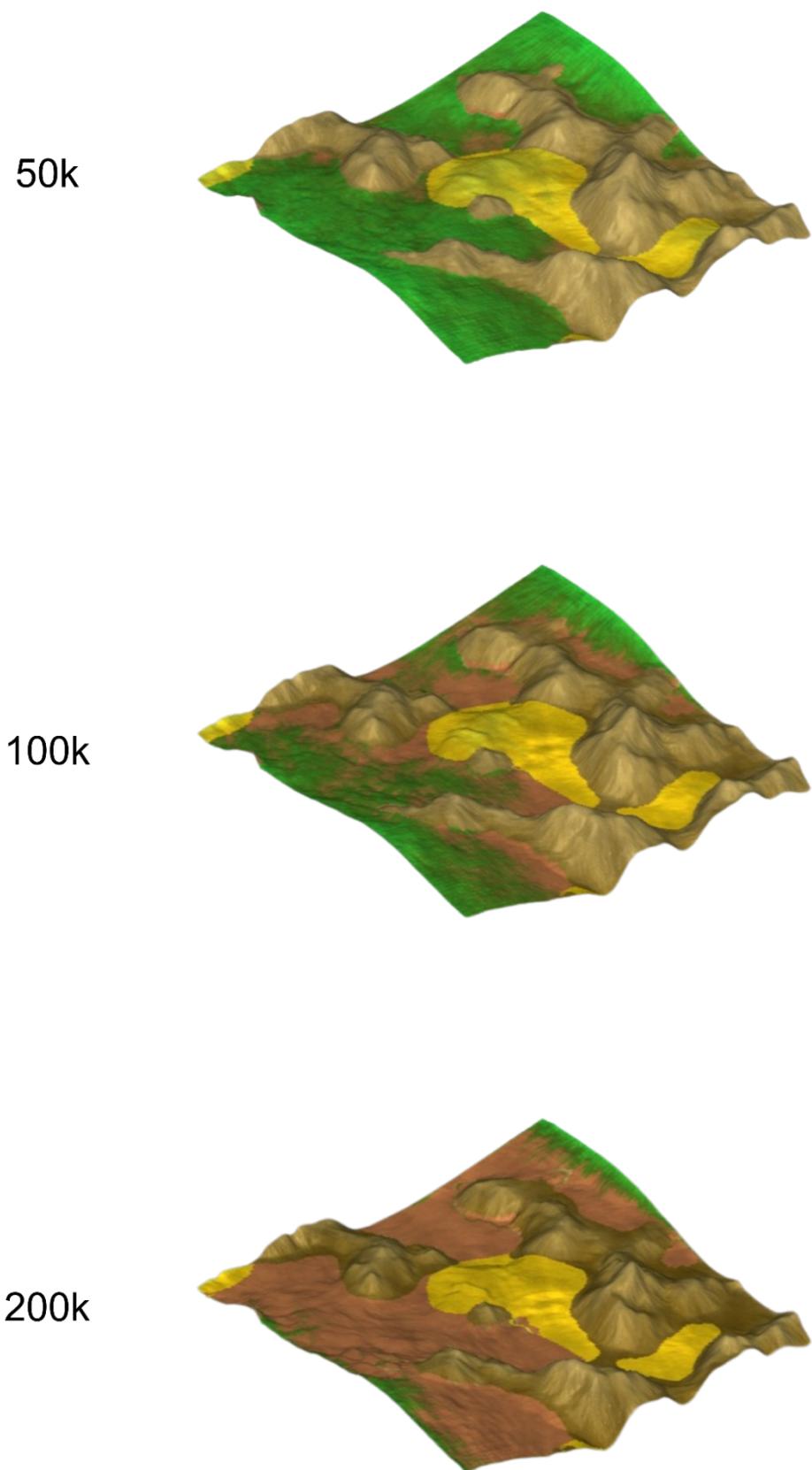


Figura 5.7 – Sucessão de impactos erosivos segundo a duração de simulação.

Para os testes deste algoritmo de erosão consideraram-se os parâmetros executados nos experimentos de [Beyer \(2015\)](#), que são:

- Raio de influência da erosão K_r igual a 4 km;
- Taxa de velocidade de erosão K_e igual a 0,7;
- Taxa de velocidade de deposição K_d igual a 0,2;
- Constante de inércia do movimento K_n igual a 0,3;
- Taxa de evaporação K_v igual a 0,02;
- Capacidade máxima de sedimentos Q igual a 8;
- Constante de inclinação mínima K_s igual a 0,01;
- Constante do fenômeno gravitacional K_g igual a 10;
- Constante de alcance máximo da partícula K_a igual a 64.

A metodologia de avaliação do algoritmo de erosão seguiu o mesmo procedimento adotado para a geração do terreno, isto é, o de obter a média de dez tempos de execução coletados através da combinação de duas variáveis: o tamanho da grade e a duração da simulação. Na composição da [Tabela 5.2](#) estão registradas as médias em foco.

Duração de Simulação (ms)	Tamanho da Grade		
	128x128	256x256	512x512
50k	2978,47	4849	6285,65
100k	6054,48	9531,01	12.205,78
200k	12.166,16	18.983,01	24.012,91

Tabela 5.2 – Médias do tempo de execução (em milissegundos) do algoritmo de erosão correspondentes às combinações entre tamanho da grade e duração de simulação.

