GERAÇÃO PROCEDURAL DE TERRENOS VIRTUAIS COM APARÊNCIA NATURAL UTILIZANDO GPU

Alex Seródio Gonçalves

Prof. Dalton Solano dos Reis – Orientador

1 INTRODUÇÃO

A forma como cidades e sociedades são construídas ao longo da história está diretamente relacionada com o relevo das paisagens da região. Essas paisagens estão em constante mudança, sendo influenciadas frequentemente não apenas por forças naturais, mas também por forças econômicas e culturais (PLIENINGER et al., 2015).

Algumas dessas mudanças podem se mostrar complexas de serem analisadas e estudadas no mundo real. Nesses casos, simulações computacionais possibilitam a criação e validação de modelos que representam estes cenários, permitindo assim a execução de experimentos, análise dos resultados e validação de teorias em um ambiente controlado (HEERMANN, 1990, p. 8). Estas simulações podem ser utilizadas para analisar os processos naturais e sociais causadores de mudanças na paisagem, sendo necessário, porém, a utilização de um cenário adequado à simulação.

Para garantir a utilização de terrenos naturais e possibilitar uma variedade de cenários diferentes para as simulações, é possível utilizar terrenos gerados de forma procedural. Este processo pode ser realizado através da utilização de modelos estocásticos e/ou modelos físicos. O primeiro permite a geração controlada de terrenos com características de relevo pseudoaleatórias (EBERT et al., 2003). Já o segundo realiza a criação de terrenos através da simulação de eventos físicos presentes na formação de terrenos reais, como processos de erosão que modificam o relevo do terreno (OLSEN, 2004). No caso dos modelos físicos os algoritmos exigem um poder de processamento considerável para atingir resultados realistas, tornando em muitos casos sua execução em tempo real impraticável (MEI; DECAUDIN; HU, 2007, p. 47).

Com os constantes avanços no desenvolvimento de *hardware*, a utilização da GPU (*Graphic Processing Unit*) para resolução de problemas computacionalmente complexos vem se tornando cada vez mais viável. Esta diferença fica evidente principalmente conforme o tamanho do dado a ser processado aumenta (HANGÜN; EYECIOĞLU, 2017, p. 34). Este poder de paralelismo pode ser utilizado para realizar o processamento dos algoritmos de erosão de forma eficiente e ainda preservar o nível de realismo necessário para simulações (MEI; DECAUDIN; HU, 2007, p. 48).

Diante do apresentado, este trabalho propõe o desenvolvimento de uma ferramenta para geração procedural de terrenos virtuais com aparência natural utilizando modelos estocásticos e físicos. Para tal, propõe também uma versão otimizada em GPU dos algoritmos de erosão de terreno comumente utilizados pelos modelos físicos, para que seja possível executá-los de forma eficiente durante a geração do terreno, a fim de alcançar uma aparência natural no menor tempo possível.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma ferramenta para geração de terrenos virtuais com aparência natural que sejam adequados para uso em simulações.

Os objetivos específicos são:

a) desenvolver um algoritmo para geração de terrenos com aparência natural;
b) realizar o processamento do algoritmo em GPU;
c) disponibilizar uma interface para visualização do terreno gerado.

2 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos que possuem características semelhantes aos principais objetivos do estudo proposto. O primeiro apresenta um método para sintetizar terrenos fractais de aparência natural utilizando algoritmos procedurais e de erosão (OLSEN, 2004), o segundo é um simulador de dinâmica de relevos que utiliza algoritmos de erosão térmica e hidráulica (DIEGOLI NETO, 2017) e o terceiro é um *framework* para modelagem de terrenos complexos com saliências, arcos e cavernas (PEYTAVIE et al., 2009).

2.1 REALTIME PROCEDURAL TERRAIN GENERATION

Olsen (2004) apresenta em seu trabalho técnicas para geração procedural e erosão de superfícies, que são utilizadas em conjunto para geração de terrenos fractais com uma aparência natural, com o objetivo de serem adequados para utilização em jogos. A implementação foi realizada na linguagem de programação Java e o mapa de altura do terreno é representado utilizando uma matriz bidimensional quadrada de dimensões 512 x 512, onde a altura de cada célula é definida por um número decimal entre 0 e 1 (OLSEN, 2004, p. 1).

O processo apresentado por Olsen (2004) pode ser dividido em duas etapas. A primeira consiste na geração do terreno base através de duas técnicas de geração procedural, sendo elas o *diamond-square* e diagrama de Voronoi, ambos algoritmos para geração de ruídos (OLSEN, 2004, p. 2-4). A segunda etapa consiste em tornar o terreno base recém criado mais natural,

submetendo-o à algoritmos de erosão térmica, que simula os materiais dos pontos mais altos se soltando e deslizando pelas encostas até o fundo; e erosão hidráulica, que simula os materiais do terreno sendo dissolvidos, carregados por água corrente e depositados em outro local (OLSEN, 2004, p. 5-11). A Figura 1 apresenta um terreno gerado a partir da execução dos processos descritos acima.

Figura 1 - Terreno gerado através das técnicas de geração procedural e erosão

Fonte: Olsen (2004, p. 1).

Além disso, são apresentadas análises de performance dos algoritmos de erosão térmica e hidráulica utilizados, as quais são utilizadas para propor uma nova versão otimizada dos mesmos, onde a qualidade da simulação (do ponto de vista físico) é prejudicada em troca de um ganho considerável na velocidade de processamento (OLSEN, 2004, p. 5-11).

