Simulação de erosão aplicada a geração procedural de terrenos

André Luiz Vasconcelos Ferreira

JUIZ DE FORA

FEVEREIRO, 2022

Simulação de erosão aplicada a geração procedural de terrenos

André Luiz Vasconcelos Ferreira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Marcelo Caniato Renhe

JUIZ DE FORA

FEVEREIRO, 2022

Simulação de erosão aplicada a geração procedural de terrenos

André Luiz Vasconcelos Ferreira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Caniato Renhe
Doutor em Engenharia de Sistemas e Computação

Igor de Oliveira Knop
Doutor em Modelagem Computacional

Luiz Maurilio da Silva Maciel
Doutor em Engenharia de Sistemas e Computação

JUIZ DE FORA
18 DE FEVEREIRO, 2022

Aos meus pais e amigos, pelo apoio e carinho.

Resumo

Métodos capazes de gerar terrenos virtuais que possuam um aspecto natural são de grande interesse para diversas aplicações. É possível gerar terrenos realistas de forma procedural sem se preocupar em simular os fenômenos físicos que dão origem a paisagens naturais. Porém, alguns detalhes são perdidos nesse tipo de abordagem. Nesse sentido, o uso de algoritmos que simulam processos erosivos se torna interessante, uma vez que isso permite gerar cenários que preservam esses detalhes. Assim, este trabalho propõe um estudo da aplicabilidade de simuladores de erosão hidráulica e térmica no contexto de geração procedural, bem como o seu uso como forma de aprimorar os terrenos obtidos por métodos procedurais não-físicos. São apresentados alguns resultados de execução dos algoritmos estudados, analisando de forma qualitativa as características dos terrenos gerados.

Palavras-chave: simulação de erosão; erosão hidráulica; erosão térmica; terrenos; geração procedural.

Abstract

Methods for generating virtual terrains with natural aspect are of great interest for many applications. It is possible to procedurally generate realistic terrains without relying on the physical phenomena associated with the formation of natural landscapes. However, some detail can be lost with this approach. In this context, the use of algorithms that simulate erosion events becomes interesting since this allows the generation of scenes that preserve such details. Therefore, this work proposes a study of the applicability of hydraulic and thermal erosion simulators in the context of procedural generation, as well as the use of these simulators to enhance terrains produced by non-physical methods. Some results obtained with the studied algorithms are presented, alongside a qualitative analysis of the characteristics of the generated terrains.

Keywords: erosion simulation; hydraulic erosion; thermal erosion; terrains; procedural generation.

Agradecimentos

A todos os meus parentes e amigos, pelo encorajamento e apoio. Ao professor Marcelo Caniato pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria. Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Água mole em pedra dura tanto bate até que fura".

Ditado popular

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
GPU	*Graphics Processing Unit*
SPH	*Smoothed Particle Hydrodynamics*

1 Introdução

Dada a grande quantidade de trabalhos que exploram a área de geração de terrenos em computação gráfica, esse campo de estudo se mostra amplo e interessante. Entre as aplicações, pode-se citar os jogos digitais, nos quais a exploração do cenário é um dos maiores motivadores para manter os jogadores retidos nessa experiência de lazer, ou os filmes, que muitas vezes possuem cenas em ambientes fictícios ou inacessíveis no mundo real (Galin et al, 2019). Com técnicas de geração procedural de terrenos é possível produzir estes cenários rapidamente.

Galin et al (2019) dividem os métodos de geração de terrenos em três categorias: métodos procedurais, que consistem em criar terrenos a partir de algoritmos sem necessariamente simular o processo físico de geração (Musgrave et al, 1989; Kamal et al, 2007); métodos baseados em exemplos, que geram terrenos a partir de dados do mundo real (Gain et al, 2015); e simulações, que consistem na reprodução de processos geomorfológicos a fim de criar uma paisagem convincente (Cordonnier et al, 2016, 2017). Neste trabalho, será adotada a mesma terminologia de Galin et al (2019). Embora os métodos de simulação também sejam uma forma procedural de gerar terrenos, apenas métodos não-físicos serão enquadrados na primeira categoria.

Os métodos de geração procedural buscam reproduzir algoritmicamente propriedades dos terrenos de diversas maneiras. Alguns exemplos de características reproduzidas por esses métodos são a aleatoriedade no formato dessas estruturas e a propriedade de autossimilaridade, que trata-se da recorrência de padrões em diferentes escalas em um mesmo objeto (Shaker et al, 2016). Entre os algoritmos de geração procedural de terrenos que possuem destaque, é possível citar o método de geração por falhas (DeLoura et al, 2000), o método do diamante-quadrado (Fournier et al, 1982), o ruído de Perlin (Perlin et al, 1985) e o método de geração de colinas (Nystrom et al, 2002).

Com relação aos algoritmos de simulação, pode-se citar como exemplos os que reproduzem processos erosivos, como a erosão hidráulica (Neidhold et al, 2005; Mei et al, 2007; St'ava et al, 2008; Jákó et al, 2011), a erosão eólica (Benes et al, 2004; Raris et al,

2019) e a erosão térmica (Benes et al, 2001; Jákó et al, 2011). Esses métodos normalmente são utilizados com o intuito de compor cenários com detalhes do mundo real uma vez que métodos procedurais não tratam desse tipo de questão (Peytavie et al, 2009; Gonçalves et al, 2020). Um exemplo disso é a comparação de áreas mais elevadas com as menos elevadas de um cenário. Em um terreno real, as bases de estruturas geológicas têm resíduos que são transportados das partes mais elevadas ao longo do tempo, o que não é observado nos terrenos gerados por algoritmos procedurais (Musgrave et al, 1989; Gonçalves et al, 2020).

1.1 Problema

Dada a diversidade de algoritmos de geração de terrenos encontrados na literatura, tanto de abordagens procedurais quanto daquelas baseadas em física, este trabalho visa fazer um estudo do uso de alguns desses métodos para a geração e modelagem de cenários. Esta pesquisa, portanto, visa compreender como os métodos de simulação de erosão podem contribuir para a geração de terrenos realistas, além de entender em quais contextos cada abordagem se torna mais interessante.

1.2 Objetivos

O objetivo geral deste trabalho é estudar o uso de algoritmos de simulação de erosão hidráulica e térmica na geração de terrenos e no aprimoramento de resultados obtidos com métodos de geração não-físicos. O trabalho tem como objetivos específicos estudar técnicas de geração e modelagem de terrenos, compreender e implementar métodos de simulação de erosão hidráulica e térmica, e analisar a aplicabilidade do uso de simuladores de erosão na construção de cenários e na melhoria de terrenos previamente gerados por métodos procedurais.

1.3 Organização do trabalho

Esta monografia está organizada da seguinte maneira: no Capítulo 2, é feita uma revisão de diversos trabalhos associados ao tema, além de introduzir conceitos fundamentais sobre terrenos e geração procedural de terrenos; o Capítulo 3 discute os detalhes de implementação dos métodos de simulação de erosão estudados; no Capítulo 4, são apresentados e analisados os resultados dos experimentos realizados com os métodos desenvolvidos; por fim, o Capítulo 5 apresenta as conclusões e perspectivas para o futuro do trabalho.

2 Revisão Bibliográfica e Fundamentação

Neste capítulo são discutidos trabalhos relevantes da literatura nos contextos de geração procedural de terrenos e simulação de erosão. Esta discussão está presente na Seção 2.1. Além disso, nas seções subsequentes, são introduzidos conceitos essenciais relativos à representação de terrenos e alguns métodos selecionados de geração procedural e simulação de erosão hidráulica e térmica.

2.1 Revisão Bibliográfica

Um dos trabalhos que fundamentam a simulação de erosão hidráulica, considerando os que adotam uma abordagem baseada em grade, é o apresentado em Neidhold et al (2005). Os trabalhos anteriores a esse simulavam o processo erosivo com fractais (Kelley et al, 1988; Prusinkiewicz et al, 1993). O diferencial de Neidhold et al (2005) está no fato de ele conseguir simular erosão hidráulica em níveis interativos para cenários de pequena escala, preservando o rigor físico do processo. Este rigor físico se dá graças à simulação de fluidos utilizada no trabalho, que foi baseada naquela apresentada por Stam et al (1999, 2003). O algoritmo de erosão em si é parecido com o que foi desenvolvido por seus antecessores, Musgrave et al (1989) e Chiba et al (1998), e segue a ideia de determinar um limite de sedimentos dissolvidos em um certo volume de água. Quando esse limite é ultrapassado, o fluido deposita material no solo. Caso contrário, o material é dissolvido no fluido. Este método de simulação de erosão hidraúlica tem seu padrão reproduzido e desenvolvido por trabalhos posteriores ao de Neidhold et al (2005), mas não é o único.

Pouco tempo depois, Benes et al (2006) publicou um trabalho em que é simulado um processo de erosão hidráulica totalmente tridimensional, com um simulador de fluidos baseado nas equações de Navier-Stokes, e o terreno erodido sendo representado em uma grade tridimensional. Apesar de um maior rigor físico nos resultados obtidos, o método é limitado a terrenos de pequena escala, devido a questões de desempenho. Enquanto o trabalho de Neidhold et al (2005) utiliza um mapa de alturas 2D para a representação do

terreno, o de Benes et al (2006) representa tanto o fluido quanto o terreno na grade 3D.

Em Benes et al (2007), os autores descrevem um outro método de simulação de erosão hidráulica. O método em questão vem da ideia de que à medida que a água penetra na superfície do terreno, é gerada uma camada solta de materiais que cobrem o solo, conhecida como regolito, que é transportada pelo fluido até que este evapore e, enfim, deposite os sedimentos em um novo local. Além de uma abordagem diferente para a simulação de erosão hidráulica, este trabalho propõe o uso de *shallow water* no algoritmo a fim de melhorar seu desempenho. *Shallow water* é um método de simulação de fluidos simplificado que possui menor custo computacional quando comparado a outros simuladores. O trabalho proposto em Mei et al (2007), publicado em paralelo a Benes et al (2007), também faz uso de *shallow water* como forma de melhorar o desempenho do simulador de erosão, com a diferença de que ele utiliza um algoritmo de simulação de erosão hidráulica parecido com o proposto em Neidhold et al (2005).

Posteriormente, diversos trabalhos sobre simulação de erosão hidráulica focaram em melhorar o desempenho dos simuladores. Alguns desses trabalhos buscam atingir esse objetivo utilizando métodos de simulação de fluidos com menor custo computacional, como os que usam as equações de *shallow water*, vistos em Benes et al (2007); Mei et al (2007). Outros propõem implementações que tiram proveito do paralelismo da GPU, como também pode ser visto em Mei et al (2007) e Jákó et al (2011). Além de visar um melhor desempenho, Jákó et al (2011) ainda apresenta novas estratégias para a geração de terrenos realistas com a simulação de erosão hidráulica, como, por exemplo, considerar a profundidade da água em pontos específicos do terreno ao realizar o cálculo do transporte de sedimento da erosão.

Outro trabalho que se destaca é o de St'ava et al (2008). Esse trabalho também visa um melhor desempenho para a simulação de erosão hidráulica, utilizando as técnicas de melhoria já citadas. Os resultados obtidos são interativos e podem ser usados em terrenos de larga escala. Além disso, quando comparado com os trabalhos citados anteriormente, o método desenvolvido em St'ava et al (2008) permite gerar uma maior variedade de terrenos, já que há a presença de três algoritmos distintos de erosão e de uma estrutura de dados que permite o uso de diferentes materiais para o solo. A estrutura de

dados em questão é apresentada em Benes et al (2001). O primeiro algoritmo de erosão, o *Force-based algorithm*, é baseado naquele que foi desenvolvido em Neidhold et al (2005) e melhorado em Mei et al (2007), e é utilizado para simular o efeito da dinâmica da água em altas velocidades sobre o solo. O segundo algoritmo, o *Dissolution-based algorithm*, é baseado naquele desenvolvido em Benes et al (2007), e é utilizado para simular o efeito da dinâmica da água em baixas velocidades. O terceiro algoritmo, o *Material Slippage*, não se trata de um processo de erosão hidráulica, mas de um processo erosivo gerado pela ação da gravidade sobre materiais do solo que resistem à ação da água mas ficam soltos no chão, sendo transportados de um ponto a outro devido à inclinação da superfície em que se encontram. Apesar de o *Material Slippage* não ser referenciado como "erosão térmica" em St'ava et al (2008), ele é considerado como tal neste trabalho de conclusão de curso por ter todas as características de um simulador desse tipo.

Os trabalhos mais recentes ligados ao problema de simulação de erosão hidráulica possuem tratamentos para diversos aspectos desta questão. Há ainda aqueles que visam melhorar o desempenho do processo. Um exemplo de trabalho que faz isso é o de Müller et al (2016), que consiste em acelerar o algoritmo de erosão hidráulica adiantando os cálculos que não dependem da simulação de física através do paralelismo da GPU. Outras aplicações são aquelas que utilizam a simulação de erosão hidráulica, em conjunto com outros métodos de modelagem de terrenos, como ferramenta auxiliadora para gerar terrenos realistas, como em Cordonnier et al (2017), Cristea et al (2015) e Gonçalves et al (2020). Também existem pesquisas que visam acrescentar mais detalhes à simulação de erosão, como em Hawkins et al (2021), que considera a influência da vegetação do cenário na intensidade do processo erosivo.

Diferentemente dos modelos de erosão hidráulica que utilizam *shallow water*, existem aqueles que exploram o uso de métodos de simulação de fluidos mais elaborados, como o *Smoothed Particle Hydrodynamics* (SPH) (Monaghan et al, 1994). Esse é o caso dos métodos propostos em Krištof et al (2009), Krs et al (2020) e Gu-ping et al (2020). Como estas abordagens fogem ao escopo deste trabalho, esses métodos não serão discutidos aqui.

A erosão térmica foi introduzida em Musgrave et al (1989) junto de outras técnicas de simulação de erosão em terrenos fractais. Posteriormente, Roudier et al (1993) mostrou

a noção de resistência do material para o algoritmo de erosão térmica. O trabalho também apresenta esse conceito para a erosão hidráulica. A resistência do material é incluída no cálculo da quantidade de sedimento transportado no processo por meio de um valor que é multiplicado por essa quantidade. Esse valor então diminui o volume de sedimento transportado. Essas noções podem ser encontradas em trabalhos posteriores, como Jákó et al (2011).

Entre as aplicações do algoritmo ao longo do tempo, pode-se citar o uso dele na composição de terrenos com formas específicas. Em Benes et al (2005), por exemplo, a erosão térmica é utilizada em conjunto com outros processos para gerar mesas, que são estruturas geológicas elevadas cujo topo é plano. A função da erosão térmica nesse contexto é transformar partes rígidas e maciças das mesas em sedimentos granulares, que por sua vez sofrem com a atuação de outras intempéries.