Uma dedução possível fornecida pelos dados do [Tabela 5.2](#) é a de que a duração da simulação possui complexidade algorítmica maior do que a extensão de área receptora da precipitação. A comprovação disto reside na comparação entre o cruzamento das células 128x128 e 200k, com tempo de execução igual a 12.166,16 com o cruzamento das células 512x512 e 50k, com tempo de execução igual a 6285,65, que é aproximadamente a metade de 12.166,16.

A Figura 5.8 ilustra a relação entre o tempo de execução do algoritmo de erosão e a duração da simulação, expressa em milhares de iterações (50k, 100k e 200k), para três diferentes tamanhos de grade (128x128, 256x256 e 512x512).

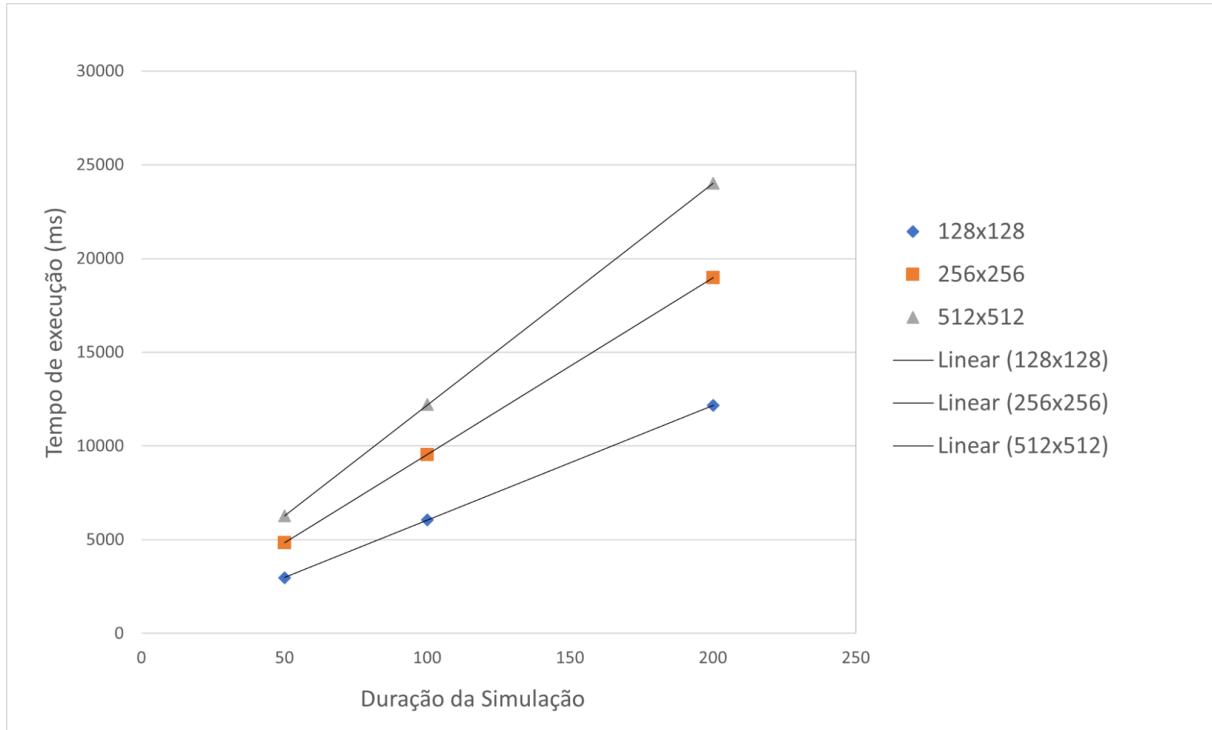


Figura 5.8 – Gráfico de dispersão e regressão linear dos valores de tempo de execução relativos à duração de simulação.

Observa-se que, à proporção que a duração da simulação aumenta, o tempo de execução cresce de modo aproximadamente linear para cada um dos tamanhos de grade.

Os pontos de dados para cada tamanho de grade foram ajustados por uma regressão linear, como indicado pelas linhas retas nos gráficos. Por exemplo, para a grade de 128x128, os tempos de execução registrados foram aproximadamente 2978 ms para 50k iterações, e 6054 ms para 100k iterações, resultando em uma linha de tendência que sugere um comportamento linear.

De forma similar, as grades de 256x256 e 512x512 também demonstram um comportamento quase linear nos tempos de execução em relação à duração da simulação. A proximidade das linhas de regressão aos pontos de dados, com coeficientes de determinação R^2 próximos de 1, reforça a linearidade observada, sugerindo que o tempo de execução é proporcional à duração da simulação para cada tamanho de grade.

O algoritmo de erosão realiza simulações em um laço de repetição com um número fixo de iterações S , como 50k, 100k e 200k, onde cada iteração representa uma partícula de água movendo-se pelo terreno, causando erosão e deposição de sedimentos. Tais considerações reforçam a tendência dos dados a descreverem um comportamento linear, pois

o tempo de complexidade, na perspectiva do algoritmo, aumenta proporcionalmente ao número de iterações S .