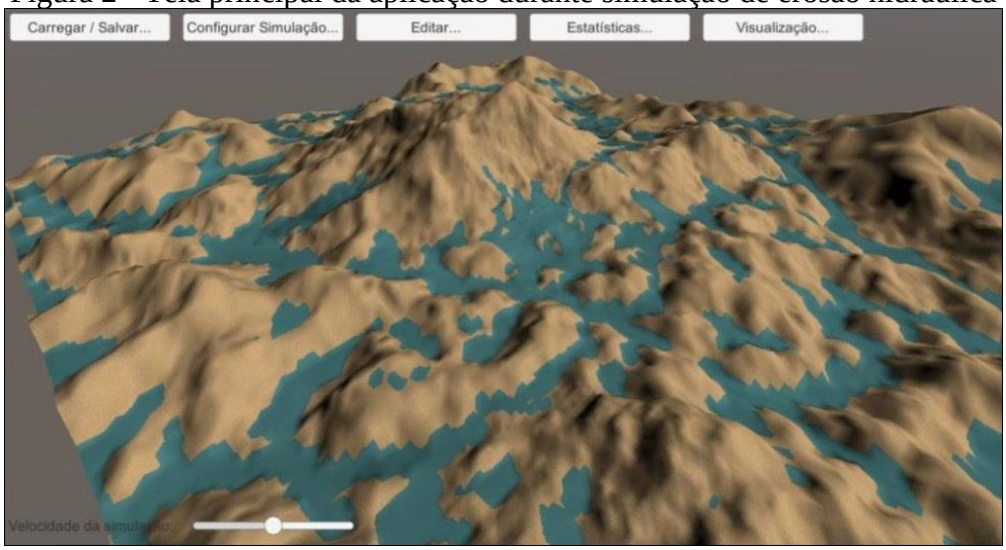
Por fim, Olsen (2004) conclui que as otimizações realizadas no algoritmo de erosão térmica fizeram com que ele se tornasse rápido o suficiente para ser utilizado em um processamento em tempo real, porém resultando em uma redução na qualidade do terreno. Já no caso da versão otimizada do algoritmo de erosão hidráulica, a qualidade do terreno gerado se mostrou maior, porém o tempo de execução continuou sendo muito lento para utilização em tempo real. Ainda assim, os terrenos gerados mostraram-se adequados ao objetivo principal de serem utilizados em jogos (OLSEN, 2004, p. 19).

2.2 SIMULADOR DE DINÂMICA DO RELEVO

Diegoli Neto (2017) apresenta o desenvolvimento de um simulador de deslizamentos de terra. A ferramenta foi desenvolvida utilizando o motor gráfico Unity para visualização da simulação e a linguagem de programação C# para criação dos algoritmos (DIEGOLI NETO, 2017, p. 14). A ferramenta também utiliza algoritmos de erosão térmica e hidráulica para

transformar o relevo e assim como Olsen (2004), utiliza uma matriz bidimensional quadrada para representar o terreno, na qual o valor de cada célula representa a altura do terreno naquela posição. A Figura 2 apresenta a tela principal da aplicação durante a simulação de erosão hidráulica.

Figura 2 - Tela principal da aplicação durante simulação de erosão hidráulica



Fonte: Diegoli Neto (2017, p. 40).

Antes da simulação começar o usuário deve carregar um relevo a partir de um mapa de altura já existente de um terreno (DIEGOLI NETO, 2017, p. 48), podendo escolher diferentes tipos de superfície para diferentes áreas do terreno. As superfícies disponíveis são solo exposto, grama, floresta e concreto, sendo que o tipo da superfície impacta na execução da simulação (DIEGOLI NETO, 2017, p. 42).

Antes de iniciar a simulação é possível selecionar quais algoritmos de erosão serão utilizados, podendo escolher entre erosão térmica e erosão hidráulica (ou ambas). Além disso, é possível configurar algumas variáveis para cada simulação. Para a erosão térmica as variáveis são *fator de alteração* (multiplicador aplicado a todas as alterações) e *inclinação máxima* (maior diferença permitida entre alturas), enquanto que para a erosão hidráulica são *intensidade de chuva*, *intervalo entre chuvas*, *fator de solubilidade* (proporção de solo que será dissolvido pela água) e *fator de evaporação* (DIEGOLI NETO, 2017, p. 49-50).

Uma vez que a simulação foi iniciada, é possível acompanhar visualmente as alterações no terreno em tempo real. Para informações mais detalhadas sobre o estado atual do relevo é possível acessar a tela de estatísticas, onde é fornecido várias informações sobre o solo, umidade, altitude, entre outras (DIEGOLI NETO, 2017, p. 45). Também é possível selecionar uma visão por mapas de calor, podendo escolher entre mapas de profundidade do solo, profundidade da água, umidade do solo e inclinação (DIEGOLI NETO, 2017, p. 46).