Outra aplicação para a erosão térmica é o seu uso em terrenos com estruturas côncavas. Tal ideia foi desenvolvida por Peytavie et al (2009), que simula os efeitos da erosão térmica em uma estrutura de dados 3D. Esse efeito é obtido a partir do processamento de materiais que se desprendem de partes verticais do cenário, o que gera sedimentos que se estabilizam em amontoados no solo. O espaço entre esses dois elementos do cenário é vazio, o que gera a aparência côncava. O problema desse trabalho é o fato de seu sistema ter as dimensões limitadas devido ao seu custo de memória. Outro trabalho com o mesmo objetivo e que também simula o terreno com uma estrutura de dados 3D é o de Jones et al (2010).

A erosão térmica também já foi usada para objetivos diferentes da modelagem de terrenos. Sumner et al (1999) utilizou as técnicas de simulação de erosão térmica para gerar a aparência de pegadas ou marcas de bicicletas no solo. Isso foi feito aplicando o algoritmo apenas em áreas ocupadas por objetos que pudessem deixar essas marcas.

Muitas pesquisas utilizam a abordagem de erosão térmica junto com a erosão hidráulica, como em Musgrave et al (1989), St'ava et al (2008), Jákó et al (2011) e Gonçalves et al (2020). Um trabalho recente que une a erosão hidráulica e térmica e se destaca é o de Skorkovská et al (2019). Partindo do fato de que os sistemas que simulam erosão somente geram terrenos complexos a um custo de memória elevado, por meio do

uso de estruturas de dados tridimensionais, o trabalho em questão propõe um modelo unificado de erosão hidráulica e térmica que atua sobre uma malha de triângulos para simplificar o custo da erosão em paisagens extensas.

2.2 Representação de um Terreno

Uma forma comum de representação de terrenos em computação gráfica são os modelos de elevação. Estes consistem em funções $h : \mathbb{R}^2 \to \mathbb{R}$ que mapeiam alturas em um espaço bidimensional (Galin et al, 2019). Entre as estruturas de dados que se enquadram nesse conjunto de representações, pode-se citar os mapas de alturas e os modelos divididos em camadas (Galin et al, 2019). O primeiro trata-se de uma grade quadrada ou retangular que carrega em cada célula uma altura, ao passo que o segundo é uma coleção de camadas que possuem propriedades distintas, como resistência, densidade ou a presença de materiais específicos, sobrepostos uns aos outros. Como dito em Galin et al (2019), os modelos divididos em camadas, em termos de implementação, são uma pilha de mapas de alturas. Alguns exemplos de trabalhos que usam essas estruturas são comentados a seguir.

Em Neidhold et al (2005), Mei et al (2007) e Jákó et al (2011), o terreno é representado por um mapa de alturas de duas dimensões. Esses mapas de alturas são acompanhados de grades que carregam informações sobre a simulação, como volume de sedimentos dissolvidos na água, campo de velocidades da água e a altura do terreno nos pontos do cenário. Já em Benes et al (2001) e em St'ava et al (2008), o terreno é representado seguindo um modelo baseado em camadas, sendo cada uma delas uma representação de um material diferente do solo.

O uso de modelos de elevação possui desvantagens. Um mapa de alturas, por exemplo, não é capaz, sozinho, de representar cavernas ou buracos (Galin et al, 2019). Os modelos baseados em camadas conseguem contornar este detalhe, mas o custo de armazenamento deles é maior. Comparando um mapa de alturas e um modelo baseado em camadas, ambos com n linhas e n colunas, o primeiro tem um custo de armazenamento de memória na ordem de $O(n^2)$, ao passo que o segundo possui custo de armazenamento na ordem de $O(kn^2)$, sendo k o número de camadas da representação do terreno (Benes et al, 2001).

Uma outra forma comum de representação de terrenos é a volumétrica, que se baseia em funções $h : \mathbb{R}^3 \rightarrow M$, em que M é um índice que representa algum material (Galin et al, 2019). Uma forma de implementar esse tipo de estrutura de dados é por meio do uso de grades tridimensionais (Galin et al, 2019). Um trabalho que faz uso dessa técnica é o de Benes et al (2006). Com essa estrutura é possível a representação de cavernas e buracos, mas seu custo de armazenamento é na ordem de $O(n^3)$, sendo n a altura, largura e comprimento da grade que representa o terreno (Galin et al, 2019).

Neste trabalho, será utilizada a representação por mapa de alturas. Assim, todos os métodos e resultados que serão apresentados adiante assumem o uso de terrenos representados dessa forma.

2.3 Geração Procedural de Terrenos

Os algoritmos de geração procedural produzem resultados satisfatórios e rápidos por meio de técnicas matemáticas ou algorítmicas, mas não consideram processos físicos em sua execução, o que pode gerar terrenos menos convincentes. Para este trabalho, foram selecionados alguns dos algoritmos procedurais mais relevantes, como o método de geração de colinas, o método de geração por falhas e o método do diamante-quadrado, explicados nas Seções 2.3.1, 2.3.2 e 2.3.3, respectivamente.

2.3.1 Método de geração de colinas

O método de geração de colinas (Nystrom et al, 2002) consiste em gerar elevações parabólicas em pontos específicos do mapa de alturas a partir da Equação (2.1):

$$h(p) = r^2 - ((x_2 - x_1)^2 + (y_2 - y_1)^2), \tag{2.1}$$

em que $h(p)$ é o valor do mapa de alturas em um ponto arbitrário p, r é a distância entre o ponto central da elevação e de sua base, x_1 e y_1 são as coordenadas (x, y) do ponto p e x_2 e y_2 são as coordenadas do mapa de alturas intermediárias à base e ao centro p da montanha.

O algoritmo recebe como entrada uma célula p do mapa de alturas, cujas co-

ordenadas são (x_1, y_1), e um raio r indicando a distância entre o centro da montanha e sua base. Em seguida, as células cujos pontos (x_2, y_2) têm suas distâncias ao ponto p menores ou iguais a r recebem o resultado da Equação (2.1). Por fim, o algoritmo gerador de montanhas é aplicado em n pontos aleatórios do mapa de alturas, com n sendo uma entrada do algoritmo.

2.3.2 Método de geração por falhas

O método de geração por falhas (DeLoura et al, 2000) consiste em traçar linhas retas que atravessem o cenário. Todos os valores da estrutura de dados que se encontram em um lado da reta são somados por um fator k, ao passo que os valores do outro lado são subtraídos pelo mesmo fator.

Como mostrado em Fernandes (2006), existem diversas abordagens para traçar a reta. Ela pode ser criada por meio da forma normal de Hesse (Bôcher et al, 1915), por uma reta gerada a partir de dois pontos aleatórios do cenário, ou por um vetor também obtido a partir de dois pontos aleatórios. Cada abordagem determina quais partes do mapa terão suas altitudes somadas ou subtraídas pelo fator k_d. Neste trabalho foi adotado o método baseado na forma normal de Hesse, que é explicado em mais detalhes a seguir.

Dados um segmento de reta AB representado no plano cartesiano, um segmento de reta OP de comprimento p passando pela origem e perpendicular a AB, e um ângulo α entre o segmento OP e o eixo X, a reta AB pode ser representada pela Equação (2.2) (Bôcher et al, 1915):

$$x\cos(\alpha) + y\,\mathrm{sen}(\alpha) = p. \tag{2.2}$$

Se deseja-se descobrir a distância de um ponto aleatório p_r à reta AB, basta imaginar que suas coordenadas (x_r, y_r) são o ponto de origem do plano cartesiano. Nesse caso, a distância c entre p_r e AB seria dada por:

$$\begin{aligned} (x - x_r)\cos(\alpha) + (y - y_r)\,\mathrm{sen}(\alpha) &= c \\ x\cos(\alpha) - x_r\cos(\alpha) + y\,\mathrm{sen}(\alpha) - y_r\,\mathrm{sen}(\alpha) &= c \\ p - x_r\cos(\alpha) - y_r\,\mathrm{sen}(\alpha) &= c. \end{aligned} \tag{2.3}$$

Com isso, é possível calcular se um ponto (x, y) qualquer do mapa de alturas terá seu valor somado ou subtraído. Basta verificar se o valor c possui sinal negativo ou não.

Assim, para gerar uma reta aleatória no terreno, é sorteado um valor para α e um valor para p. O valor de p deve estar dentro dos limites do cenário, portanto no intervalo $[-\frac{d}{2}, \frac{d}{2}]$, sendo $d = \sqrt{l^2 + w^2}$, l a largura e w o comprimento do cenário. Com esses valores sorteados já se possui todas as informações da reta aleatoriamente traçada sobre o plano.

Para suavizar os resultados, ao invés de apenas somar ou subtrair as alturas nos pontos (x, y) dependendo do sinal do valor c obtido na Equação (2.2), é interessante aplicar uma função cosseno ou seno perto da linha gerada. Dessa forma, se o valor absoluto da distância de um ponto (x, y) à linha imaginária for menor do que um valor como 2π ou $\frac{\pi}{2}$, aplica-se uma das funções trigonométricas sugeridas. Caso contrário, o valor da altura se mantém o mesmo ou sofre alguma modificação parecida com a feita na abordagem padrão do método de geração por falhas.

Nas Equações (2.4), (2.5) e (2.6) estão descritas as funções $h(p)^t$ possíveis para a altitude do terreno após a linha imaginária ser traçada. A primeira é a função padrão de tratamento das alturas, a segunda é a função seno e a terceira é a função cosseno. O t é o instante de tempo em que a função é aplicada:

$$h(p)^t = \begin{cases} h(p)^{t-1} + k_d & , c \geq 0 \\ h(p)^{t-1} - k_d & , c < 0 \end{cases}, \tag{2.4}$$

$$h(p)^t = \begin{cases} h(p)^{t-1} + k_d & , c > 2\pi \\ h(p)^{t-1} - k_d & , c < -2\pi \\ h(p)^{t-1} + \operatorname{sen}(c) & , c \in [-2\pi, 2\pi] \end{cases}, \tag{2.5}$$

$$h(p)^t = \begin{cases} h(p)^{t-1} + 0 & , c \notin [-\pi, \pi] \\ h(p)^{t-1} + \cos(c) & , c \in [-\pi, \pi] \end{cases}. \tag{2.6}$$

2.3.3 Método do diamante-quadrado

O algoritmo do diamante-quadrado (Fournier et al, 1982) é um método recursivo de geração procedural de terrenos. Seu funcionamento ocorre em cenários de base quadrada

e é dividido em duas etapas: passo do quadrado e passo do diamante. A cada iteração o algoritmo executa as duas operações em subdivisões quadradas do cenário cada vez menores. A recursão é interrompida no momento em que não há mais subdivisões possíveis de serem encontradas, ou seja, quando não é mais possível encontrar um ponto central na subdivisão quadrada (Olsen et al, 2004; Gonçalves et al, 2020).

O processo se inicia com a determinação de valores aleatórios para os vértices do quadrado inicial (Gonçalves et al, 2020). Depois disso começam as iterações recursivas do algoritmo.

A primeira etapa da iteração, o passo do quadrado, consiste em calcular a altura central do subquadrado a partir das alturas de seus vértices. Primeiro é efetuada uma média aritmética destes e em seguida o valor obtido é somado a um número aleatório no intervalo $\left[-\frac{d}{2}, \frac{d}{2}\right]$, sendo d um valor arbitrário (DeLoura et al, 2000; Gonçalves et al, 2020). O resultado dessas operações é atribuído à altura do ponto central do subquadrado. DeLoura et al (2000) sugere que um bom valor candidato inicial de d seja o comprimento do primeiro quadrado tratado pelo método.

A segunda etapa da iteração, o passo do diamante, consiste em calcular a altura dos pontos médios das arestas dos subquadrados. Esse cálculo é feito a partir da média aritmética entre as alturas dos vértices que compõem o segmento de reta no qual o ponto médio está e a altura do ponto central do subquadrado (DeLoura et al, 2000; Gonçalves et al, 2020). Novamente, o resultado obtido da média aritmética é somado a um valor aleatório no intervalo $\left[-\frac{d}{2}, \frac{d}{2}\right]$, com d sendo o mesmo número arbitrário utilizado no passo do quadrado.

Com a execução das duas etapas, o valor d é dividido pela metade e os passos descritos são executados recursivamente para cada um dos subquadrados gerados. A Figura 2.1 ilustra a execução do algoritmo em duas iterações para uma grade com 5 colunas e linhas.

2.4 Simulação de Erosão

Como é explicado por Galin et al (2019), a erosão pode ser vista como um processo de retirada, movimentação e depósito de materiais em um cenário. Esse processo ocorre com

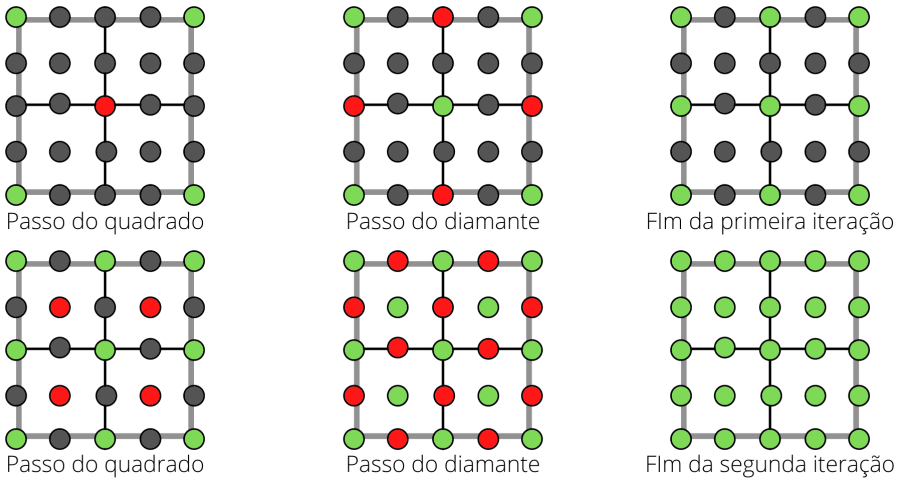


Figura 2.1: Ilustração de duas iterações do algoritmo diamante-quadrado. Em verde, as células da grade com as alturas já tratadas; em vermelho, as células da grade com as alturas em tratamento; em cinza, as células da grade ainda não tratadas.

a ação de um agente, seja ele o vento, a gravidade, a água, ou outros elementos presentes na natureza. O que um simulador de erosão faz é calcular os efeitos das interações do solo e do agente erosivo ao longo do tempo. Nesta seção, serão apresentados dois modelos de erosão: a hidráulica e a térmica.