Do ponto de vista da evolução do tempo de execução em relação ao tamanho da grade, nota-se que o gráfico da Figura 5.9 exibe uma regressão linear para cada conjunto de dados, representado pelas linhas ajustadas. É necessário dizer que o número de camadas que compõem o terreno exerce insignificante influência no tempo de execução do algoritmo, uma vez que somente a camada superficial interage com o agente erosivo. Entretanto, o processo de infiltração de água pode acarretar em maiores tempos de execução mediante a quantidade de camadas.

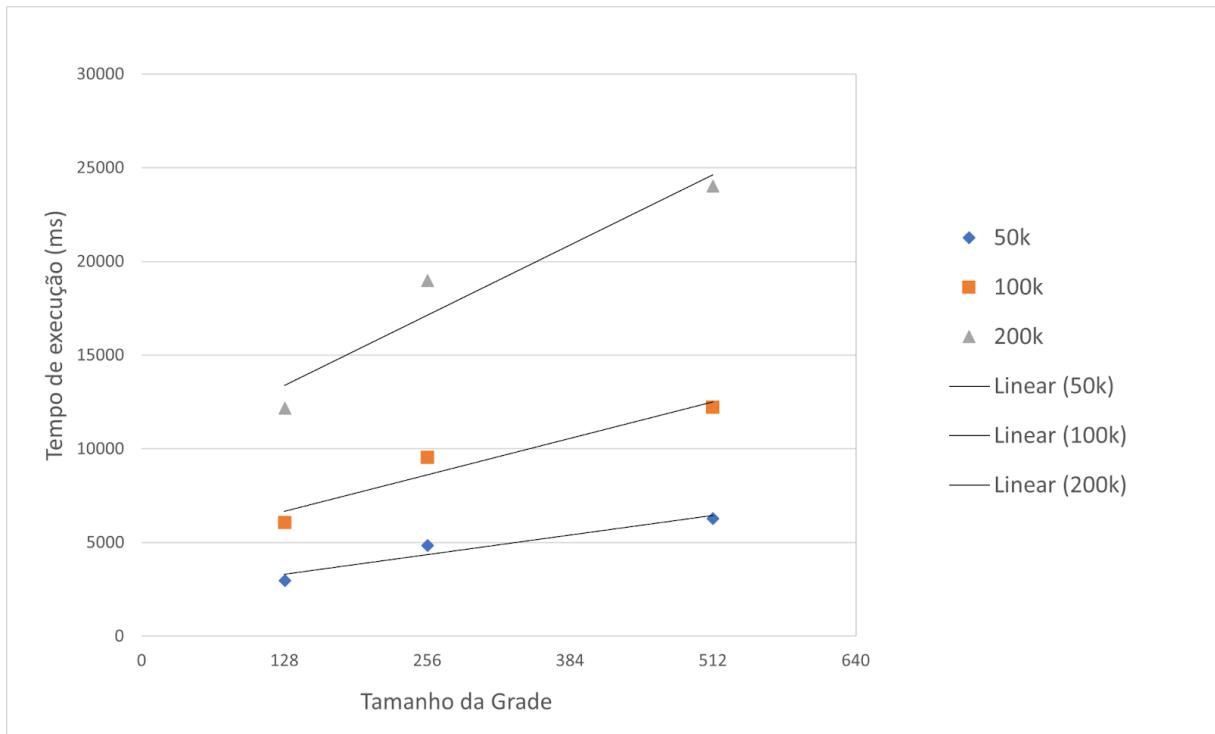


Figura 5.9 – Gráfico de dispersão e regressão linear dos valores de tempo de execução relativos ao tamanho da grade.

Ao efetuar o cálculo do coeficiente de determinação para as regressões, verifica-se que o valor não é elevado, indicando que a variabilidade dos dados não é completamente explicada pelo modelo linear. Ainda assim, o ajuste linear foi considerado o mais adequado uma vez que os resultados de tempo de execução não revelam tendência quadrática. Isto é corroborado ao se analisar as etapas do algoritmo: ao se gerar uma gota, seu tempo de vida corresponde ao alcance máximo determinado K_a e, portanto, não equivale à duração de simulação ou número de iterações S . Ao mesmo tempo, não constitui uma complexidade logarítmica, dado que não são empregadas, por exemplo, estratégias de dividir para conquistar.

Uma possível explicação para essa dispersão dos dados a partir da estimativa linear

é o comportamento das gotas durante a simulação de erosão. Como as gotas podem sair do mapa durante a simulação, tal fato introduz uma variabilidade adicional nos tempos de execução, uma vez que a área efetivamente erodida pode variar. Além disso, o alcance máximo das gotas K_a foi mantido constante para todos os três valores de tamanho da grade, o que também pode contribuir para a dispersão dos dados, já que em uma grade extensa as gotas têm mais espaço para se mover e, potencialmente, menor chance de sair do mapa. Em vista disso, apesar de uma tendência linear ser observada, estes fatores intrínsecos ao algoritmo influenciam a precisão da regressão linear, resultando em coeficientes de determinação mais baixos.