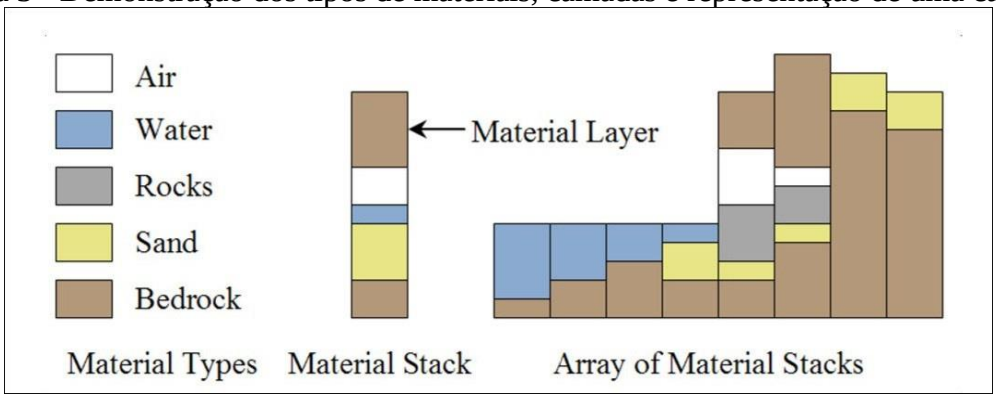
Diegoli Neto (2017) conclui que a simulação desenvolvida apresenta uma visualização condizente com a realidade, mesmo não sendo capaz de oferecer uma representação precisa dos processos realizados. Além disso, relata que mesmo com o motor gráfico Unity apresentando facilidades no desenvolvimento da ferramenta, a plataforma não se mostrou otimizada para trabalhar com atualizações constantes no relevo, o que resultou em uma perda de performance durante as simulações (DIEGOLI NETO, 2017, p. 57-58).

2.3 ARCHES

Enquanto que os trabalhos de Olsen (2004) e Diegoli Neto (2017) apresentados até então se concentram apenas nas características mais gerais do terreno, como montanhas e vales, Peytavie et al. (2009) apresenta um *framework* focado principalmente na modelagem de formações rochosas mais complexas e específicas, como cavernas, arcos, saliências e penhascos.

Para conseguir modelar essas formações rochosas, Peytavie et al. (2009) utiliza uma forma diferente de representar o terreno gerado. Ao invés de utilizar uma matriz bidimensional onde a altura de cada ponto é representada por um valor escalar (como nos dois trabalhos anteriores), mantem-se a matriz, porém utilizando uma estrutura de camadas em cada posição para representar os vários materiais existentes naquele ponto, sendo que cada material é caracterizado por sua espessura e tipo, podendo ser dos tipos: ar, água, areia, rocha e pedra. Dessa forma, torna-se possível representar uma caverna, por exemplo, simplesmente por inserir uma camada de ar entre duas camadas de rocha (PEYTAVIE et al., 2009, p. 3), como mostrado na Figura 3, possibilitando assim a criação de relevos que não seriam possíveis de se representar utilizando apenas um mapa de altura comum.

Figura 3 - Demonstração dos tipos de materiais, camadas e representação de uma caverna



Fonte: Peytavie et al. (2009, p. 3).

Ao invés de trabalhar com geração de terrenos do zero, o *framework* proposto fornece ferramentas de edição de terreno que permitem que o usuário modele as características

principais do terreno através de dois modelos diferentes: um modelo volumétrico discreto, contendo várias camadas de diferentes materiais e um modelo implícito, que representa a superfície do relevo (PEYTAVIE et al., 2009, p. 2).

Após a modelagem inicial, o terreno é submetido a um algoritmo de simulação de estabilização de materiais, no qual materiais granulares como areia, pedra e água interagem uns com os outros e se misturam, a fim de criar um cenário mais próximo do real de forma automática, poupando o usuário deste trabalho (PEYTAVIE et al., 2009, p. 4). Além disso, a etapa de edição do terreno também engloba ferramentas que possuem algoritmos para criar rachaduras ou redes de túneis, realizar a erosão de superfícies e criar pilhas de rochas de forma procedural (PEYTAVIE et al., 2009, p. 5-8).

Peytavie et al. (2009) conclui afirmando ter apresentado uma abordagem original para a representação de estruturas complexas em terrenos, disponibilizando ferramentas que possibilitam a modelagem de arcos, cavernas, penhascos, pilhas de rochas e outras estruturas. Afirma também que a ferramenta se mostrou eficaz, sendo utilizada na construção de uma variedade de cenas com características geológicas únicas (PEYTAVIE et al., 2009, p. 10).

3 PROPOSTA DA FERRAMENTA

Nessa seção será apresentada a justificativa para a realização da pesquisa proposta, assim como os requisitos principais da ferramenta a ser desenvolvida e a metodologia utilizada durante a realização da pesquisa.

3.1 JUSTIFICATIVA

O Quadro 1 abaixo apresenta uma comparação entre os trabalhos correlatos apresentados anteriormente na seção 2.

Quadro 1 - Comparativo entre os trabalhos correlatos

Correlatos / Características	Olsen (2004)	Diegoli Neto (2017)	Peytavie et al. (2009)
tipo de ferramenta	geração	simulação	modelagem
utiliza algoritmos de erosão	Sim	Sim	Sim
utiliza algoritmos de geração procedural	Sim	Não	Sim
permite visualização do terreno	Sim	Sim	Sim
utiliza processamento em GPU	Não	Não	Não

Fonte: elaborado pelo autor.

Conforme é possível observar no Quadro 1, os três trabalhos apresentados possuem objetivos diferentes. Olsen (2004) é o único cuja finalidade é a mesma que o trabalho proposto, ou seja, a geração de terrenos virtuais, enquanto Diegoli Neto (2017) apresenta um

simulador de processos de dinâmica de relevo e Peytavie et al. (2009) apresenta uma ferramenta para modelagem de terrenos.