2.4.1 Erosão Hidráulica

No caso da erosão hidráulica, o *force-based algorithm* apresentado em St'ava et al (2008) se assemelha em muitos aspectos com os algoritmos de erosão apresentados em outros trabalhos, como Neidhold et al (2005), Mei et al (2007), Anh et al (2007), Cristea et al (2015) e Müller et al (2016), e até em trabalhos mais antigos, como Musgrave et al (1989) e Chiba et al (1998). O fator comum entre essas implementações de simuladores de erosão hidráulica está na determinação de uma capacidade máxima de sedimento dissolvido na água responsável pela erosão. Quando a capacidade máxima é superada, ocorre depósito de material no solo.

Sendo assim, pode-se associar ao fluido responsável pela erosão hidráulica uma constante c, que será chamada de constante de capacidade de sedimento, que determina a quantidade máxima de material que pode estar presente na água. Concomitantemente, o fluido ainda possui uma constante K_D de deposição e uma constante K_S de dissolução.

Caso a quantidade ΔS de material dissolvido no fluido ultrapasse c, o solo no qual este fluido está atuando terá uma nova altura H determinada pela seguinte equação:

$$H = H_0 + \frac{K_D}{\Delta t} \cdot (\Delta S - c), \tag{2.7}$$

enquanto a quantidade de material dissolvido na água, S, passará a ter o seguinte valor:

$$S = S_0 + \Delta S - \frac{K_D}{\Delta t} \cdot (\Delta S - c). \tag{2.8}$$

Caso a quantidade ΔS de material dissolvido no fluido seja menor que c, a altura H do solo e a quantidade S de material dissolvido na água serão dadas por:

$$H = H_0 - \frac{K_S}{\Delta t} \cdot (c - \Delta S), \tag{2.9}$$

$$S = S_0 + \Delta S + \frac{K_S}{\Delta t} \cdot (c - \Delta S). \tag{2.10}$$

, em que H_0 e S_0 são, respectivamente, a altura e a quantidade de sedimento antes do processo de deposição ou dissolução, ΔS é a quantidade de sedimento transportada para o ponto em que ocorre a deposição e dissolução de sedimento na iteração corrente, e Δt é o tempo, determinado na implementação do simulador, entre o momento anterior e posterior do processo erosivo.

A ideia geral do algoritmo de erosão mostrado é muito parecida, mesmo que com algumas modificações, com o que é descrito em Neidhold et al (2005). Da mesma forma, muitos algoritmos de simulação de erosão hidráulica presentes em outros trabalhos seguem uma lógica similar a esta (Mei et al, 2007; Anh et al, 2007; St'ava et al, 2008; Jákó et al, 2011; Cristea et al, 2015; Müller et al, 2016; Gonçalves et al, 2020). Assim, o sistema de erosão desenvolvido neste trabalho de conclusão de curso também se baseia nessa abordagem.

Shallow water

Como é explicado por Benes et al (2007), muitos trabalhos que visam simular fluidos se baseiam nas equações de *Navier-Stokes*, como é o caso de Stam et al (1999) e outros que

o seguiram. Entretanto, a complexidade de tempo desse tipo de simulação é da ordem de $O(n^3)$, em que n é o número de células da grade que representa o fluido em um ambiente 3D. Para muitas aplicações esse custo computacional é elevado, o que motiva o uso da *shallow water* (Kass et al, 1990), uma solução simplificada das equações de *Navier-Stokes*. A implementação dessa abordagem consiste em utilizar um mapa de alturas para representar o fluido, tendo como desvantagem a não reprodução de efeitos que um simulador de fluidos completo teria, como a quebra de ondas, mas compensando isso com a possibilidade de se obter resultados interativos.

Entre as implementações possíveis da *shallow water*, a utilizada em St'ava et al (2008), Jákó et al (2011) e Mei et al (2007) é conhecida como *pipe-model* e consiste em gerar fluxo de água a partir da diferença de altitude entre as células do mapa de alturas que representa a *shallow water*.

Cada célula da *shallow water* possui um valor $w(p)$ que representa a altura de água em um ponto p do mapa. Quanto maior a altura da água em um dado ponto, maior o volume nele.

Sendo assim, a altura $A(p)$ da água em um ponto p do mapa é dada por:

$$A(p) = w(p) + h(p), \tag{2.11}$$

em que $h(p)$ é o valor do mapa de alturas do terreno no ponto p.

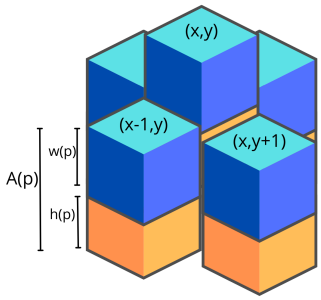


Figura 2.2: Representação gráfica da estrutura de dados da *shallow water*.

Comparando duas alturas adjacentes, $A(x, y)$ e $A(x+i, y+j)$, com $i, j \in [-1, 1]$, $i, j \in \mathbb{Z}$ e $|i| \neq |j|$, nota-se uma diferença ΔA entre elas. Essa diferença gera uma pressão $P(p)$ entre os dois pontos que visa estabilizar suas altitudes. A equação da pressão é dada por:

$$P(p) = \rho g \Delta A, \tag{2.12}$$

em que ρ é a densidade do fluido, g é a aceleração da gravidade e ΔA é a diferença de alturas entre os pontos nos quais a pressão está atuando.

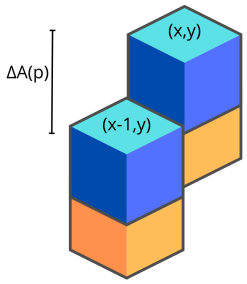


Figura 2.3: Representação gráfica da diferença de alturas entre células vizinhas da *shallow water*.

Por consequência, essa pressão gera um fluxo entre os pontos, o que resulta no movimento da água. A aceleração da água saindo do ponto $p = (x, y)$ e indo ao ponto adjacente $p = (x + i, y + j)$ é descrita pela equação a seguir:

$$a(p) = \frac{P(p)}{l}, \tag{2.13}$$

em que l é o comprimento de um cano imaginário existente entre as duas células da grade que representa a *shallow water*. Dessa forma, o fluxo de água que sai do ponto $p = (x, y)$ e vai para o ponto $p = (x + i, y + j)$, no tempo $t + \Delta t$, é dado por:

$$f_{t+\Delta t}(p) = f_t(p) + \Delta t C a(p), \tag{2.14}$$

sendo C a área do cano imaginário existente entre as duas células e $f_t(p)$ o fluxo no tempo t.

Com isso, uma célula (i, j) do sistema possui até quatro fluxos de saída: um direcionado para a célula $(i - 1, j)$, outro para a célula $(i + 1, j)$, outro para a célula $(i, j - 1)$ e outro para a célula $(i, j + 1)$, chamados, respectivamente, de f_l,f_r,f_b,f_t. Se uma dessas células possui a altura A maior do que o da célula (i, j), o fluxo direcionado para ela é igual a zero.

Como a soma dos fluxos pode resultar em um volume de água maior do que o presente na célula (i, j), ainda é feito um tratamento a respeito desse valor por meio de um fator K dado por:

$$K = \min\left(1.0, \frac{w(i,j) * I_x * I_y}{f_r + f_l + f_t + f_b}\right), \tag{2.15}$$

em que I_x e I_y são as dimensões da célula. Este fator representa a porcentagem do total de água presente na célula (i, j) em relação ao total de água retirada com as somas dos fluxos de saída. Se a porcentagem for maior que 100%, o fator K é igual a 1.0, caso contrário seu valor é igual à porcentagem obtida com o cálculo. O fator K então é multiplicado a cada fluxo de saída, impedindo assim que o total retirado da célula seja maior que o volume de água presente nela.

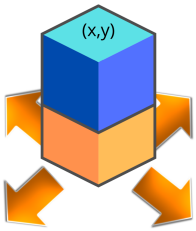


Figura 2.4: Representação gráfica dos fluxos de saída da *shallow water*.

Após calcular os fluxos de saída de cada célula do sistema, as novas alturas são calculadas. Isso é feito a partir da diferença entre a soma dos fluxos de saída e entrada da célula. Os fluxos de entrada de uma célula (i, j) são os fluxos f_r de $(i-1, j)$, f_l de $(i+1, j)$, f_t de $(i, j-1)$ e f_b de $(i, j+1)$.

O tratamento de bordas de um sistema baseado em *shallow water* é feito utilizando células-fantasma, que são células adicionais incluídas ao redor de toda a grade, de forma a facilitar os cálculos. Para um sistema de N linhas e N colunas, essas células, que podem ser identificadas com os pontos $(-1, j)$, $(i, -1)$, $(N+1, j)$, $(i, N+1)$, $i, j \in [-1, N+1]$, têm valores de altura da água iguais aos das células $(0, j)$, $(i, 0)$, (N, j), (i, N), respectivamente. Dessa forma não há diferença de altura entre essas células e, portanto, não há geração nem de fluxos de saída nem de entrada entre elas.

2.4.2 Erosão Térmica

Na natureza, a erosão térmica ocorre pela dilatação de estruturas geológicas, ocasionada por mudanças na temperatura. Essa dilatação provoca rachaduras que deixam sobre o local os chamados regolitos, que são camadas de sedimento suspenso em superfícies rochosas. Por conta da posição e da ação da gravidade, esses regolitos são transportados de partes mais elevadas para partes menos elevadas (Cristea et al, 2015; Galin et al, 2019). O que determina essa movimentação é a inclinação da superfície em que se inicia o transporte de sedimentos. Quando esta ultrapassa um limite imposto pelo ângulo de talude, medida que condiciona a queda de um objeto de uma superfície geológica para outra, ocorre a erosão (Benes et al, 2001; St'ava et al, 2008; Jákó et al, 2011; Galin et al, 2019; Gonçalves et al, 2020).

O padrão visto nos trabalhos que implementam a simulação de erosão térmica é a comparação do gradiente das alturas entre células adjacentes do cenário com a tangente do ângulo de talude. Na superação deste pelo outro, ocorre o deslocamento de sedimento. A quantidade de material transportado é proporcional à diferença de altitude entre as coordenadas envolvidas.

Uma forma de calcular essa proporção é a vista nos trabalhos de Benes et al (2001) e Jákó et al (2011). Neles, cada célula C possui um conjunto de 8 vizinhos, uma área a e uma altura h. Calculando $H = \max\{h - h_i, i = 1, ..., 8\}$, em que h_i é a altura da célula vizinha C_i de C, o volume que sai de C e é distribuído entre seus adjacentes é dado por:

$$\Delta S = a\frac{H}{2}. \tag{2.16}$$

É importante ressaltar que é dito em ambos os trabalhos que um valor maior de ΔS gera oscilações indesejadas no resultado final.

Para uma célula C_i vizinha de C e com altitude h_i pertencente ao conjunto $A = \left\{h_i \,|\, h_i - h < 0,\ \mathrm{tg}(\alpha) < \frac{h-h_i}{d}, 1 \leq i \leq 8\right\}$, com α sendo o ângulo de talude e d a distância horizontal entre as células C e C_i, o volume de sedimento recebido é dado por:

$$\Delta S_i = \Delta S \frac{h_i}{\sum_{\forall h_k \in A} h_k}, \tag{2.17}$$

ou seja, cada coordenada vizinha de C que atende aos requisitos de recebimento de sedimento da erosão térmica recebe uma proporção do total de volume que a célula perderia. Essa proporção é determinada pelo resultado da divisão da altitude da célula C_i pelo somatório das alturas pertencentes ao conjunto A.

A erosão térmica é, juntamente com a erosão hidráulica, um dos métodos mais explorados dos trabalhos da subcategoria de simulações de física. Isso se dá pois ambos os intemperismos são os que geram mudanças mais significativas no ambiente (Jákó et al, 2011; Gonçalves et al, 2020). Muitos trabalhos nesse campo utilizam as duas simulações no mesmo sistema, a exemplo de St'ava et al (2008), Jákó et al (2011), Cristea et al (2015) e Gonçalves et al (2020).

3 Simulação de erosão

Neste capítulo são discutidos os detalhes associados à implementação dos métodos de simulação de erosão hidráulica e térmica apresentados no Capítulo 2. As estruturas de dados usadas são detalhadas, bem como os passos de implementação de cada simulador.

3.1 Estruturas de Dados

O sistema implementado possui três classes: `ShallowWater`, `Terreno` e `Erosao`. As três possuem grades 2D com o mesmo número N de linhas e colunas, sendo N um número arbitrário. A renderização do terreno é feita em um espaço cartesiano cujo plano se encontra no eixos x e z e as alturas y são controladas pelos valores presentes nas grades do sistema.

A primeira classe tem como principal atributo uma grade que representa a *shallow water*. Cada célula desta grade possui a altura da água na coordenada do cenário correspondente a ela. A classe também possui uma grade de fluxos cujas células calculam a quantidade de água que sai à direita, à esquerda, acima e abaixo de suas localizações. Esta grade possui, em cada célula, os valores de seus quatro fluxos de saída. Por fim, ainda existem duas grades, u e v, que representam o campo de velocidades da *shallow water*. As operações dessa estrutura de dados reproduzem a movimentação do fluido no sistema.

A segunda classe visa representar o terreno e tem como atributo principal um mapa de alturas t. Ainda existe um mapa de alturas t_0 que representa o estado do terreno no final do laço de execução anterior ao atual. Este segundo mapa de alturas é auxiliar para o simulador de erosão térmica. As operações presentes nessa classe consistem nos algoritmos de geração procedural de terrenos.

A terceira classe é a responsável pelas simulações de erosão do sistema. Ela possui instâncias das classes `ShallowWater` e `Terreno`, uma vez que precisa de ambas para o processo erosivo. Ela ainda possui constantes utilizadas nas erosões hidráulica e térmica

citadas na Seção 2.4.

Por fim, a classe de simulação de erosão também possui os atributos s e s_0, importantes para o algoritmo de erosão hidráulica. O s consiste em uma grade que representa os sedimentos dissolvidos na água nos diferentes pontos do sistema. A grade s_0 representa a quantidade de sedimentos dissolvidos na água no final do laço de execução anterior ao atual. As operações dessa classe envolvem os algoritmos de erosão hidráulica e térmica.

3.2 Erosão Hidráulica

A implementação da erosão hidráulica consiste na movimentação da *shallow water*, seguida da operação de deposição e dissolução de sedimentos, finalizando com o transporte de sedimentos.

3.2.1 Shallow water

A *shallow water* é implementada com uma função que controla a saída e outra que controla a entrada de água nas células da matriz que representa o fluido.

A primeira função calcula os fluxos de saída para cima, para baixo, para a esquerda e para a direita das células da matriz que representa a *shallow water*. Esse cálculo é feito através da diferença de alturas entre as células adjacentes.