A análise técnica dos resultados de geração e erosão foi detalhada com base em métricas quantitativas. No entanto, para fornecer uma compreensão mais completa dos efeitos dos algoritmos aplicados, é preciso complementar essa análise com uma avaliação visual das alterações no terreno. As Figuras 5.10, 5.11 e 5.12, apresentadas a seguir, ilustram diferentes cenários de terrenos gerados e suas subsequentes transformações após a erosão, oferecendo uma perspectiva das mudanças morfológicas que ocorrem. As imagens permitirão observar texturas, formas e distribuições de tipos de solo, complementando a análise técnica.

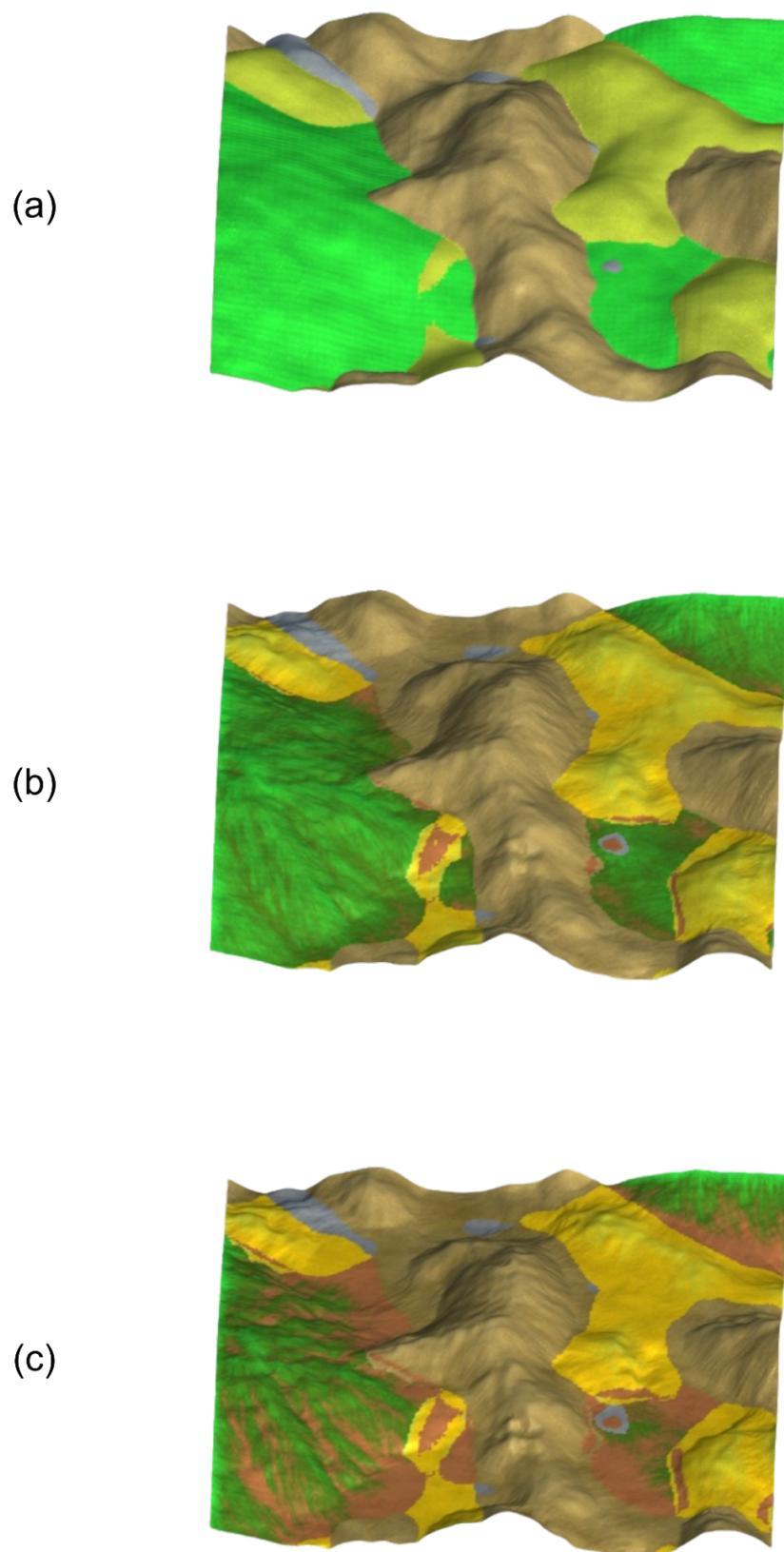


Figura 5.10 – Exemplo de terreno com 256x256 e 4 camadas: (a) terreno original; (b) simulação erosiva de 100 mil gotas; (c) simulação erosiva de 200 mil gotas.