Apesar disso, todos realizam a modificação do terreno base através de algoritmos de erosão, porém em contextos diferentes. Olsen (2004) os utiliza para melhorar a qualidade do terreno gerado, tornando-o mais natural, Diegoli Neto (2017) os utiliza como componente principal nas simulações de deslizamento e Peytavie et al. (2009) os utiliza como uma ferramenta de edição que pode ser aplicada em uma área específica do terreno.

Os algoritmos de geração procedural, por outro lado, são utilizados apenas por Olsen (2004) e Peytavie et al. (2009), sendo que o primeiro os utiliza para a geração do terreno base, enquanto o segundo não os utiliza diretamente no terreno, mas sim na criação de pilhas de rochas. Além disso, também é possível observar que todos possuem uma forma de visualização dos terrenos dentro da própria ferramenta, sem a necessidade de utilizar ferramentas externas para tal.

Em relação a utilização de GPU como forma de otimização dos algoritmos, nenhum deles possui tal implementação, porém todos apresentam certa preocupação em relação a desempenho. Peytavie et al. (2009) menciona que a etapa de renderização do terreno produzido por sua ferramenta poderia ser facilmente implementada como um *shader* em GPU (PEYTAVIE et al., 2009, p. 2). Diegoli Neto (2017) sugere que a renderização do terreno poderia ser feita através de outra plataforma que não a Unity, a fim de conseguir melhor desempenho (DIEGOLI NETO, 2017, p. 58). E por fim, Olsen (2004) apresenta uma versão otimizada de alguns algoritmos de erosão, mas que não se mostraram totalmente eficazes (OLSEN, 2004, p. 19).

A partir do apresentado conclui-se que todos os correlatos possuem uma certa preocupação com performance, apesar de nenhum deles tentar resolver este problema com a utilização de programação em GPU. Todos apresentam técnicas semelhantes (porém adaptadas as suas necessidades) de algoritmos de erosão e dois deles, de geração procedural.

Dessa forma, o trabalho proposto mostra-se relevante no que diz respeito a apresentar uma forma de otimização em GPU dos algoritmos de geração procedural e erosão já amplamente utilizados, o que possibilitará um tempo de execução melhor sem necessitar de uma perda de qualidade no terreno resultante. A pesquisa realizada também estudará a viabilidade de utilizar uma ferramenta de desenvolvimento de jogos em alto nível como Unity em conjunto com programação em GPU. Além disso, por se tratar de uma área ainda não muito estudada em nossa universidade, este trabalho tem potencial de possibilitar a abertura de novas pesquisas referentes a programação em GPU, como simulação de fluídos e

partículas, que tendem a ser direcionadas a GPU e portanto utilizar os mesmos princípios que serão estudados aqui.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta proposta neste trabalho deverá:

a) permitir a geração de terrenos virtuais com aparência natural (RF - Requisito Funcional);
b) permitir a parametrização de algumas propriedades do terreno gerado (RF);
c) permitir a visualização do terreno em 3D (RF);
d) permitir a navegação pelo terreno através da movimentação da câmera (RF);
e) permitir a exportação do terreno gerado no formato de mapa de altura (RF);
f) permitir a importação de um terreno previamente gerado (RF);
g) permitir a visualização de características do terreno através de mapas de calor como mapas de altura e inclinação (RF);
h) ser desenvolvido para computadores com placa de vídeo, porém sem necessitar de um modelo específico (RNF - Requisito Não Funcional);
i) ser desenvolvido utilizando o motor gráfico Unity para a visualização do terreno e a linguagem C# para implementação dos algoritmos gerais (RNF);
j) utilizar as linguagens ShaderLab e HLSL para otimização dos algoritmos de erosão em GPU (RNF).

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar levantamento bibliográfico sobre formação natural de relevos, técnicas para geração de terrenos virtuais com aparência natural e processamento paralelo em GPU;
b) elicitação de requisitos: detalhar e reavaliar os requisitos de acordo com o levantamento bibliográfico;
c) especificação da solução: formalizar as funcionalidades da ferramenta através dos diagramas da *Unified Modeling Language* (UML), utilizando o programa Draw.io;
d) implementação da ferramenta: implementar a ferramenta proposta utilizando as linguagens C#, HLSL, ShaderLab e o motor gráfico Unity;
e) validação: validação dos resultados alcançados através de testes de performance para avaliar a velocidade dos algoritmos implementados e análise da naturalidade

dos terrenos gerados por um profissional da geologia.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2020									
	ago.		set.		out.		nov.		dez.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação de requisitos										
especificação da solução										
implementação da ferramenta										
validação										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta a fundamentação dos assuntos abordados no trabalho, estando dividido em quatro seções. A seção 4.1 apresenta a relação entre modelos estocásticos e geração procedural, além de técnicas procedurais utilizadas. A seção 4.2 apresenta o conceito de modelos físicos e como podem ser aplicados para geração de terrenos naturais através de algoritmos de erosão. A seção 4.3 aborda o processamento paralelo em GPU (*Graphics Processing Unit*), assim como vantagens e cuidados a serem tomados. Por fim, a seção 4.4 apresenta algumas plataformas para programação em GPU e a utilização de *shaders* criados na linguagem HLSL.