Uma célula C arbitrária de coordenadas (x, y) tem sua altura calculada pela Equação (2.11). A diferença de alturas ΔA entre a célula C e suas adjacentes é dada por $A(x, y) - A(x + i, y + j)$, com com $i, j \in [-1, 1]$, $i, j \in \mathbb{Z}$ e $|i| \neq |j|$.

Com esse valor, é calculada a aceleração da água entre a célula C e cada uma de suas vizinhas por meio da Equação (2.13). Os valores da gravidade g, do comprimento do cano imaginário l e da densidade da água ρ são previamente determinados no construtor da classe `ShallowWater` e não têm seus valores modificados em momento algum.

Com a aceleração calculada nas quatro direções para a célula C, é possível calcular seus quatro fluxos de saída. Para isso, os valores armazenados na grade de fluxos da *shallow water* são somados com o valor previamente obtido pela Equação (2.13) multipli-

cado pela constante Δt do sistema e pela área da seção do cano que une as duas células envolvidas no fluxo, reproduzindo assim a Equação (2.14).

Com todos os fluxos obtidos, é importante calcular um fator K que controla a porcentagem de saída de água da célula. Para isso, é utilizada a Equação (2.15). O fator K é então multiplicado por todos os quatro fluxos de saída da célula.

Por fim a altura da água na célula C é atualizada na segunda função de tratamento da *shallow water*. Primeiro a soma de todos os fluxos de saída da célula C é salva em um valor f_{out}. Em seguida, o valor f_{in} é calculado com a seguinte equação:

$$f_{in} = f_l(x+1, y) + f_r(x-1, y) + f_b(x+1, y) + f_t(x, y-1), \tag{3.1}$$

em que f_l, f_r, f_b e f_t são os fluxos definidos na Seção 2.4.1.

Os valores f_{out} e f_{in} são então subtraídos, multiplicados por Δt e divididos pela área da célula da *shallow water*. O valor obtido a partir dessas operações é então somado à altura atual da *shallow water* na célula C.

Depois de todas essas operações, ainda são calculadas duas variáveis vel_u e vel_v com as expressões a seguir:

$$\begin{aligned} vel_u &= \frac{(f_r(x-1, y) - f_l(x, y) + f_r(x, y) - f_l(x-1, y))}{2}, \\ vel_v &= \frac{(f_t(x, y-1) - f_b(x, y) + f_t(x, y) - f_b(x, y+1))}{2}. \end{aligned} \tag{3.2}$$

Essas duas variáveis representam valores do campo de velocidades da *shallow water* e são utilizadas nas etapas seguintes das operações de simulação de erosão hidráulica. Seus valores são armazenados nas grades u e v da classe `ShallowWater`.

3.2.2 Deposição e dissolução de sedimento

O algoritmo da operação de deposição e dissolução de sedimento percorre cada célula do mapa de alturas. Em cada visita, as altitudes do cenário e a grade da quantidade de sedimentos são atualizados nos índices correspondentes. A forma como são feitos esses tratamentos é explicada a seguir.

Para uma célula C visitada, primeiro calcula-se o ângulo α de inclinação local

dela. Esse ângulo é obtido subtraindo a altura da célula C do mapa de alturas pela altura de suas vizinhas. A maior diferença Δa indica o seno ângulo α, que é calculado por meio da expressão $\frac{\Delta a}{\sqrt{\Delta a^2+d^2}}$, em que d é a distância entre os centros da célula C e de sua vizinha. Com este valor obtido, calcula-se a capacidade de sedimento da água na célula C. Esse valor é dado pela equação a seguir:

$$c_s = k_c \operatorname{sen}\alpha \sqrt{vel_u^2 + vel_v^2}, \tag{3.3}$$

em que k_c é a constante de capacidade de sedimento e vel_u e vel_v são os valores da velocidade da água armazenados nas grades u e v da *shallow water*.

Com o valor da capacidade de sedimento da célula, verifica-se se este é menor que a quantidade de sedimento dissolvido na célula da grade s cuja coordenada é a mesma de C. Em caso positivo o sistema deve depositar sedimento, então são aplicadas as Equações (2.7) e (2.8). Em caso negativo o sistema deve dissolver sedimento, então são aplicadas as Equações (2.9) e (2.10). O Algoritmo 3.1 reproduz a implementação do processo de deposição e dissolução de sedimentos feita no sistema.

Algoritmo 3.1: Código-fonte da deposição e dissolução de sedimento da erosão hidráulica

```
1  void erosao_hidraulica_deposita_dissolve(){
2
3      float uV;
4      float vV;
5
6      for(int i=1; i<N-1; i++)
7      for(int j=1; j<N-1; j++)
8      {
9         uV = agua->u(i,j);
10        vV = agua->v(i,j);
11
12        float b[4];
13         b[0] = alturas(i,j).y - alturas(i+1,j).y;
14         b[1] = alturas(i,j).y - alturas(i-1,j).y;
15         b[2] = alturas(i,j).y - alturas(i,j+1).y;
16         b[3] = alturas(i,j).y - alturas(i,j-1).y;
17
18         float maxB = max(max(b[0],b[1]),max(b[2],b[3]));
19         float d = (10.0/float(N));
20         float sinAlpha = maxB/(sqrt(maxB*maxB + d*d));
21
22         sinAlpha = std::max(sinAlpha,0.1f);
23
24         float constSed= kc*sinAlpha*sqrtf((uV*uV)+(vV*vV));
```

```
25
26              if(constSed<=sedimento_suspenso(i,j)){
27                  depositaSedimento(i,j,sedimento_suspenso(i,j) -
                        constSed);
28              }else {
29                  dissolveSedimento(i,j,constSed -
                        sedimento_suspenso(i,j));
30              }
31          }
32  }
33
34  void dissolveSedimento(int i,int j, float ds){
35
36      if(alturas(i,j).y>0)
37      {
38          sedimento_suspenso(i,j) += ks*ds;
39          alturas(i,j).y -= ks*ds;
40      }
41  }
42
43  void depositaSedimento(int i,int j, float ds){
44
45      if(sedimento_suspenso(i,j)>0)
46      {
47          sedimento_suspenso(i,j) -= kd*ds;
48          alturas(i,j).y += kd*ds;
49      }
50  }
```

3.2.3 Transporte de sedimento

Tal como é sugerido por Mei et al (2007), o transporte de sedimento no sistema é realizado por meio de uma advecção semi-lagrangiana inspirada nos trabalhos de Stam et al (1999, 2003). A estrutura de dados que soluciona a erosão do sistema possui uma matriz de sedimentos suspensos, s, e uma matriz de sedimentos suspensos do laço de execução anterior ao atual, s_0. O algoritmo percorre cada célula da matriz de sedimentos suspensos s_0. Se em uma dada célula C de coordenadas (C_x, C_y) a velocidade no eixo x é vel_u e no eixo y é vel_v, tem-se um ponto p cujas coordenadas (p_x, p_y) são dadas por:

$$\begin{aligned} p_x &= C_x - vel_u \Delta t, \\ p_y &= C_y - vel_v \Delta t, \end{aligned} \tag{3.4}$$

em que Δt é o intervalo de tempo utilizado na simulação. A quantidade de sedimentos nas coordenadas (p_x, p_y) da grade s_0 será então transportada para o ponto (C_x, C_y) na grade s.

Uma vez que os valores obtidos para p_x e p_y nem sempre são coordenadas identificadas com os centros de células das grades s_0 e s, para calcular a quantidade de sedimento que será transportado para a célula C é feita uma interpolação bilinear dos valores de s_0 cujos pontos da grade rodeiam a coordenada (p_x, p_y).

Os passos descritos definem a advecção semi-lagrangiana, que calcula para quais pontos do cenário os sedimentos serão transportados. O cálculo é feito baseando-se no posicionamento dos materiais em s_0 e salvando as novas localizações em s. Esses cálculos são feitos baseando-se nas velocidades vel_u e vel_v das grades u e v, respectivamente, da *shallow water*.

3.2.4 Evaporação da shallow water

A função de evaporação da *shallow water*, $f(w)$, nada mais é do que a subtração da altura atual da água, w, por um fator de evaporação, k_e, a cada laço de execução do sistema. Caso a subtração $w - k_e$ resulte em um número negativo, a altura w tem seu valor igual a 0.

$$f(w) = \begin{cases} w - k_e, & w - ke \geq 0 \\ 0, & w - k_e < 0 \end{cases} \tag{3.5}$$

3.2.5 Adição de volume de água ao sistema

Existem duas formas de adicionar água ao sistema. A primeira é por meio de chuva e a segunda por meio de um foco de água. A adição por meio da chuva consiste em somar um valor arbitrário à altura da água em pontos aleatórios do sistema a cada laço de iteração. A adição por meio de um foco de água consiste em somar uma quantidade arbitrária à altura da água em um ponto fixo do sistema.

3.3 Erosão Térmica

O algoritmo de erosão térmica percorre todas as células do mapa de alturas. Para uma célula C, são declarados dois *arrays* b e ΔS_{new}. Cada espaço de b representa a diferença de alturas entre a célula C e uma de suas oito células vizinhas. Cada espaço de ΔS_{new} representa um valor que será somado à altura da célula vizinha de C no final da execução do algoritmo. Se a altura de uma célula vizinha é maior que a altura de C, seu valor nos *arrays* b e ΔS_{new} é ignorado.

O maior valor do *array* b é então utilizado para calcular a Equação (2.16). Com isso, a quantidade de sedimento que será retirada da célula C é calculada. A área a da célula é obtida pela expressão $\left(\frac{10}{N}\right)^2$, uma vez que N é o número de linhas ou colunas do mapa de alturas e 10 é a dimensão do cenário definida na implementação do sistema.

As oito células adjacentes de C então são visitadas. Para cada uma dessas células, se sua altitude é menor que a altura de C e se o cálculo da tangente da inclinação entre a célula C e sua vizinha, dado por $\frac{b}{d}$, em que d é a distância horizontal entre as duas células, for menor ou igual à tangente do ângulo de talude, 45°, o índice de ΔS_{new} correspondente a esta célula recebe o valor da Equação (2.17) e a célula C perde a mesma quantidade em altura. Caso um dos requisitos citados não seja atendido, o índice de ΔS_{new} recebe o valor 0.

Por fim, cada novo valor de ΔS_{new} é somado à altura das células vizinhas de C no mapa de alturas. É importante ressaltar que a estrutura de dados do sistema possui dois mapas de alturas: o h_0 e o h. O h_0 representa o mapa de alturas do laço de execução anterior e é com ele que são calculados os valores salvos em b e ΔS_{new}. A etapa da soma dos valores de ΔS_{new} às alturas das células vizinhas de C é feita sobre o mapa de alturas h, que é o mapa de alturas atualizado. O Algoritmo 3.2 é a implementação do simulador de erosão térmica no sistema.

Algoritmo 3.2: Código-fonte da erosão térmica

```
1 void erosaoTermica(){
2
3     terreno->alturas0 = terreno->alturas;
4     float d = (10.0/float(N));
5     float a = d*d;
6
```

```
7       for(int i=0; i<N; i++)
8       for(int j=0; j<N; j++)
9       {
10          float b[8];
11          float newdS[8];
12
13
14          b[0] = altura(i,j).y - altura(i+1,j).y;
15          b[1] = altura(i,j).y - altura(i-1,j).y;
16          b[2] = altura(i,j).y - altura(i,j+1).y;
17          b[3] = altura(i,j).y - altura(i,j-1).y;
18          b[4] = altura(i,j).y - altura(i+1,j+1).y;
19          b[5] = altura(i,j).y - altura(i+1,j-1).y;
20          b[6] = altura(i,j).y - altura(i-1,j+1).y;
21          b[7] = altura(i,j).y - altura(i-1,j-1).y;
22
23          float H = max(b);
24
25          float sum;
26          for(int k=0; k<8; k++)
27              b[k]>0?sum+=b[k]:sum+=0;
28
29          float dS = dt*H*0.5f*a;
30
31          for(int k=0; k<8; k++)
32          {
33              float talus = b[k]/d;
34
35              if(b[k]>0 && talus>1)
36              {
37                  newdS[k] = dS*b[k]/sum;
38                  alturas(i,j).y -=newdS[k];
39              }else
40                  newdS[k] = 0;
41          }
42
43          alturas(i+1,j).y+=newdS[0];
44          alturas(i+1,j+1).y+=newdS[1];
45          alturas(i,j+1).y+=newdS[2];
46          alturas(i-1,j+1).y+=newdS[3];
47          alturas(i-1,j).y+=newdS[4];
48          alturas(i-1,j-1).y+=newdS[5];
49          alturas(i,j-1).y+=newdS[6];
50          alturas(i+1,j-1).y+=newdS[7];
51      }
52  }
```

4 Resultados e discussão

O sistema desenvolvido foi implementado com a linguagem de programação C++, utilizando a biblioteca gráfica *OpenGL*, com os experimentos conduzidos em um notebook com processador i5 4200U 2.6GHz, 6GB RAM e sem placa de vídeo dedicada.

Todos os experimentos de erosão térmica foram realizados sobre uma grade com 64 linhas e 64 colunas. O Δt aplicado para o algoritmo de erosão térmica foi de 180. O Δt aplicado para o algoritmo de erosão hidráulica foi de 0.1. Os experimentos com a erosão hidráulica e térmica sendo aplicadas simultaneamente tinham o Δt igual a 0.1. Mesmo com um alto valor de Δt para a erosão térmica, foi possível obter resultados satisfatórios. Já para os experimentos com as duas erosões combinadas, foi necessário reduzir o Δt, tendo em vista que a simulação de erosão hidráulica requer intervalos de tempo bem menores para a obtenção de bons resultados. Devido à diferença do Δt em cada experimento, o número de iterações destes variou. É importante ressaltar que os experimentos que envolveram a erosão hidráulica foram realizados com uma água corrente advinda do canto superior esquerdo do mapa de alturas.

Na Seção 4.1, serão discutidos brevemente alguns detalhes da implementação dos algoritmos procedurais. Nas demais seções, os resultados das simulações, com e sem o auxílio dos algoritmos procedurais, serão apresentados e analisados.

4.1 Algoritmos Procedurais

O mapa de alturas do sistema começa com todos os valores de sua grade iguais a 0. Para gerar um cenário, é aplicado sobre a estrutura de dados um dos algoritmos procedurais discutidos na Seção 2.3.

O algoritmo que executa o método de geração de colinas consiste em gerar n montanhas ao longo do cenário, com n sendo um número passado por parâmetro. Antes de gerar as elevações, são determinados para elas valores máximos e mínimos para seus raios, respectivamente r_{max} e r_{min}, e para suas alturas, respectivamente a_{max} e a_{min}. Com

isso, são sorteadas n células do mapa de alturas das quais surgem montanhas com alturas entre a_{min} e a_{max} e com raios da base entre r_{min} e r_{max}. Todas as montanhas são geradas com a Equação (2.1).