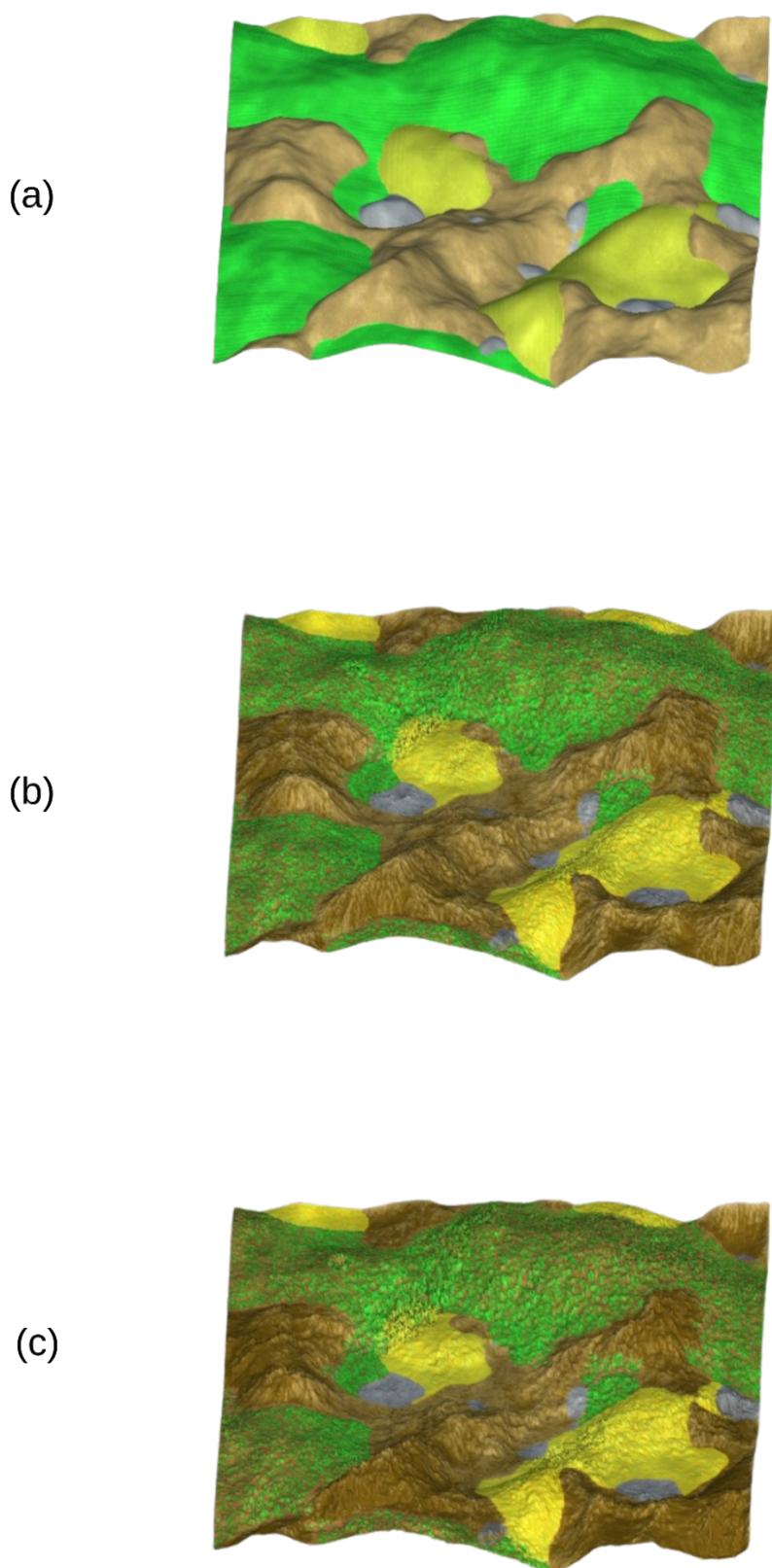


Figura 5.11 – Exemplo de terreno com 364x364 e 4 camadas: (a) terreno original; (b) simulação erosiva de 120 mil gotas; (c) simulação erosiva de 240 mil gotas.