4.1 GERAÇÃO PROCEDURAL E MODELOS ESTOCÁSTICOS

Tradicionalmente, objetos gráficos virtuais são modelados a partir de polígonos . Estes polígonos podem ser descritos matematicamente através de equações determinísticas, o que faz com que suas características sejam previsíveis e facilmente reproduzíveis. Porém, ao tentar modelar objetos naturais como rochas, nuvens e árvores utilizando tais equações, o resultado tende a ser objetos que não se assemelham totalmente com a realidade, justamente por possuírem características irregulares difíceis de serem reproduzidas por este método (FOURNIER; FUSSELL; CARPENTER, 1982, p. 371). Uma alternativa para modelar tais objetos é a utilização de modelos estocásticos.

Enquanto em um modelo determinístico, dado um conjunto de pontos de um objeto, o resultado é previsível, em um modelo estocástico o resultado para o mesmo conjunto seria imprevisível, o que possibilita a criação de um objeto com características pseudoaleatórias. Desta forma a superfície de um objeto pode ser construída através da simples interpolação entre um ponto inicial e final (determinístico), ou pode variar de forma irregular ao longo do

trajeto (estocástico), possibilitando um resultado mais próximo da realidade ao modelar objetos naturais (FOURNIER; FUSSELL; CARPENTER, 1982, p. 372). Estes resultados podem ser alcançados através do uso de técnicas procedurais.

Segundo Ebert et al. (2003, p. 1, tradução nossa) "técnicas procedurais são segmentos de código ou algoritmos que especificam alguma característica de um modelo ou efeito gerado por computador.". Isso permite que características como forma, altura e textura do objeto sejam determinadas automaticamente através da utilização de algoritmos e funções matemáticas, eliminando a necessidade de escanear um objeto real para reproduzi-lo virtualmente ou modelar o objeto virtual por completo (EBERT et al., 2003, p. 1).

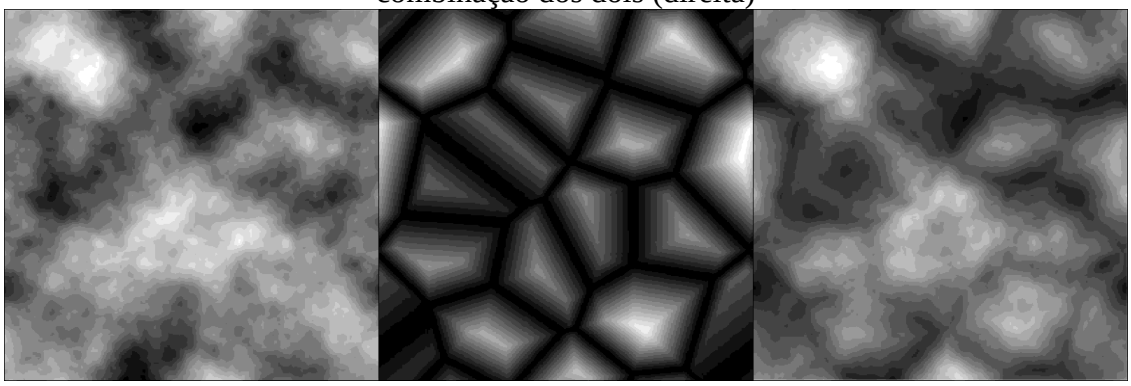
Estas técnicas podem ser utilizadas para criação de vários tipos diferentes de objetos naturais, como por exemplo rochas (PEYTAVIE et al., 2009, p. 7-8), nuvens e estrelas (RODEN; PARBERRY, 2005), árvores (LONGAY et al., 2012), rios (GÉNEVAUX et al., 2013) e até mesmo cenários não naturais como cidades com estradas e prédios (KELLY; MCCABE, 2007). Apesar de cada um destes trabalhos proporem a geração de objetos totalmente diferentes, algo comum entre todos é a utilização de padrões de ruídos como ferramenta principal em seus algoritmos.

Ruídos são funções estocásticas utilizadas para criar irregularidades nos objetos gerados. O tipo mais simples de ruído chama-se *white noise* e consiste na geração de um conjunto de números aleatórios, sem nenhuma relação entre si (EBERT et al., 2003, p. 67). Como os números gerados não possuem relação, este tipo de ruído é raramente utilizado sozinho por algoritmos procedurais. Ao invés disso, é mais comum a utilização de uma versão suavizada do *white noise* que possui transições graduais entre os números gerados, apelidada de *pink noise* (EBERT et al., 2003, p. 68). Além dos dois ruídos mencionados, Ebert et al. (2003) apresenta outros padrões de ruídos que podem ser utilizados, além da possibilidade de combinar diferentes ruídos para criar outros padrões.

Tratando-se da geração de terrenos, Olsen (2004) realiza este processo através da combinação de duas técnicas para geração de ruídos, sendo elas: *midpoint displacement* (que gera *pink noise*) e diagrama de Voronoi. Por mais que os pontos gerados pelo *midpoint displacement* sejam pseudoaleatórios, eles tendem a variar dentro de uma mesma amplitude, resultando em terrenos com características monótonas. A fim de quebrar a monotonia, Olsen (2004, p. 4) propõe a combinação do *midpoint displacement* com o diagrama de Voronoi, que pode ser utilizado para criar regiões similares a montanhas. A Figura 4 apresenta exemplos de ruídos gerados pelo *midpoint displacement*, diagrama de Voronoi e a combinação dos dois,

onde áreas mais claras representam pontos mais elevados do terreno e áreas escuras pontos mais baixos.