O algoritmo que executa o método de geração por falhas consiste em sortear valores aleatórios para o α e o p da Equação (2.2), de forma que sejam calculados os valores c da Equação (2.3) para todas as células do mapa de altura. O tratamento da altura da célula depende do valor c obtido e da função de tratamento adotada no método, que pode ser a padrão, a função seno ou a função cosseno (Equações (2.4), (2.5) e (2.6), respectivamente), passada por parâmetro na função. Esses passos são repetidos n vezes, com n sendo um valor passado no parâmetro da função.

O algoritmo do diamante-quadrado consiste em uma função recursiva que executa até um medidor c chegar no valor 1. Esse medidor inicialmente é igual a N e é dividido pela metade a cada passo da recursão. Junto dele, um valor d_{height}, que começa com o valor 20, também é dividido pela metade a cada recursão. A função se inicia determinando um valor de altura aleatório dentro do intervalo $[0, 40]$ para as quatro células dos cantos do mapa de alturas. Em seguida, a recursão se inicia. A cada iteração, as funções do passo do quadrado e do passo do diamante são chamadas e os valores de c e d_{height} são atualizados.

4.1.1 Normalizando o terreno

Uma vez que os algoritmos de geração procedural podem criar montanhas de alturas muito elevadas, chegando ao ponto de as figuras não serem mais visíveis pelo usuário do sistema, é importante normalizar os valores gerados neste processo.

Para fazer isso, basta encontrar o pico mais elevado e o menos elevado de todo o mapa de alturas para, em seguida, aplicar a seguinte função em cada célula da grade:

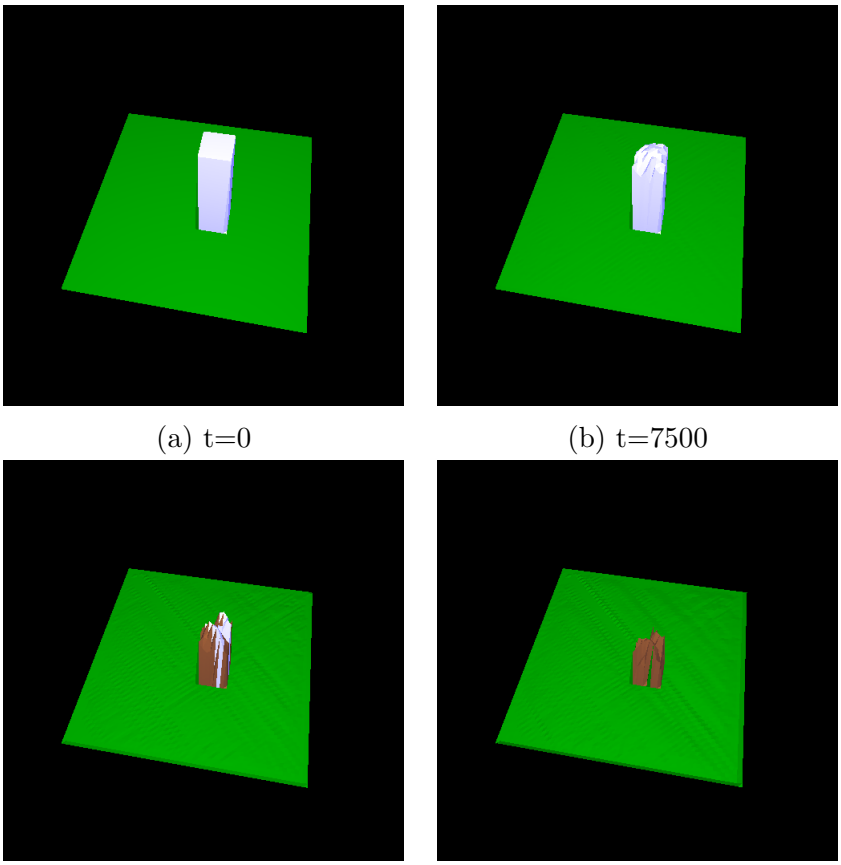
$$a_{new}(p) = \frac{a(p) - a_{min}}{a_{max} - a_{min}}, \tag{4.1}$$

com a_{new} sendo a altura da célula no ponto p após a normalização, $a(p)$ a altura da célula no ponto p antes da normalização e a_{max} e a_{min} a maior e a menor alturas, respectivamente, do mapa antes da normalização. Dessa forma, todos os valores do mapa de alturas estarão

no intervalo $[0, 1]$. Caso seja interessante um intervalo i maior, basta multiplicar cada altura do mapa pelo valor máximo do intervalo.

4.2 Geração procedural utilizando simulação de erosão

Primeiramente, foi analisada a possibilidade de gerar terrenos proceduralmente por meio da aplicação dos simuladores de erosão em entradas arbitrárias para o mapa de alturas que não se parecem em nenhum aspecto com um terreno. A primeira entrada se trata de um monólito no formato de um paralelepípedo localizado no centro do cenário. Como é visível na Figura 4.1, quando o objeto sofre apenas a ação da erosão hidráulica, seu topo é desgastado pela água. Ao longo do tempo a altura da estrutura diminui. Paralelamente, nota-se um maior detalhamento do solo ao redor do poliedro, o que é resultado do transporte dos sedimentos de seu topo para outras partes do mapa de alturas.



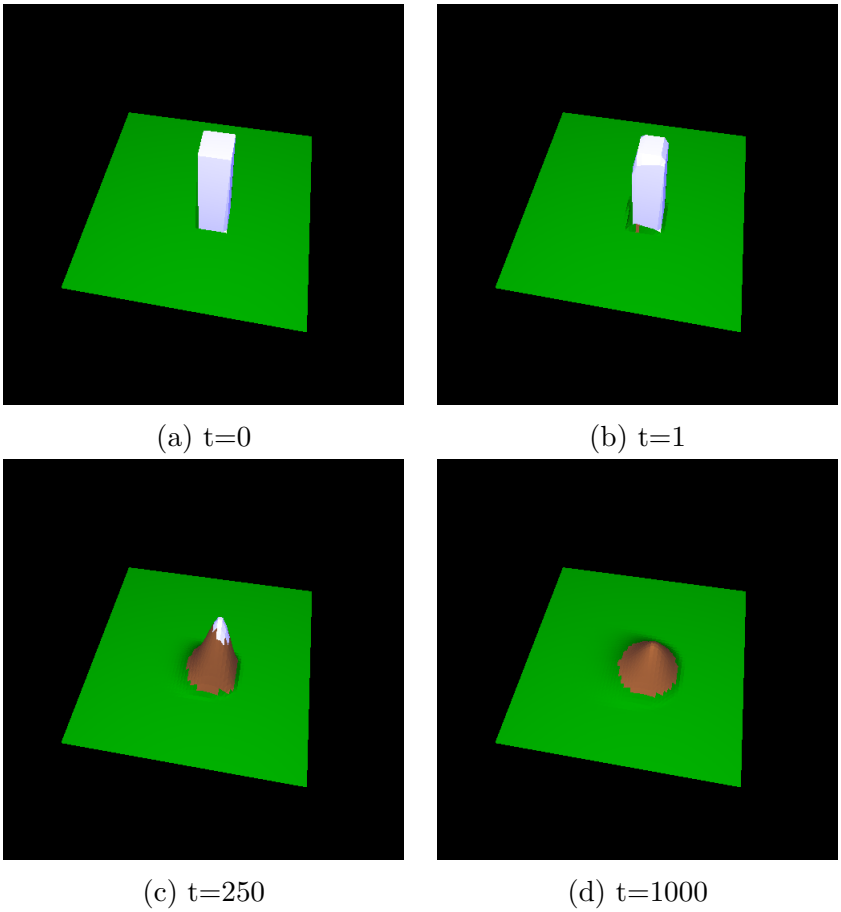
(a) t=0 (b) t=7500

(c) t=30000 (d) t=50000

Figura 4.1: Erosão hidráulica de um monólito ao longo de 50000 iterações.

Quando o monólito é submetido apenas à erosão térmica, nota-se uma tendência

do mesmo de deixar sedimentos nas bordas de sua base. Ao longo do tempo, o objeto assume uma característica montanhosa.



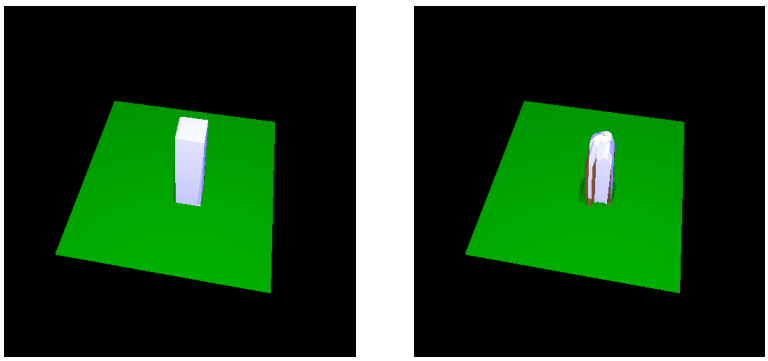
(a) t=0 (b) t=1

(c) t=250 (d) t=1000

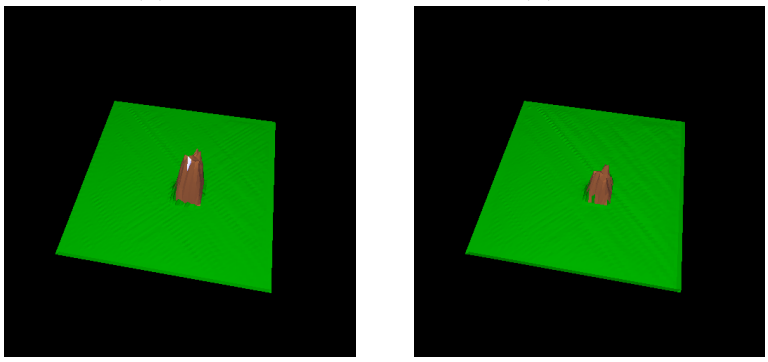
Figura 4.2: Erosão térmica do monólito ao longo de 1000 iterações.

Inicialmente é possível notar um alargamento da base da estrutura e um afunilamento de seu topo. Entretanto, sua altura não diminui nas primeiras iterações. Isso acontece pois, para haver diminuição na altura do solo, é necessário que a tangente da inclinação do mesmo seja maior que a tangente do ângulo de talude, o que não acontece enquanto o topo da estrutura for plano.

Como observado na Figura 4.3, quando se aplica a erosão hidráulica e térmica ao mesmo tempo no terreno, nota-se que a estrutura muda ao longo do tempo da mesma forma que quando é submetida apenas à erosão hidráulica. No entanto, é possível observar que suas bases se expandem e seu topo afunila tal como na erosão térmica. Além disso, a erosão hidráulica faz com que o espaço em torno do objeto possua formas mais irregulares. Isso se dá pelo transporte dos sedimentos no topo do monólito e daqueles depositados no solo devido à erosão térmica.

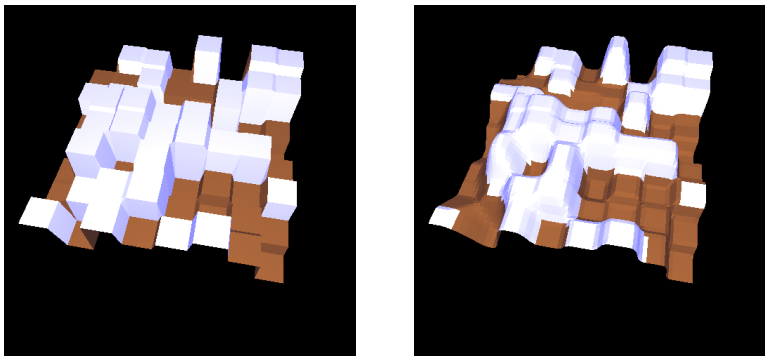


(a) t=0 (b) t=7500

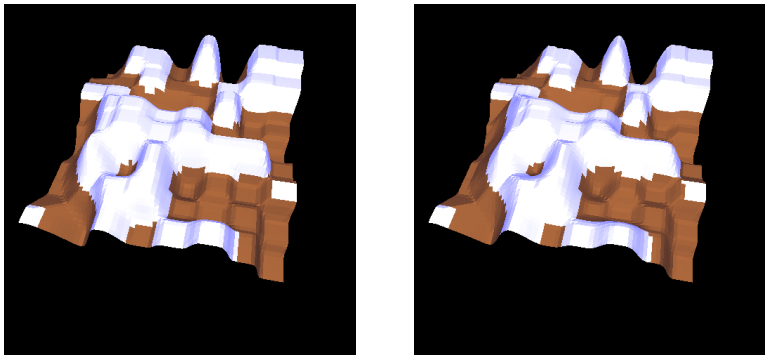


(c) t=30000 (d) t=50000

Figura 4.3: Monólito sofrendo os efeitos das erosões hidráulica e térmica combinadas ao longo de 50000 iterações.



(a) t=0 (b) t=500

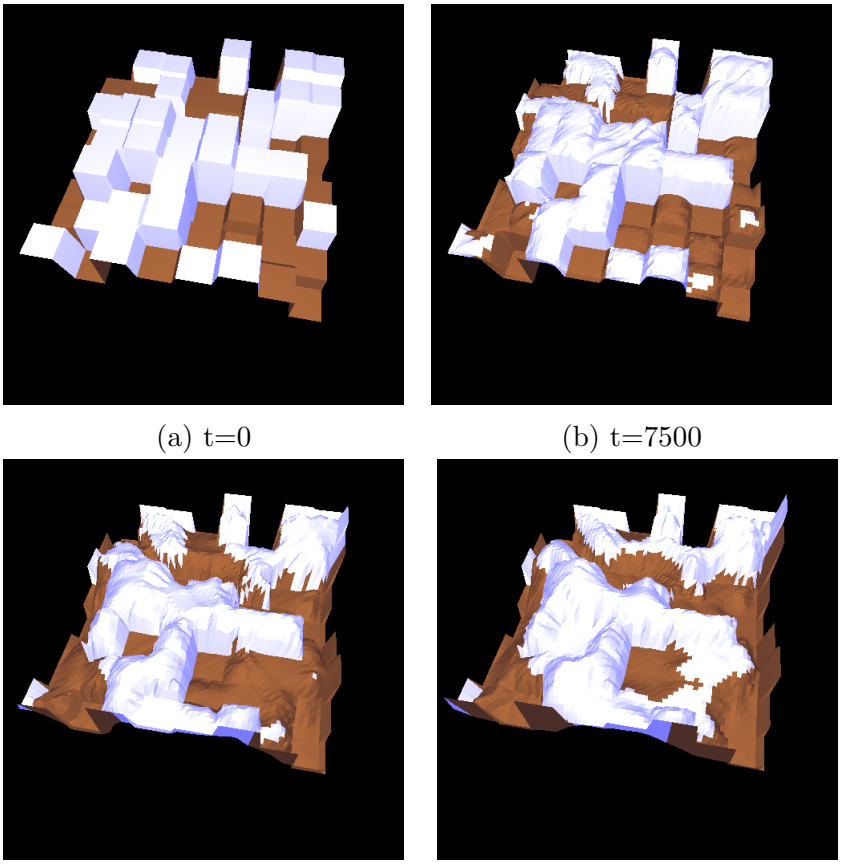


(c) t=1000 (d) t=2000

Figura 4.4: Conjunto de paralelepípedos sofrendo os efeitos da erosão térmica ao longo de 2000 iterações.