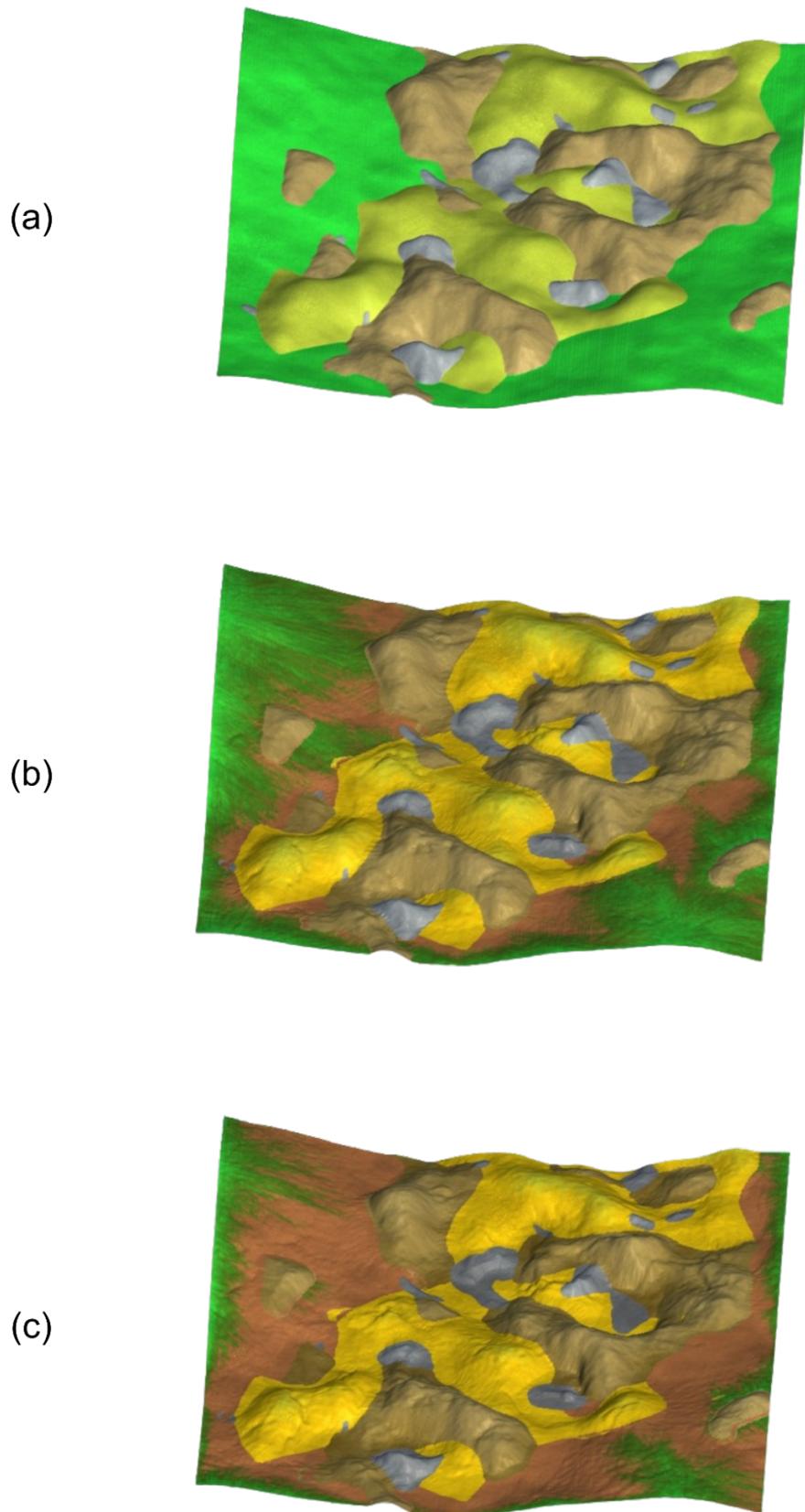


Figura 5.12 – Exemplo de terreno com 512x512 e 4 camadas: (a) terreno original; (b) simulação erosiva de 150 mil gotas; (c) simulação erosiva de 300 mil gotas.

Os terrenos exibidos em seus estados iniciais são caracterizados por diferentes tipos de solo dispostos de maneira variada, refletindo a aplicação do algoritmo de geração de terrenos em camadas. A disposição irregular das camadas superficiais mostra como o algoritmo pode simular processos dinâmicos do solo, resultando em terrenos que possuem características topográficas variadas.

Nos terrenos afetados pelo processo erosivo, observa-se que a erosão de sedimentos é mais pronunciada nas áreas elevadas e íngremes. Esta condição reflete a implementação do algoritmo de erosão que considera o gradiente na posição relativa da partícula em sua movimentação. Em contrapartida, áreas mais rebaixadas acumulam água, como visto na transição de cobertura vegetal para o húmus, assim como material sedimentar.

Áreas anteriormente elevadas exibem marcas de desgaste, enquanto as depressões acumulam sedimentos, criando um terreno visualmente mais irregular. As diferenças nas taxas de absorção e permeabilidade entre diferentes tipos de solo são evidenciadas pela variação na intensidade das transformações observadas. Além do tamanho da grade, parâmetros associados à erosão - como sua velocidade, raio de influência e taxa de evaporação - refletem consideravelmente nas alterações promovidas pelo fenômeno em questão.

Em síntese, as Figuras 5.10, 5.11 e 5.12 fornecem uma validação visual das mudanças induzidas pelos algoritmos de geração e erosão e permitem a compreensão dos processos simulados. A análise dos dados sugere que os parâmetros definidos para os tipos de solo e a configuração das camadas são adequados para gerar paisagens heterogêneas, destacando a viabilidade do modelo em capturar variações topográficas e dinâmicas de erosão de maneira satisfatória.

6 Considerações Finais

Neste trabalho, foi desenvolvido um sistema para geração procedural de terrenos com simulação de processos erosivos, em que se utilizou uma abordagem que integra múltiplas camadas de solo. O estudo explorou a aplicação de algoritmos de ruído, como o *Perlin Noise*, para a formação de camadas de terreno, e implementou um algoritmo de erosão hidráulica que simula o transporte e deposição de sedimentos pelas gotas d'água. A abordagem proposta demonstrou ser razoável na criação de paisagens dinâmicas, proporcionando uma representação mais fiel dos processos naturais de formação de terreno.

Os resultados obtidos foram analisados em termos de tempo de execução e plotados em gráficos de dispersão. A análise dos mesmos revelou uma complexidade algorítmica consistente com as expectativas teóricas, destacando a plausibilidade da implementação. Além disso, a comparação dos cenários de teste mostrou que a abordagem de múltiplas camadas proporciona uma maior fidelidade na representação dos fenômenos erosivos. Através de figuras ilustrativas, foi possível validar visualmente as mudanças induzidas pelos algoritmos de geração e erosão, permitindo uma melhor compreensão dos processos simulados.