Figura 4 - Exemplo de *midpoint displacement* (esquerda) diagrama de Voronoi (meio) e combinação dos dois (direita)



Fonte: adaptado de Olsen (2004, p. 3-4).

4.2 TERRENOS REALISTAS E MODELOS FÍSICOS

À primeira vista terrenos gerados através de técnicas procedurais como as descritas na seção 4.1 parecem convincentemente naturais. Porém, uma análise meticulosa revelaria a ausência de características presentes em terrenos reais. Um exemplo ocorre ao comparar áreas altas e baixas de uma mesma montanha. Em um terreno real, as partes inferiores da montanha estariam cobertas por resíduos provenientes do topo que se desprendem ao longo do tempo, porém isto não ocorre em terrenos gerados de forma procedural, visto que tais técnicas não levam processos físicos em consideração (MUSGRAVE; KOLB; MACE, 1989, p. 41).

Modelos físicos são utilizados para reproduzir fenômenos físicos computacionalmente, a fim de produzir distribuições espaciais e temporais detalhadas do fenômeno em questão (LONG; HE, 2009, p. 1). Isto é feito através da implementação de algoritmos que simulam processos físicos reais ao longo de um determinado tempo. Este trabalho utilizará modelos físicos para geração de terrenos, porém estes modelos também podem ser utilizados em outras áreas gráficas, como simulação de fluídos (MÜLLER; CHARYPAR; GROSS, 2003), nuvens (HARRIS et al., 2003), neve (GOSWAMI; MARKOWICZ; HASSAN, 2019) e fumaça (ISHIDA; ANDO; MORISHIMA, 2019).

No caso de terrenos, modelos físicos são normalmente utilizados para simular processos de erosão, como vazão de água, choques térmicos e ventos, assim como os efeitos causados na superfície do terreno. Os processos de erosão mais utilizados tendem a ser os de erosão térmica e hidráulica, visto que estes causam mudanças mais impactantes no terreno (JÁKÓ; TÓTH, 2011, p. 57).

O algoritmo de erosão hidráulica consiste em simular o processo de chuvas em terrenos, que faz com que a água corrente resultante da chuva dissolva materiais de certas partes do terreno, que são carregados pela correnteza e depositados em outro local (OLSEN, 2004, p. 8). Já a erosão térmica simula o processo de materiais se desprenderem de pontos elevados do terreno e deslizarem até pontos mais baixos (OLSEN, 2004, p. 5). Estes algoritmos foram utilizados ao longo dos anos por trabalhos como Musgrave, Kolb e Mace (1989), Olsen (2004), Mei, Decaudin e Hu (2007), Long e He (2009), Jákó e Tóth (2011) e Diegoli Neto (2017). Sendo que cada um apresenta sua própria implementação dos algoritmos, tendo como objetivo a geração de terrenos naturais ou a simulação dos processos em questão.

4.3 GPU

Com a crescente necessidade de processamento de grandes quantidades de dados, surge a demanda por plataformas capazes de processar massas de dados de forma eficiente. Uma forma de atingir isto é através do paralelismo. Isto fica evidente em áreas como Processamento de Imagem e Visão Computacional, que tendem a aplicar a mesma operação repetidamente em cada *pixel* de uma imagem. Este tipo de operação, por mais que realizada frequentemente de forma sequencial, é possível de ser paralelizada (PARK et al., 2011, p. 1).

Recentemente as GPUs tem se mostrado ferramentas adequadas para este tipo de processamento em paralelo (SAHA et al., 2016, p. 516). Isso se deve principalmente ao fato de sua arquitetura ser desenvolvida pensando exclusivamente em processamento paralelo, diferente das CPUs (*Central Process Unit*) que foram criadas para processamento serial e adaptadas para paralelo (HANGÜN; EYECIOĞLU, 2017, p. 34). A Figura 5 apresentada por Saha et al. (2016) ilustra a diferença arquitetural entre CPU e GPU, onde é possível observar que a GPU possui vários blocos de processamento compostos por ALUs (*Arithmetic Logic Unit*), o que permite o processamento de grandes quantidades de dados simultaneamente.

Figura 5 - Diagrama de blocos ilustrando a diferença arquitetural entre CPU e GPU

Fonte: Saha et al. (2016, p. 517).

Diferentes áreas da computação podem se beneficiar desta arquitetura para otimizar o processamento de seus algoritmos. O ganho de performance é visível em trabalhos como os de Hangün e Eyecioğlu (2017) e Saha et al. (2016), que implementam diversos algoritmos de filtros de imagens tanto em CPU como em GPU a fim de comparar as diferenças de performance entre os dois, observando melhorias de até 120% na performance de alguns dos algoritmos em GPU. Che et al. (2007) propõe o mesmo estilo de análise, porém com algoritmos de propósitos gerais como simulação térmica e *K-Means* (utilizado em mineração de dados). Em todos os casos observa-se um ganho considerável de performance nas versões em GPU. Esta diferença de performance fica mais perceptível conforme o tamanho do dado a ser processado aumenta (HANGÜN; EYECIOĞLU, 2017, p. 39-40).

Também se destacam os casos em que a versão em GPU do algoritmo não apresenta ganhos consideráveis se comparada com a versão em CPU. Isso ocorre principalmente quando existem muitas operações sendo executadas em *threads* diferentes porém dependentes entre si, o que diminui a eficácia do paralelismo (CHE et al., 2007, p. 8); ou em casos em que são necessários constantes acessos à memória e o processo não foi corretamente configurado para isso, fazendo mau uso da memória compartilhada da GPU (PARK et al., 2011, p. 3).