O segundo experimento foi feito sobre uma coleção de monólitos sem espaçamento horizontal entre eles. Como pode ser visto na Figura 4.4, aplicando apenas o simulador de erosão térmica nesta entrada, novamente os monólitos tendem a assumir formas montanhosas. Como vários deles foram criados lado a lado, há a tendência da formação de uma cadeia de montanhas, o que demonstra que é possível gerar um terreno proceduralmente apenas com o simulador de erosão térmica.

Quando o mesmo tipo de entrada é submetido ao algoritmo de erosão hidráulica, o resultado, novamente, é cadeia de montanhas. Dessa forma, conclui-se que também é possível gerar novos terrenos proceduralmente utilizando apenas o simulador de erosão hidráulica.



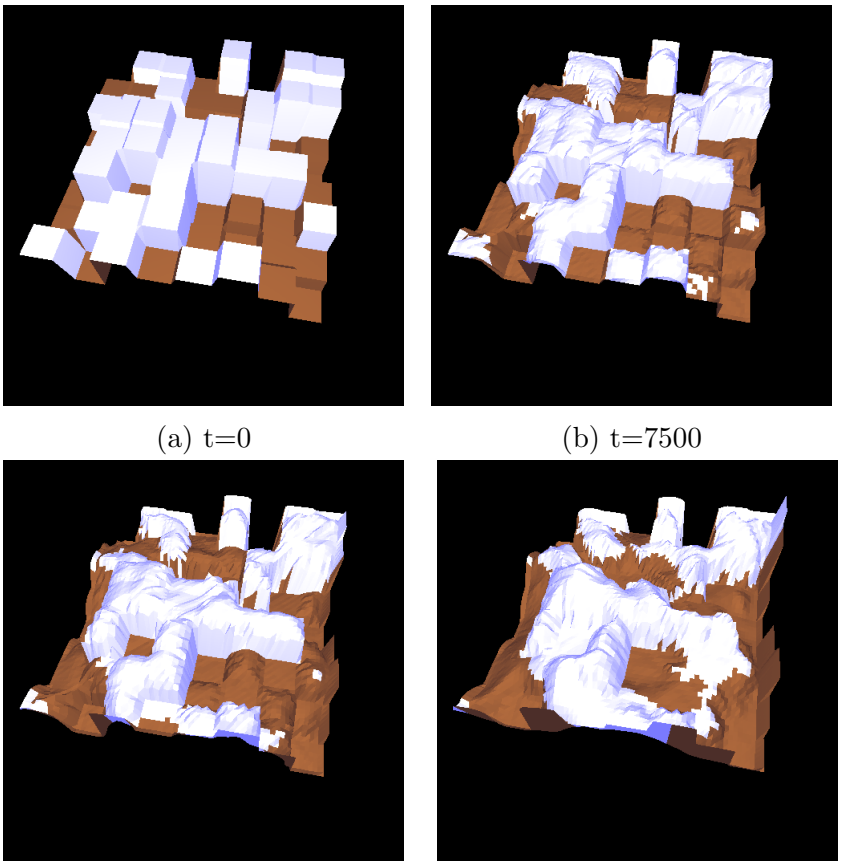
(a) t=0 (b) t=7500

(c) t=30000 (d) t=50000

Figura 4.5: Conjunto de paralelepípedos sofrendo os efeitos da erosão hidráulica ao longo de 50000 iterações.

Aplicando os dois métodos de erosão neste tipo de entrada, nota-se, na Figura 4.6, que o resultado final é muito parecido com o da erosão hidráulica aplicada sozinha, com algumas pequenas diferenças. Entre as possíveis causas disso, pode-se dizer que a erosão térmica, em comparação com a erosão hidráulica, é mais lenta e depende de mais fatores

para ter um resultado destacado. Dessa forma, um ponto que sofreria a ação da erosão térmica, por exemplo, pode ter seu ângulo de inclinação local modificado pela erosão hidráulica de forma que sua tangente seja menor que a tangente do ângulo de talude. Com isso, os efeitos da erosão hidráulica se sobressaem aos efeitos da erosão térmica.



(a) t=0 (b) t=7500

(c) t=30000 (d) t=50000

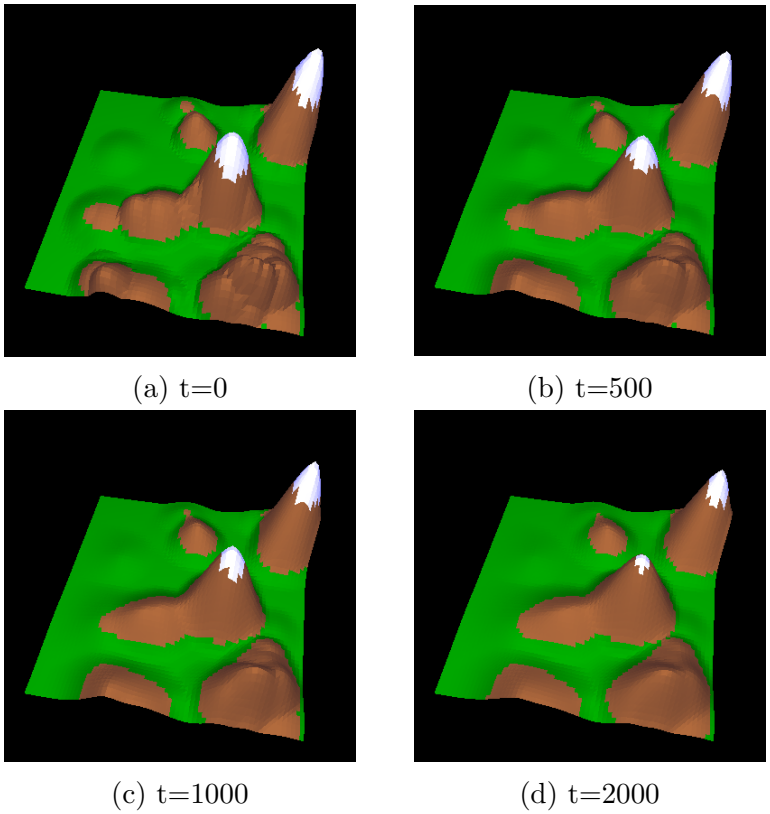
Figura 4.6: Conjunto de monólitos sofrendo os efeitos das erosões hidráulica e térmica ao longo de 50000 iterações.

4.3 Simuladores de erosão em terrenos procedurais

Na sequência, foram realizados experimentos aplicando os algoritmos de erosão sobre os resultados obtidos pelos algoritmos procedurais descritos no Capítulo 2. O objetivo era verificar a contribuição da erosão para a melhoria dos terrenos gerados, conferindo um caráter mais natural.

4.3.1 Algoritmo de colinas

Quando o terreno gerado pelo método de geração de colinas é submetido à erosão térmica, nota-se uma tendência de suas elevações diminuírem de altura, como ocorre com a montanha no centro da Figura 4.7. É importante ressaltar que nem todas as montanhas têm suas alturas diminuídas uma vez que algumas têm a tangente do ângulo de inclinação de suas estruturas menor que a tangente do ângulo de talude, o que impede o transporte de sedimentos.



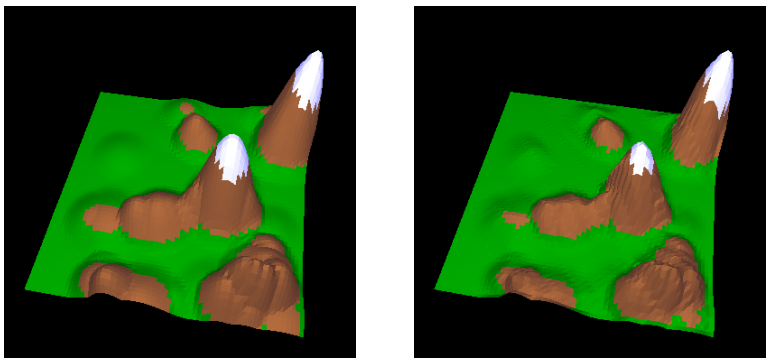
(a) t=0 (b) t=500

(c) t=1000 (d) t=2000

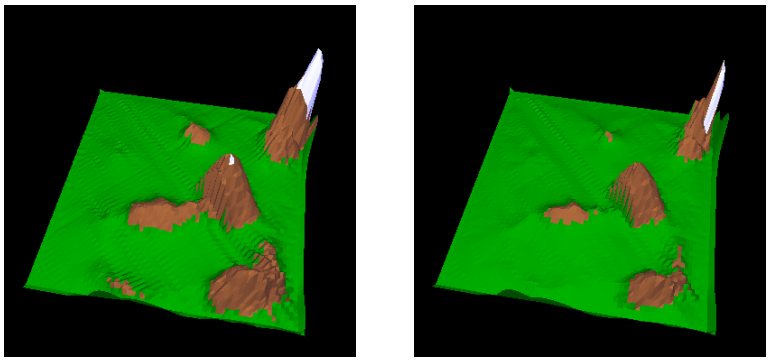
Figura 4.7: Terreno gerado com o método de geração de colinas sofrendo os efeitos do simulador de erosão térmica ao longo de 2000 iterações.

Observando a Figura 4.8, é possível notar que quando é aplicada no cenário apenas a erosão hidráulica, a montanha de seu centro é destruída ao final da simulação. Analisando a Figura 4.1b, nota-se que inicialmente as estruturas geológicas adquirem uma aparência rugosa, mas as estruturas iniciais do cenário se mantêm. A partir da Figura 4.1c, nota-se que uma grande quantidade das montanhas tiveram seus sedimentos dissolvidos na água, o que gera uma modificação expressiva no cenário quando comparada ao que ocorre quando se aplica apenas a erosão térmica.

Como é possível ver na Figura 4.9, a aplicação simultânea dos dois algoritmos

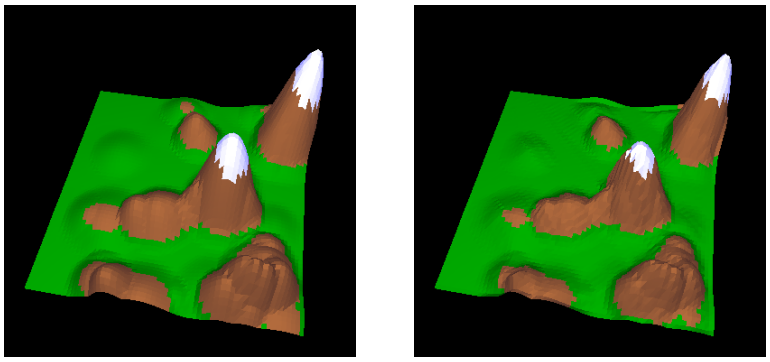


(a) t=0 (b) t=7500

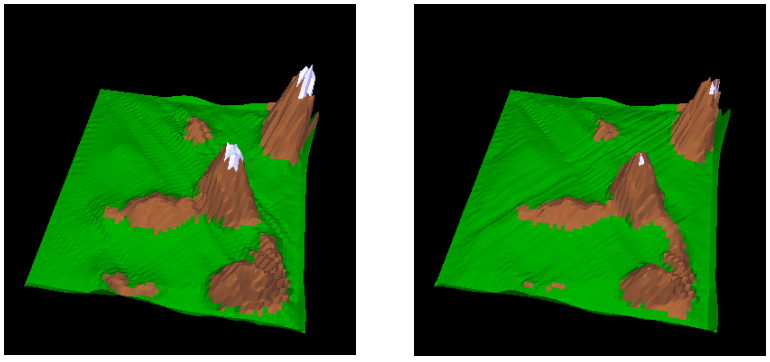


(c) t=30000 (d) t=50000

Figura 4.8: Terreno gerado pelo método de geração de colinas sofrendo os efeitos do simulador de erosão hidráulica ao longo de 50000 iterações.



(a) t=0 (b) t=7500



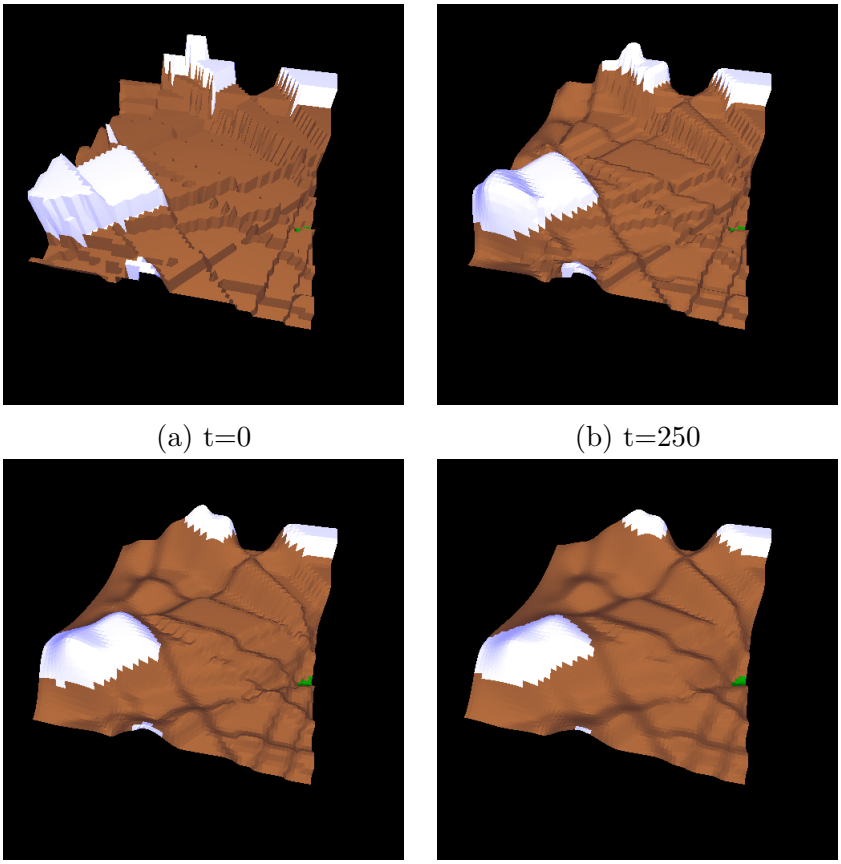
(c) t=30000 (d) t=50000

Figura 4.9: Terreno gerado pelo método de geração de colinas sofrendo os efeitos dos simuladores de erosão hidráulica e térmica ao longo de 50000 iterações.

de erosão no terreno pelo método de geração de colinas faz com que as montanhas do centro do cenário sejam novamente desgastadas pela ação da erosão hidráulica. Apesar de o resultado ser muito parecido com a aplicação apenas da erosão hidráulica, nota-se a diminuição na altitude de algumas montanhas devida à erosão térmica. É o caso da montanha do canto superior direito da Figura 4.9d, cuja altitude diminuiu quando comparada com seu estado na Figura 4.9a.