Apesar dos aspectos positivos, o modelo desenvolvido apresentou algumas limitações. Uma delas diz respeito às discretas remoções de camadas superficiais, dado a baixa retirada de sedimentos nas áreas inferiores. Além disso, a abordagem atual não leva em conta fatores como a vegetação e a compactação do solo, que podem influenciar significativamente os processos erosivos. A inclusão destes fatores poderia proporcionar uma simulação ainda mais autêntica.

Para trabalhos futuros, recomendam-se as seguintes pesquisas complementares:

- Implementar os efeitos da compactação do solo, onde áreas compactadas têm menor permeabilidade e maior escoamento superficial, influenciando a distribuição de água;
- Aperfeiçoar o algoritmo de erosão objetivando um comportamento de fluidos baseado em Física, como visto em Št'ava et al. (2008);
- Otimizar os algoritmos de geração e erosão para execução em ambientes de computação paralela utilizando a GPU a fim de reduzir significativamente os tempos de processamento;
- Desenvolver um sistema de renderização volumétrica utilizando o algoritmo Marching Cubes (LORENSEN; CLINE, 1998). Apesar de trazer complexidade, proporciona deformações nas camadas e transições mais evidentes entre elas.

Referências

- ALONSO, J. A.; ARINYO, R. J.; ALBAJÉS, L. S. The grounded heightmap tree: A new data structure for terrain representation. 2008. Citado na página [18](#).
- ARCHER, T. Procedurally generating terrain. In: *44th annual midwest instruction and computing symposium, Duluth*. [S.l.: s.n.], 2011. p. 378–393. Citado na página [13](#).
- AZAD, S. et al. Mixed reality meets procedural content generation in video games. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. [S.l.: s.n.], 2016. v. 12, n. 2, p. 22–26. Citado na página [11](#).
- BENES, B.; FORSBACH, R. Layered data representation for visual simulation of terrain erosion. In: IEEE. *Proceedings Spring Conference on Computer Graphics*. [S.l.], 2001. p. 80–86. Citado 7 vezes nas páginas [12](#), [13](#), [19](#), [24](#), [27](#), [28](#) e [70](#).
- BEYER, H. T. Implementation of a method for hydraulic erosion. *Munich, Germany: Technische Universität München*, 2015. Citado 7 vezes nas páginas [26](#), [30](#), [39](#), [42](#), [43](#), [44](#) e [75](#).
- BOGGUS, M.; CRAWFIS, R. Procedural creation of 3d solution cave models. In: *Proceedings of the 20th IASTED International Conference on Modelling and Simulation*. [S.l.: s.n.], 2009. p. 180–186. Citado na página [18](#).
- DEMBOGURSKI, B. J. Geraçao procedural de terrenos através de extraçao de isosuperficies na gpu. 2009. Citado na página [21](#).
- DEY, R.; DOIG, J. G.; GATZIDIS, C. Procedural feature generation for volumetric terrains using voxel grammars. *Entertainment Computing*, Elsevier, v. 27, p. 128–136, 2018. Citado na página [11](#).
- EBERT, D. S. *Texturing & modeling: a procedural approach*. [S.l.]: Morgan Kaufmann, 2003. Citado 2 vezes nas páginas [16](#) e [21](#).
- FURTADO, H. *Procedural Generation of Volumetric Data for Terrain*. 2019. Citado 2 vezes nas páginas [19](#) e [30](#).
- GALIN, E. et al. A review of digital terrain modeling. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2019. v. 38, n. 2, p. 553–577. Citado 5 vezes nas páginas [17](#), [19](#), [24](#), [26](#) e [27](#).
- KOCA, Ç. *Representation, editing and real-time visualization of complex 3D terrains*. Dissertação (Mestrado) — Bilkent Universitesi (Turkey), 2012. Citado 2 vezes nas páginas [17](#) e [18](#).
- KOCA, Ç.; GÜDÜKBAY, U. A hybrid representation for modeling, interactive editing, and real-time visualization of terrains with volumetric features. *International Journal of Geographical Information Science*, Taylor & Francis, v. 28, n. 9, p. 1821–1847, 2014. Citado na página [49](#).