4.4 DIRECTX E HLSL

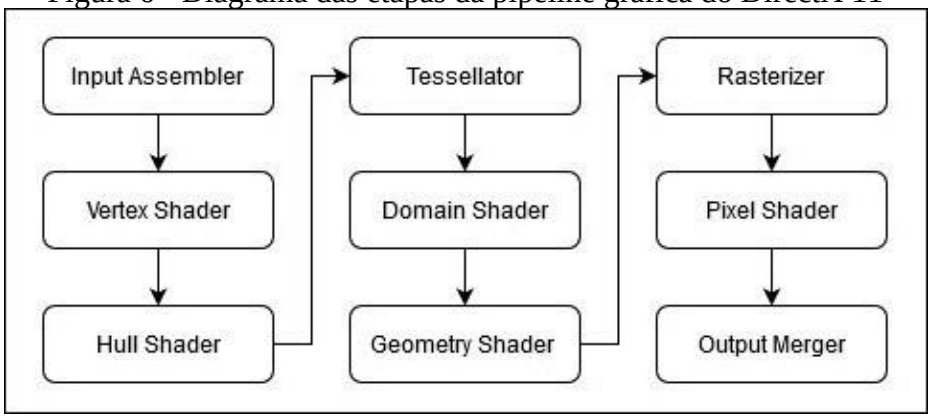
Park et al. (2011, p. 2) destaca a existência de estratégias diferentes de programação em GPU dependendo da plataforma adotada. Isto significa que um algoritmo implementado em uma plataforma não é facilmente adaptável para outra, o que torna a escolha de uma plataforma algo importante para o desenvolvimento. Exemplos de plataformas seriam: CUDA (*Compute Unified Device Architecture*) da Nvidia, DirectX da Microsoft e Metal da Apple. Por razões de maior compatibilidade com o motor gráfico Unity, este trabalho será focado no uso do DirectX 11 com a linguagem HLSL (*High Level Shading Language*).

Como o próprio nome sugere, HLSL é uma linguagem para criação de *shaders*. Um *shader* pode ser definido como um *script* responsável por comunicar à GPU quais cálculos realizar para produzir uma transformação ou efeito desejado (VALDETARO et al., 2010, p. 1). Algo importante para se ter em mente é a existência de diferentes tipos de *shaders*, que são responsáveis por diferentes etapas da *pipeline* gráfica. A Figura 6 apresenta um diagrama com todas as oito etapas existentes na *pipeline* gráfica do DirectX 11, sendo que das oito, cinco são compostas por *shaders* programáveis através da linguagem HLSL (MICROSOFT, 2018).

Por se tratar de um conteúdo extenso, este trabalho não entrará em detalhes sobre cada etapa da *pipeline*, porém este conhecimento se faz necessário para melhor uso da *pipeline*.

Explicações detalhadas sobre todas as etapas são apresentadas em Microsoft (2018), assim como um diagrama detalhado da *pipeline*. Adicionalmente, Valdetaro et al. (2010) apresenta informações e exemplos práticos das etapas *Hull Shader*, *Tessellator* e *Domain Shader*, que estão disponíveis apenas a partir do DirectX 11.

Figura 6 - Diagrama das etapas da pipeline gráfica do DirectX 11



Fonte: elaborado pelo autor.

REFERÊNCIAS

CHE, Shuai et al. **A Performance Study of General Purpose Applications on Graphics Processors**. 2007. 10 f. Department of Computer Science, University of Virginia, Charlottesville.

DIEGOLI NETO, Guilherme. **Simulação de dinâmica do relevo através da transformação de mapas de altura**. 2017. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

EBERT, David. S. et al. **Texturing and Modeling**: A Procedural Approach. 3. ed. São Francisco: Morgan Kaufmann, 2003.

FOURNIER, Alain; FUSSELL, Don; CARPENTER, Loren. Computer rendering of stochastic models. **Communications of the ACM**, New York, v. 25, n. 6, p. 371-384, jun. 1982.

GÉNEVAUX, Jean-David et al. Terrain generation using procedural models based on hydrology. **ACM Transactions on Graphics**, New York, v. 32, n. 4, p. 1-13, jul. 2013.

GOSWAMI, Prashant; MARKOWICZ, Christian; HASSAN, Ali. Real-time particle-based snow simulation on the GPU. In: EUROGRAPHICS SYMPOSIUM ON PARALLEL GRAPHICS AND VISUALIZATION, 2019, Porto. **Proceedings…** Goslar: Eurographics Association, 2019. p. 49-57.

HANGÜN, Batuhan; EYECIOĞLU, Önder. Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations. **International Journal of Engineering Science and Application**, Istanbul, v. 1, n. 2, p. 35-41, jul. 2017.

HARRIS, Mark J. et al. Simulation of cloud dynamics on graphics hardware. In: ACM SIGGRAPH/EUROGRAPHICS CONFERENCE ON GRAPHICS HARDWARE, 3., 2003, San Diego. **Proceedings…** Goslar: Eurographics Association, 2003. p. 92-101.

HEERMANN, Dieter W. **Computer Simulation Methods in Theoretical Physics**. 2. ed. Heidelberg: Springer, 1990.