4.3.2 Algoritmo de falhas

O terreno gerado pelo algoritmo de geração por falhas com a função de tratamento padrão gera cenários que não aparentam naturalidade devido ao aspecto geométrico de suas estruturas. A aplicação da erosão térmica nesses cenários suaviza o formato do terreno, tornando-o mais convincente e realista. Esse fenômeno é observável na Figura 4.10.



(a) t=0 (b) t=250

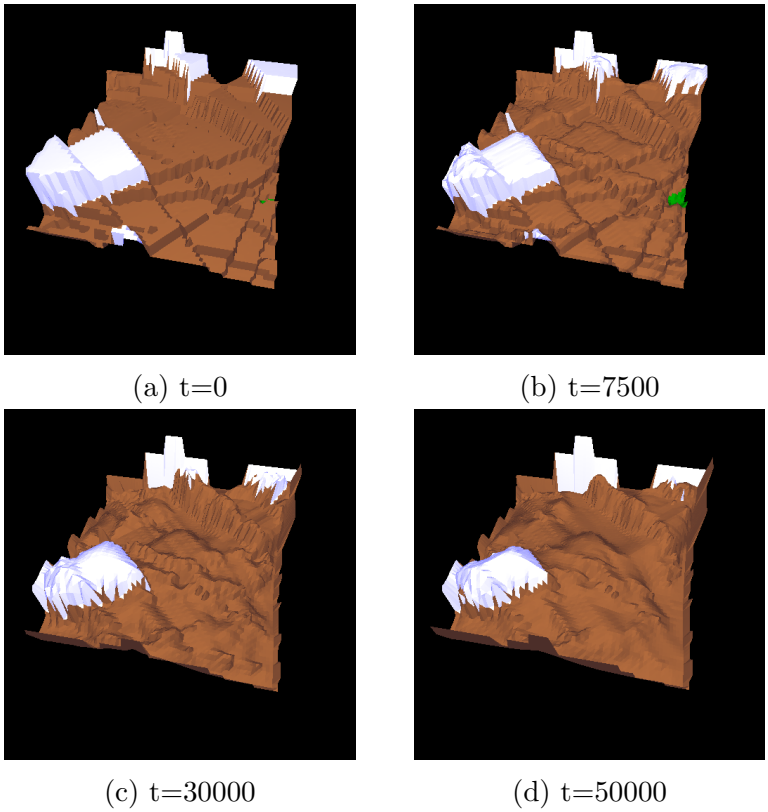
(c) t=1000 (d) t=2000

Figura 4.10: Terreno gerado com o método de geração por falhas sofrendo os efeitos do simulador de erosão térmica ao longo de 2000 iterações.

A aplicação da erosão hidráulica faz com que a aparência do terreno gerado pelo

método de geração por falhas fique mais natural. As partes com formatos geométricos adquirem um caráter mais irregular e desgastado, tal como um terreno real.

Quando submetido aos dois simuladores de erosão simultaneamente, o terreno apresenta as mesmas irregularidades da erosão hidráulica, porém algumas estruturas, tal como a elevação do canto inferior esquerdo do cenário, têm o formato suavizado pela erosão térmica. Os cantos do cenário sofrem menos com a ação da água, uma vez que a água corrente não passa por eles na mesma frequência que passa pelo centro do cenário. Dessa forma, ocorre uma atuação mais expressiva da erosão térmica sobre essas regiões.



(a) t=0 (b) t=7500

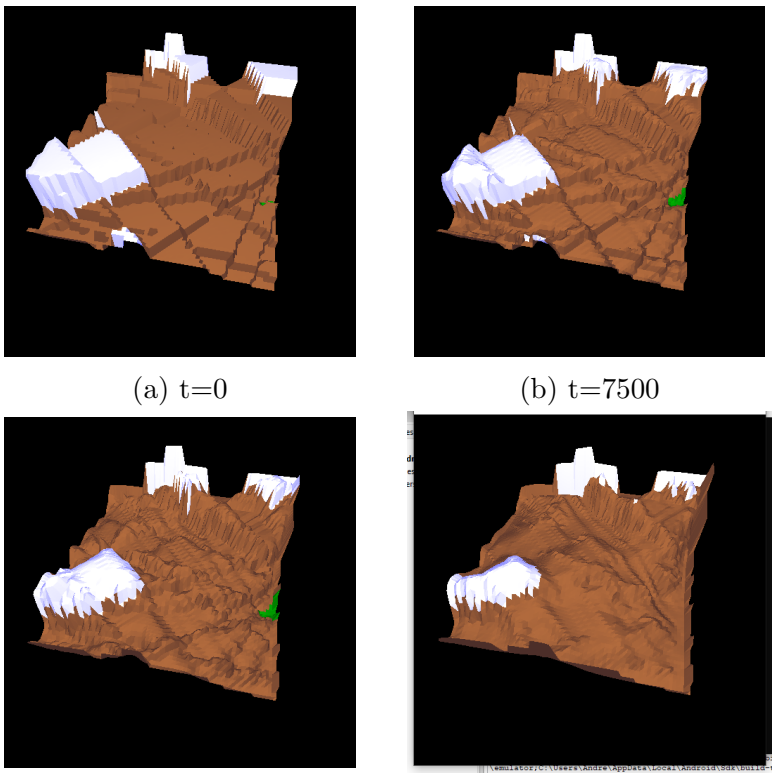
(c) t=30000 (d) t=50000

Figura 4.11: Terreno gerado pelo método de geração por falhas sofrendo os efeitos do simulador de erosão hidráulica ao longo de 50000 iterações.

4.3.3 Algoritmo diamante-quadrado

Analisando a Figura 4.13a, é perceptível que os terrenos gerados com o algoritmo do diamante-quadrado possuem estruturas geológicas pontiagudas. Com a aplicação da erosão térmica nele, nota-se uma suavização dessas partes do cenário como é visível nas Figuras 4.13b, 4.13c e 4.13d.

Também é perceptível a capacidade da erosão hidráulica de suavizar o terreno



(a) t=0 (b) t=7500

(c) t=20000 (d) t=50000

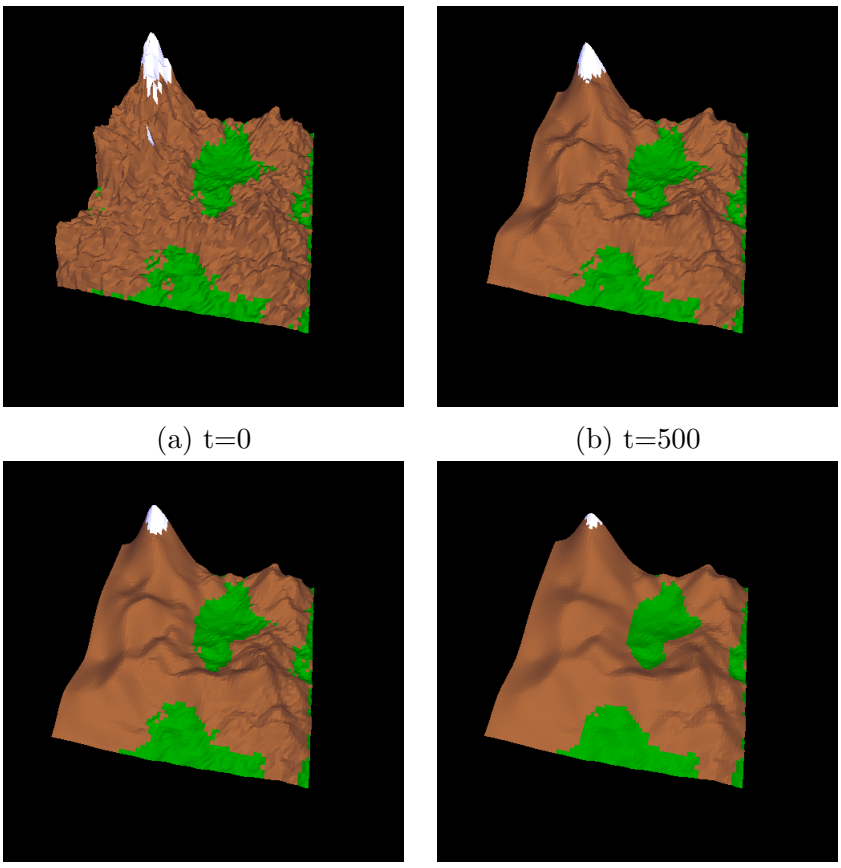
Figura 4.12: Terreno gerado pelo método de geração por falhas sofrendo os efeitos dos simuladores de erosão hidráulica e térmica ao longo de 50000 iterações.

gerado com o método do diamante quadrado. Na Figura 4.14 é possível perceber o efeito dessa suavização nas partes mais rugosas do cenário. Além disso, nota-se um depósito de sedimentos nas partes de menor altitude do cenário, representadas pela cor verde. Comparando as Figuras 4.14a e 4.14b com as Figuras 4.14c e 4.14d é possível perceber o desaparecimento das regiões verdes do cenário, já que sedimento é depositado sobre elas e sua cor é trocada para o marrom.

Como observado na Figura 4.15, as estruturas pontiagudas do terreno gerado pelo método do diamante-quadrado são suavizadas pelo simulador de erosão hidráulica. Os efeitos do simulador de erosão térmica são inexpressivos.

4.4 Discussão dos resultados

O principal efeito detectado do simulador de erosão térmica foi sua capacidade de suavizar terrenos. Isso é perceptível nos experimentos feitos com os terrenos dos monólitos, com o método de geração por falhas e com o método do diamante-quadrado, cujas mudanças



(a) t=0 (b) t=500

(c) t=1000 (d) t=2000

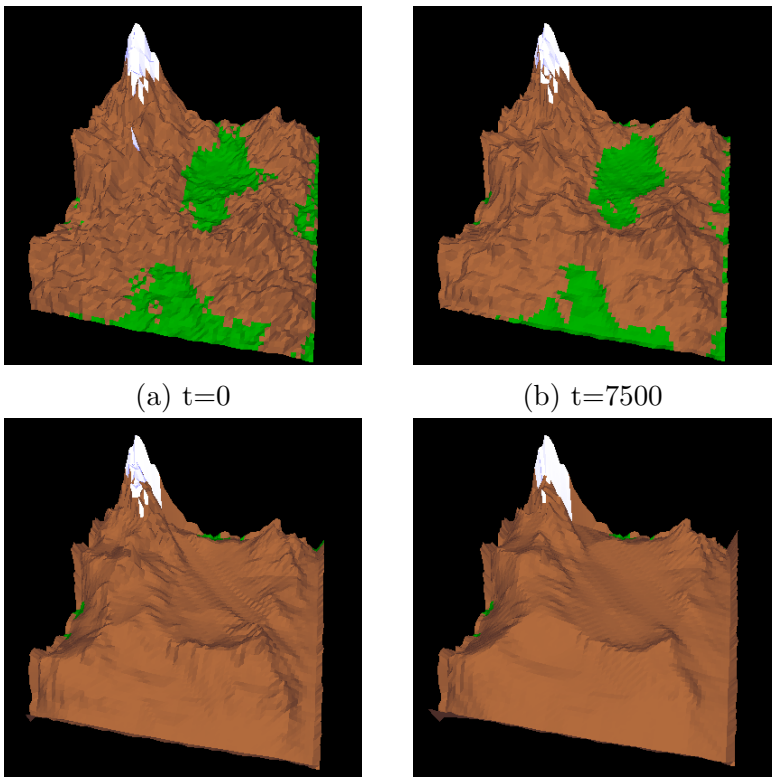
Figura 4.13: Terreno gerado com o método do diamante-quadrado sofrendo os efeitos do simulador de erosão térmica ao longo de 2000 iterações.

ocasionadas pela erosão térmica são mais pronunciadas. Contudo, este efeito pode não ser interessante em certos contextos. Por exemplo, pode não ser desejável perder o caráter rugoso dos cenários produzidos pelo algoritmo diamante-quadrado.

No caso do terreno criado com o método de geração por falhas, o efeito suavizador da erosão térmica funciona como uma alternativa a outras formas de suavização, uma vez que também é possível obter terrenos mais suaves utilizando as Equações (2.5) e (2.6) no lugar da Equação (2.4) para tratar as alturas do terreno.

Fora do contexto de terrenos previamente criados por algoritmos procedurais, a erosão térmica também se mostrou bastante interessante na geração de cadeias de montanhas a partir de objetos genéricos, como no caso do conjunto de monólitos da Seção 4.2. Isso permite a criação de terrenos de aspecto natural não somente através do preenchimento aleatório inicial do mapa de alturas (como feito no exemplo dos monólitos), como também a partir de um conjunto de estruturas artificiais definidas manualmente.