- LAINÉ, S.; KARRAS, T. Efficient sparse voxel octrees. In: *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. [S.l.: s.n.], 2010. p. 55–63. Citado na página 19.
- LÖFFLER, F.; MÜLLER, A.; SCHUMANN, H. Real-time rendering of stack-based terrains. In: VMV. [S.l.: s.n.], 2011. p. 161–168. Citado 2 vezes nas páginas 18 e 30.
- LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In: *Seminal graphics: pioneering efforts that shaped the field*. [S.l.: s.n.], 1998. p. 347–353. Citado 2 vezes nas páginas 18 e 83.
- MCDONALD. [S.l.]: An Efficient Data Structure for 3D Multi-Layer Terrain and Erosion Simulation, 2022. Disponível em <<https://nickmcd.me/2022/04/15/soilmachine/>>. Acesso em 21 de agosto 2023. Citado 6 vezes nas páginas 31, 32, 35, 36, 39 e 55.
- MEI, X.; DECAUDIN, P.; HU, B.-G. Fast hydraulic erosion simulation and visualization on gpu. In: IEEE. *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. [S.l.], 2007. p. 47–56. Citado 2 vezes nas páginas 30 e 31.
- MUSGRAVE, F. K.; KOLB, C. E.; MACE, R. S. The synthesis and rendering of eroded fractal terrains. *ACM Siggraph Computer Graphics*, ACM New York, NY, USA, v. 23, n. 3, p. 41–50, 1989. Citado 5 vezes nas páginas 12, 24, 26, 27 e 28.
- NADIG, S. M. *Example-Based Terrain Authoring with Complex Features*. Tese (Doutorado) — Purdue University, 2022. Citado na página 17.
- O'BRIEN, J. F.; HODGINS, J. K. Dynamic simulation of splashing fluids. In: IEEE. *Proceedings Computer Animation'95*. [S.l.], 1995. p. 198–205. Citado na página 31.
- OLSEN, J. Realtime procedural terrain generation. Citeseer, 2004. Citado na página 25.
- PERLIN, K. An image synthesizer. *ACM Siggraph Computer Graphics*, ACM New York, NY, USA, v. 19, n. 3, p. 287–296, 1985. Citado 3 vezes nas páginas 14, 22 e 27.
- PERLIN noise. [S.l.]: Wikimedia, 2023. Disponível em <https://en.wikipedia.org/wiki/Perlin_noise>. Acesso em 12 de fevereiro 2024. Citado na página 22.
- PEYTAVIE, A. et al. Arches: a framework for modeling complex terrains. In: WILEY ONLINE LIBRARY. *Computer graphics forum*. [S.l.], 2009. v. 28, n. 2, p. 457–467. Citado 5 vezes nas páginas 19, 28, 29, 30 e 50.
- PINTO, H. F. A. *Framework Para a Modelação Procedimental Integrada de Terrenos Completos*. Dissertação (Mestrado) — Instituto Politecnico do Porto (Portugal), 2017. Citado na página 19.
- ROSE, T. J.; BAKAOUKAS, A. G. Algorithms and approaches for procedural terrain generation-a brief review of current techniques. In: IEEE. *2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. [S.l.], 2016. p. 1–2. Citado 3 vezes nas páginas 21, 23 e 27.
- RUBIO, V. E. F.; RÍO, G. Guzmán del; ARRIBAS, C. L. Generación procedural de contenido basado en aprendizaje automático. 2020. Citado na página 11.

- SANTAMARÍA-IBIRIKA, A. et al. Volumetric virtual worlds with layered terrain generation. In: IEEE. *2013 International Conference on Cyberworlds*. [S.l.], 2013. p. 20–27. Citado na página [36](#).
- SANTOS, G. G.; GRIEBELER, N. P.; OLIVEIRA, L. F. de. Chuvas intensas relacionadas à erosão hídrica. *Revista Brasileira de Engenharia Agrícola e Ambiental*, SciELO Brasil, v. 14, p. 115–123, 2010. Citado na página [26](#).
- SCHAAL, J. Procedural terrain generation. a case study from the game industry. *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, Springer, p. 133–150, 2017. Citado na página [23](#).
- SHAKER, N.; TOGELIUS, J.; NELSON, M. J. Procedural content generation in games. Springer, 2016. Citado na página [11](#).
- SMELIK, R. M. et al. A survey of procedural methods for terrain modelling. In: SN. *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*. [S.l.], 2009. v. 2009, p. 25–34. Citado na página [16](#).
- ŠT'AVA, O. et al. Interactive terrain modeling using hydraulic erosion. In: *Proceedings of the 2008 acm siggraph/eurographics symposium on computer animation*. [S.l.: s.n.], 2008. p. 201–210. Citado 2 vezes nas páginas [24](#) e [83](#).
- VALDIVIA, D. B.; GARCÍA, J. R.; LASTRA, G. S. Generación de terrenos fractales para escenas 3d, en opengl y vulkan. 2020. Citado na página [21](#).
- VALENCIA-ROSADO, L. O.; STAROSTENKO, O. Methods for procedural terrain generation: a review. In: SPRINGER. *Pattern Recognition: 11th Mexican Conference, MCPR 2019, Querétaro, Mexico, June 26–29, 2019, Proceedings 11*. [S.l.], 2019. p. 58–67. Citado 2 vezes nas páginas [12](#) e [16](#).
- WANG, J.; LIU, Y.; JUN, Y. Introducing 3d model search into case study library management for engineering education. In: ATLANTIS PRESS. *2015 3rd International Conference on Education, Management, Arts, Economics and Social Science*. [S.l.], 2015. p. 600–605. Citado na página [17](#).
- WARSZAWSKI, K. K.; NIKIEL, S. S. A proposition of erosion algorithm for terrain models with hardness layer. *J Theor Appl Comput Sci*, v. 8, n. 1, p. 76–84, 2014. Citado na página [24](#).
- WILLEMSE, J.; HAWICK, K. Generation and rendering of fractal terrains on approximated spherical surfaces. In: *Proc. 17th Int. Conf. Comput. Graph. Virtual Reality*. [S.l.: s.n.], 2013. Citado na página [18](#).