ISHIDA, Daichi; ANDO, Ryoichi; MORISHIMA, Shigeo. GPU Smoke Simulation on Compressed DCT Space. In: EUROGRAPHICS, 40., 2019, Genoa. **Proceedings…** Goslar: Eurographics Association, 2019. p. 5-8.

JÁKÓ, Balázs; TÓTH, Balázs. Fast Hydraulic and Thermal Erosion on the GPU. In: 15TH CENTRAL EUROPEAN SEMINAR ON COMPUTER GRAPHICS, 15., 2011, Vinicné. **Proceedings…** Viena: TU Wien, 2011. p. 139-146.

KELLY, George; MCCABE, Hugh. **Citygen**: An Interactive System for Procedural City Generation. 2007. 9 f. Department of Informatics, Institute of Technology Blanchardstown, Dublin.

LONG, Mansheng; HE, Dongjian. Hydraulic erosion simulation using finite volume method on graphics processing unit. In: INTERNATIONAL CONFERENCE ON INFORMATION ENGINEERING AND COMPUTER SCIENCE, 2009, [Wuhan]. **Proceedings...** Piscataway: IEEE, 2009. p. 1-4.

LONGAY, Steven et al. Treesketch: interactive procedural modeling of trees on a tablet. In: SYMPOSIUM ON SKETCH-BASED INTERFACES AND MODELING, 5., 2012, Annecy. **Proceedings...** Goslar: Eurographics Association, 2012. p. 107-120.

MEI, Xing; DECAUDIN, Philippe; HU, Bao-Gang. Fast hydraulic erosion simulation and visualization on GPU. In: PACIFIC CONFERENCE ON COMPUTER GRAPHICS AND APPLICATIONS, 15., 2007, Maui. **Proceedings…** Piscataway: IEEE, 2007. p. 47-56.

MICROSOFT. **Graphics Pipeline**. [S.l.], 2018. Disponível em: <https://docs.microsoft.com/en-us/windows/win32/direct3d11/overviews-direct3d-11-graphics-pipeline>. Acesso em: 1 de junho de 2020.

MÜLLER, Matthias; CHARYPAR, David; GROSS, Markus. Particle-based fluid simulation for interactive applications. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION, 3., 2003, San Diego. **Proceedings…** Goslar: Eurographics Association, 2003. p. 154-159.

MUSGRAVE, F. Kenton; KOLB, Craig E.; MACE, Robert S. The synthesis and rendering of eroded fractal terrains. **ACM Siggraph Computer Graphics**, [S.l.], v. 23, n. 3, p. 41-50, jul. 1989.

OLSEN, Jacob. **Realtime Procedural Terrain Generation**: Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games. 2004. 20 f. Department of Mathematics and Computer Science, University of Southern Denmark, Odense.

PARK, In Kyu et al. Design and Performance Evaluation of Image Processing Algorithms on GPUs. **IEEE Transactions on Parallel and Distributed Systems**, Piscataway, v. 22, n. 1, p. 91-104, jan. 2011.

PEYTAVIE, Adrien et al. Arches: a framework for modeling complex terrains. **Eurographics**, Blackwell, v. 28, n. 2, p. 457-467, abr. 2009.

PLIENINGER, Tobias et al. Exploring ecosystem-change and society through a landscape lens: recent progress in European landscape research. **Ecology and Society**, [S.l.], v. 20, n. 2, art. 5, jun. 2015.

RODEN, Timothy; PARBERRY, Ian. Clouds and stars: efficient real-time procedural sky rendering using 3d hardware. In: ACM SIGCHI INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTER ENTERTAINMENT TECHNOLOGY, 5., 2005, Valencia. **Proceedings...** New York: Association for Computing Machinery, 2005. p. 434-437.

SAHA, Diptarup et al. Implementation of Image Enhancement Algorithms and Recursive Ray Tracing using CUDA. **Procedia Computer Science**, Netherlands, v. 79, p. 516-524, 2016.

VALDETARO, Alexandre et al. Understanding Shader Model 5.0 with DirectX 11. In: SIMPÓSIO BRASILEIRO DE JOGOS E ENTRETENIMENTO DIGITAL, 9., 2010, Florianópolis. **Anais…** Florianópolis: [s.n.], 2010. p. 1-18.

ASSINATURAS

(Atenção: todas as folhas devem estar rubricadas)

Assinatura do(a) Aluno(a): _____

Assinatura do(a) Orientador(a): _____

Assinatura do(a) Coorientador(a) (se houver): _____

Observações do orientador em relação a itens não atendidos do pré-projeto (se houver):

FORMULÁRIO DE AVALIAÇÃO (PROJETO) – PROFESSOR TCC I

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. REVISÃO BIBLIOGRÁFICA Os assuntos apresentados são suficientes e têm relação com o tema do TCC?			
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			
	9. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?			
	10. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?			
	11. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?			
	As citações obedecem às normas da ABNT?			
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?			

PARECER – PROFESSOR DE TCC I OU COORDENADOR DE TCC:

O projeto de TCC será reprovado se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADO ( ) REPROVADO

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

FORMULÁRIO DE AVALIAÇÃO (PROJETO) – PROFESSOR AVALIADOR

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. REVISÃO BIBLIOGRÁFICA Os assuntos apresentados são suficientes e têm relação com o tema do TCC?			
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			

PARECER – PROFESSOR AVALIADOR:

O projeto de TCC será reprovado, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADO ( ) REPROVADO

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.