UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

OTÁVIO AUGUSTO DELLAY SECCO

CLASSIFICAÇÃO PARA CONTEÚDO GERADO DE FORMA PROCEDURAL

UTILIZANDO FRACTIONAL BROWNIAN MOTION

CHAPECÓ
2021

OTÁVIO AUGUSTO DELLAY SECCO

CLASSIFICAÇÃO PARA CONTEÚDO GERADO DE FORMA PROCEDURAL
UTILIZANDO FRACTIONAL BROWNIAN MOTION

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dr. Fernando Bevilacqua

CHAPECÓ
2021

Secco, Otávio Augusto Dellay

Classificação para conteúdo gerado de forma Procedural : Utilizando Fractional Brownian Motion / Otávio Augusto Dellay Secco. – 2021.

46 f.: il.

Orientador: Prof. Dr. Fernando Bevilacqua.

Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2021.

1. Geração Procedural. 2. *Fractional Brownian Motion*. 3. Classificação. I. Bevilacqua, Prof. Dr. Fernando, orientador. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: otavio.delaiseco@gmail.com

OTÁVIO AUGUSTO DELLAY SECCO

CLASSIFICAÇÃO PARA CONTEÚDO GERADO DE FORMA PROCEDURAL
UTILIZANDO FRACTIONAL BROWNIAN MOTION

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Fernando Bevilacqua

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 27/11/2021.

BANCA AVALIADORA

_____
Prof. Dr. Fernando Bevilacqua - UFFS

_____
Prof. Dr. Emilio Wuerges - UFFS

_____
Prof. Dr. Guilherme Dal Bianco - UFFS

RESUMO

Os videogames estão ganhando cada vez mais popularidade nos dias atuais, com isso, nasce uma demanda por conteúdos cada vez mais complexos, detalhados e completos. Takatsuki (20) e Iosup (9) comentam em suas obras que essa produção não é escalável. A Geração Procedural visa facilitar a criação de ambientes detalhados reduzindo-se ou eliminando o trabalho manual. Uma das técnicas que existe dentro da Geração Procedural é o Fractional Brownian motion (FBm). Nesse trabalho foi testado se uma implementação do FBm é capaz de recriar duas cenas do jogo Valheim, uma simples e outra complexa, respeitando métricas já existentes na literatura. Como resultado, a técnica se mostrou extremamente eficiente em construir suas próprias paisagens, contudo, ao tentar recriar cenas que já são geradas proceduralmente, suas limitações ficaram aparentes.

Palavras-chave: Geração Procedural. *Fractional Brownian Motion*. Classificação.

ABSTRACT

Video games are gaining more and more popularity nowadays, and with it, grows the demand for increasingly complex content, detailed and complete come along. Takatsuki (20) and Iosup (9) comment in their works that this production is not scalable. Procedural Generation aims to facilitate the creation of detailed environments by reducing or eliminating manual labor. One of the techniques that exists within Procedural Generation is the Fractional Brownian motion (FBm). In this work, it has been tested whether an FBm implementation is capable of generating two scenes of the game Valheim, one simple and other complex, respecting metrics already existing in the literature. As result, the technic was extremely eficient in building its own landscapes, however, by trying to recreate proceduraly generated scenes, its limitations became apparent.

Keywords: Procedural Generation. Fractional Brownian Motion. Classification.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de soma de ruídos utilizando o FBm, fonte: Randi Rost (17) . . . 15
Figura 2 – Mundo bidimensional criado de forma procedural do jogo Terraria, fonte: chriskronen (2) . . . 16
Figura 3 – Rede de Estradas Gerada de Forma Procedural, fonte: Dgreen (3) . . . 24
Figura 4 – Mundo gerado proceduralmente no jogo Valheim (12). . . . 25
Figura 5 – Masmorra Gerada de Forma Procedural. Reproduzido de Unzaga (23) . . . 25
Figura 6 – Cidade Gerada de Forma Procedural, fonte Greuter et al. (5) . . . 26
Figura 7 – Relação Entre Frequência e Amplitude . . . 27
Figura 8 – Imagem Mostrando o FBm usando 3 lacunaridades Diferentes . . . 28
Figura 9 – Contexto simples que foi replicado pelo algoritmo. . . . 31
Figura 10 – Contexto complexo que foi replicado pelo algoritmo. . . . 31
Figura 11 – Noise Map . . . 32
Figura 12 – Colour Map . . . 33
Figura 13 – Mapa 3D . . . 33
Figura 14 – Mapa 3D com Árvores . . . 33
Figura 15 – Adicionando Grama à imagem anterior . . . 34
Figura 16 – Adicionando Pedras à imagem anterior . . . 34
Figura 17 – Implementação dá água . . . 34
Figura 18 – GUI . . . 35
Figura 19 – Cena Simples Gerada de Forma Procedural . . . 37
Figura 20 – Tempos de Geração da Cena Simples . . . 37
Figura 21 – Comparação entre as cenas simples . . . 38
Figura 22 – Cena Complexa Gerada de Forma Procedural . . . 38
Figura 23 – Tempos de Geração da Cena Simples . . . 39
Figura 24 – Comparação entre as cenas complexas . . . 39