A erosão hidráulica também apresentou a capacidade de suavizar terrenos, como



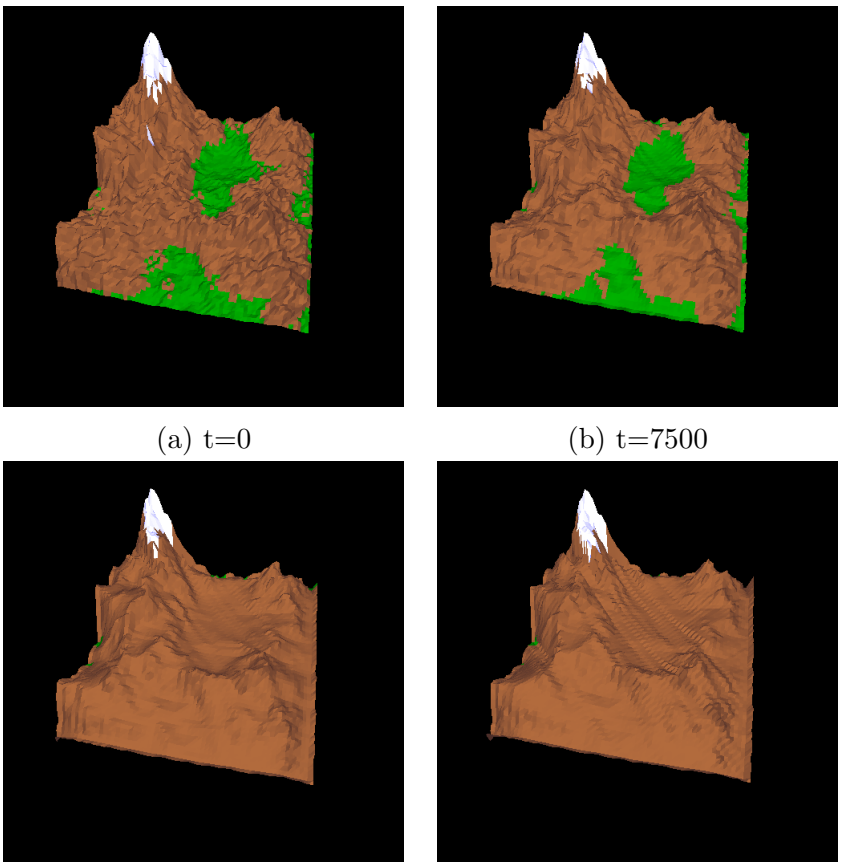
(a) t=0 (b) t=7500

(c) t=30000 (d) t=50000

Figura 4.14: Terreno gerado pelo método do diamante-quadrado sofrendo os efeitos do simulador de erosão hidráulica ao longo de 50000 iterações.

foi possível observar nos experimentos envolvendo o método do diamante-quadrado. Além disso, o simulador de erosão hidráulica também se mostrou capaz de gerar detalhamentos sutis nos terrenos gerados por algoritmos procedurais. Um exemplo disso é o visto na Figura 4.8b, em que a montanha do centro se mostra desgastada pela erosão e, assim, possui um aspecto mais natural quando comparada ao seu estado anterior, mostrado na Figura 4.8a.

Quando os dois algoritmos de erosão são aplicados simultaneamente, notou-se que os efeitos do simulador de erosão hidráulica se sobressaíram aos efeitos do simulador de erosão térmica. Entre os motivos de isso acontecer, pode-se deduzir que, quando os dois estão com um mesmo Δt, os efeitos da erosão hidráulica ocorrem mais rápido que os efeitos da erosão térmica. Como consequência, a erosão hidráulica atua sobre o terreno de forma a diminuir o ângulo de inclinação local dos pontos do cenário que seriam mais afetados pela erosão térmica. Além da maior velocidade da erosão hidráulica, esta ainda possui menos condições de ocorrência do que a erosão térmica. Se um cenário não possui pontos cujo ângulo de inclinação é maior que o ângulo de talude, a erosão térmica não ocorrerá.



(a) t=0 (b) t=7500

(c) t=20000 (d) t=30000

Figura 4.15: Terreno gerado pelo método do diamante-quadrado sofrendo os efeitos dos simuladores de erosão hidráulica e térmica ao longo de 30000 iterações.

A erosão hidráulica, por outro lado, modifica o cenário constantemente na presença de água no solo.

Os experimentos envolvendo o simulador de erosão hidráulica ainda mostram que é possível gerar novos terrenos com ele, como ocorreu com o conjunto de monólitos. Uma comparação interessante entre a cadeia de montanhas gerada pelas erosões hidráulica e térmica, separadamente, é a de que o primeiro cenário apresenta uma tendência de gerar elevações em lugares distintos da configuração inicial do terreno, ao passo que o segundo tende a manter as montanhas nas localidades em que se originaram, apenas com variações nas alturas dos picos e no alargamento das bases nesse pontos.

Por fim, é importante ressaltar que algumas pequenas variações de altura no cenário podem ser fruto de erros numéricos, e não exatamente resultado do processo de deposição de sedimentos da erosão hidráulica. Uma análise mais detalhada dos resultados do simulador implementado precisa ser realizada nesse sentido. Apesar disso, o simulador gera resultados condizentes com um processo erosivo, como pode ser observado nos

experimentos realizados.

5 Conclusão

Este trabalho se propôs a estudar o uso de algoritmos de simulação de erosão na geração de terrenos e no detalhamento de cenários gerados por algoritmos procedurais. Foram discutidos e explorados os conceitos que fundamentam os algoritmos de simulação de erosão hidráulica e térmica, além de alguns métodos de geração procedural de terrenos que possuem destaque na literatura, como o método de geração de colinas, o método de geração por falhas e o método do diamante-quadrado. Foi implementado um sistema que aplica os simuladores de erosão hidráulica e térmica a um terreno de entrada. Esse terreno, representado por um mapa de alturas, pode ser composto por um conjunto artificial de objetos arbitrários ou por um conjunto de montanhas previamente gerado. Experimentos foram conduzidos para analisar os efeitos da erosão hidráulica e térmica na criação de cadeias de montanhas e na modificação de terrenos gerados proceduralmente.

A partir dos resultados dos experimentos, foi possível verificar que os algoritmos de erosão são úteis para a construção de terrenos com aspecto natural a partir de objetos não-naturais, como demonstrado no experimento dos paralelepípedos. Embora experimentos com outros tipos de objetos não tenham sido realizados, é plausível esperar que resultados similares possam ser obtidos com objetos de diferentes formas. Enquanto a erosão térmica possui um efeito maior de suavização das estruturas, a erosão hidráulica adiciona irregularidades às superfícies em que atua, o que gera um aspecto mais natural a elas. A capacidade de suavização da erosão térmica também pode ser bastante interessante no aprimoramento de terrenos pré-existentes, especialmente naqueles gerados por métodos procedurais mais simples, como é o caso do método de geração por falhas com o uso da função padrão do cálculo de alturas. Em outros casos, essa suavização pode não ser desejada, como no caso do algoritmo diamante-quadrado discutido na Seção 4.4.

Em trabalhos futuros, é interessante a implementação de melhorias nos algoritmos do sistema. Entre elas, inclui-se investigar e aperfeiçoar o processo de deposição de sedimentos da erosão hidráulica, além de aplicar a erosão a terrenos gerados por outros métodos de geração procedural, como o ruído de Perlin. Outro caminho interessante seria

a implementação de um simulador de erosão eólica, utilizando o método de simulação de fluidos desenvolvidos por Stam et al (1999). O método de Stam et al (1999) é ideal para a simulação de gases e poderia ser adaptado para o transporte de sedimentos. Além disso, o uso de métodos de simulação de fluidos mais complexos que a *shallow water* para simular a água da erosão hidráulica também pode gerar resultados interessantes. Por fim, pode-se expandir o sistema simulador de erosão para trabalhar com terrenos representados por camadas, ou incluir na simulação fatores capazes de atenuar ou intensificar os efeitos da erosão, como a presença de vegetação no solo.

Bibliografia

Anh, N. H.; Sourin, A. ; Aswani, P. **Physically based hydraulic erosion simulation on graphics processing unit**. In: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, p. 257–264, 2007.

Benes, B.; Forsbach, R. **Layered data representation for visual simulation of terrain erosion**. In: Proceedings Spring Conference on Computer Graphics, p. 80–86. IEEE, 2001.

Beneš, B.; Roa, T. Simulating desert scenery. 2004.

Beneš, B.; Arriaga, X. **Table mountains by virtual erosion**. In: Proceedings of the Eurographics Workshop on Natural Phenomena (NPH), volume 15, 2005.

Beneš, B.; Těšínskỳ, V.; Hornyš, J. ; Bhatia, S. K. Hydraulic erosion. **Computer Animation and Virtual Worlds**, v.17, n.2, p. 99–108, 2006.

Benes, B. Real-time erosion using shallow water simulation. **The Eurographics Association**, 2007.

Bôcher, M. **Plane analytic geometry: with introductory chapters on the differential calculus**. H. Holt, 1915.

Chiba, N.; Muraoka, K. ; Fujita, K. An erosion model based on velocity fields for the visual simulation of mountain scenery. **The Journal of Visualization and Computer Animation**, v.9, n.4, p. 185–194, 1998.

Cordonnier, G.; Braun, J.; Cani, M.-P.; Benes, B.; Galin, E.; Peytavie, A. ; Guérin, E. **Large scale terrain generation from tectonic uplift and fluvial erosion**. In: Computer Graphics Forum, volume 35, p. 165–175. Wiley Online Library, 2016.

Cordonnier, G.; Galin, E.; Gain, J.; Benes, B.; Guérin, E.; Peytavie, A. ; Cani, M.-P. Authoring landscapes by combining ecosystem and terrain erosion simulation. **ACM Transactions on Graphics (TOG)**, v.36, n.4, p. 1–12, 2017.

Cristea, A.; Liarokapis, F. **Fractal nature-generating realistic terrains for games**. In: 2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games), p. 1–8. IEEE, 2015.

DeLoura, M. **Game programming gems**. Charles River Media, Inc., 2000.

Fernandes, A. R. **Terrain tutorial**. `http://www.lighthouse3d.com/opengl/terrain/index.php?fault`, 2006. Accessed: 2021-08-12.

Fournier, A.; Fussell, D. ; Carpenter, L. Computer rendering of stochastic models. **Communications of the ACM**, v.25, n.6, p. 371–384, 1982.

Gain, J.; Merry, B. ; Marais, P. **Parallel, realistic and controllable terrain synthesis**. In: Computer Graphics Forum, volume 34, p. 105–116. Wiley Online Library, 2015.

Galin, E.; Guérin, E.; Peytavie, A.; Cordonnier, G.; Cani, M.-P.; Benes, B. ; Gain, J. **A review of digital terrain modeling**. In: Computer Graphics Forum, volume 38, p. 553–577. Wiley Online Library, 2019.

Gonçalves, A. S.; dos Reis, D. S. Geração procedural de terrenos virtuais com aparência natural utilizando gpu. 2020.

Gu-ping, Z.; Meng, Z.; Xu-qiang, S. ; Zhao-hui, W. Realistic modeling and rendering of hydraulic erosion terrain. **Journal of Graphics**, v.41, n.2, p. 169, 2020.

Hawkins, B. **Real-Time Hydraulic and Thermal Erosion Simulation over Vegetated Terrains**. 2021. Tese de Doutorado - University of Nebraska at Omaha.

Jákó, B.; Tóth, B. **Fast hydraulic and thermal erosion on gpu.** In: Eurographics (Short Papers), p. 57–60. Citeseer, 2011.

Jones, M. D.; Farley, M.; Butler, J. ; Beardall, M. Directable weathering of concave rock using curvature estimation. **IEEE Transactions on Visualization and Computer Graphics**, v.16, n.1, p. 81–94, 2010.

Kamal, K. R.; Uddin, Y. S. **Parametrically controlled terrain generation**. In: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, p. 17–23, 2007.

Kass, M.; Miller, G. **Rapid, stable fluid dynamics for computer graphics**. In: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, p. 49–57, 1990.

Kelley, A. D.; Malin, M. C. ; Nielson, G. M. **Terrain simulation using a model of stream erosion**. In: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, p. 263–268, 1988.

Krištof, P.; Beneš, B.; Křivánek, J. ; Šťava, O. **Hydraulic erosion using smoothed particle hydrodynamics**. In: Computer Graphics Forum, volume 28, p. 219–228. Wiley Online Library, 2009.

Krs, V.; Hädrich, T.; Michels, D. L.; Deussen, O.; Pirk, S. ; Benes, B. Wind erosion: Shape modifications by interactive particle-based erosion and deposition. **The Eurographics Association**, 2020.

Mei, X.; Decaudin, P. ; Hu, B.-G. **Fast hydraulic erosion simulation and visualization on gpu**. In: 15th Pacific Conference on Computer Graphics and Applications (PG'07), p. 47–56. IEEE, 2007.

Monaghan, J. J. Simulating free surface flows with sph. **Journal of computational physics**, v.110, n.2, p. 399–406, 1994.

Müller, D.; Buchenau, C. ; Guthe, M. **Partial updates to accelerate gpu-based interactive hydraulic erosion**. In: Proceedings of the 32nd Spring Conference on Computer Graphics, p. 33–40, 2016.

Musgrave, F. K.; Kolb, C. E. ; Mace, R. S. The synthesis and rendering of eroded fractal terrains. **ACM Siggraph Computer Graphics**, v.23, n.3, p. 41–50, 1989.

Neidhold, B.; Wacker, M. ; Deussen, O. **Interactive physically based fluid and erosion simulation**. In: Eurographics workshop on natural phenomena, 2005.

Nystrom, B. **Terrain generation tutorial**. `https://www.stuffwithstuff.com/robot-frog/3d/hills/index.html`, 2002. Accessed: 2021-04-21.

Olsen, J. Realtime procedural terrain generation. 2004.

Perlin, K. An image synthesizer. **ACM Siggraph Computer Graphics**, v.19, n.3, p. 287–296, 1985.

Peytavie, A.; Galin, E.; Grosjean, J. ; Mérillou, S. **Arches: a framework for modeling complex terrains**. In: Computer Graphics Forum, volume 28, p. 457–467. Wiley Online Library, 2009.

Prusinkiewicz, P.; Hammel, M. **A fractal model of mountains and rivers**. In: Graphics Interface, volume 93, p. 174–180. Canadian Information Processing Society, 1993.

Paris, A.; Peytavie, A.; Guérin, E.; Argudo, O. ; Galin, E. **Desertscape simulation**. In: Computer Graphics Forum, volume 38, p. 47–55. Wiley Online Library, 2019.

Roudier, P.; Peroche, B. ; Perrin, M. **Landscapes synthesis achieved through erosion and deposition process simulation**. In: Computer Graphics Forum, volume 12, p. 375–383. Wiley Online Library, 1993.

Shaker, N.; Togelius, J. ; Nelson, M. J. **Procedural content generation in games**. Springer, 2016.

Skorkovská, V.; Kolingerová, I. ; Vanecek, P. **A unified curvature-driven approach for weathering and hydraulic erosion simulation on triangular meshes.** In: VISIGRAPP (1: GRAPP), p. 122–133, 2019.

Stam, J. **Stable fluids**. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, p. 121–128, 1999.

Stam, J. **Real-time fluid dynamics for games**. In: Proceedings of the game developer conference, volume 18, p. 25, 2003.

Šťava, O.; Beneš, B.; Brisbin, M. ; Křivánek, J. **Interactive terrain modeling using hydraulic erosion**. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, p. 201–210, 2008.

Sumner, R. W.; O'Brien, J. F. ; Hodgins, J. K. **Animating sand, mud, and snow**. In: Computer Graphics Forum, volume 18, p. 17–26. Wiley Online Library, 1999.