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO 15
1.1 APRESENTAÇÃO 15
1.2 PROBLEMÁTICA 16
2 OBJETIVOS 19
2.1 OBJETIVOS GERAIS 19
2.2 OBJETIVOS ESPECÍFICOS 19
3 JUSTIFICATIVA 21
4 REVISÃO BIBLIOGRÁFICA 23
4.1 TRABALHOS RELACIONADOS 23
4.1.1 Panorama do campo de geração procedural de conteúdo 23
4.1.2 Importância de algoritmos de ruído para geração procedural 26
4.1.3 Geração procedural baseada em buscas 28
5 METODOLOGIA 31
5.1 EXPERIMENTO 31
5.2 IMPLEMENTAÇÃO 32
6 RESULTADOS 37
6.1 GERAÇÃO DA CENA SIMPLES 37
6.2 GERAÇÃO DA CENA COMPLEXA 38
6.3 ESPECIFICAÇÕES DA MÁQUINA UTILIZADA 39
7 DISCUSSÃO 41
7.1 ANÁLISE DOS RESULTADOS 41
7.2 SOBREPOSIÇÃO DE OBJETOS 42
7.3 DETALHE DA GRAMA 42
8 CONCLUSÃO 43
8.1 TRABALHOS FUTUROS 43
REFERÊNCIAS 45

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

Os videogames, ou jogos digitais, estão ganhando cada vez mais popularidade nos dias atuais. Essas mídias abrangem um público diversificado como adultos, crianças, adolescentes, jovens, homens e mulheres. Diante dessa popularidade, nasce uma exigência de produtos cada vez mais completos, complexos e detalhados.

Para auxiliar na produção de conteúdo, foram desenvolvidas diversas técnicas, dentre elas a geração procedural. Essa permite a geração de conteúdos pseudo-aleatório, isso é, um comportamento aleatório com os valores relacionados. Técnicas desse tipo reduzem o tempo de criação comparado ao que fosse feito manualmente. Por conta disso, a maioria dos títulos que são lançados hoje em dia possuem alguma forma de geração procedural.

Uma geração procedural é uma série de algoritmos que, quando utilizados, são capazes de criar uma infinidade de conteúdos como mapas, cidades, rios, personagens. Também ganha notoriedade o fato de serem mais rápidos e mais eficientes em algumas tarefas do que aquelas desempenhadas por humanos.

Nesse contexto, uma técnica de geração procedural existente é o Fractional Brownian Motion (FBm). Ela consiste em somar ruídos que são também conhecidos como oitavas, de forma regular, fazendo com que a frequência e amplitude mantenham uma granularidade mais fina e detalhes mais precisos.

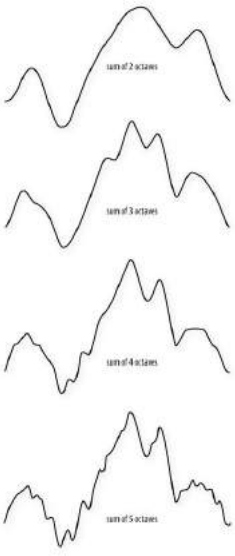

Figura 1 – Exemplo de soma de ruídos utilizando o FBm, fonte: Randi Rost (17)

A Figura 1 mostra a variação das ondas conforme a adição de ruídos, utilizando o FBm. Ao observá-la, é possível perceber que o comportamento do ruído resultante é muito semelhante à linha de relevo de um terreno.

Técnicas similares são utilizadas em títulos comerciais. Os jogos Minecraft e Terraria, por exemplo, geram seus mapas de forma totalmente procedural, utilizando um algoritmo de ruído chamado Perlin Noise. Outro jogo mais recente que vem se destacando no mercado é o Valheim que também possui o seu mapa gerado de forma pseudo-aleatória. A Figura 2 ilustra esse processo. Ao se comparar com a primeira imagem, é possível reconhecer a semelhança entre os dois relevos e como o terreno se molda juntamente com poços d'água, cavernas e minérios no subsolo que vão sendo posicionados.

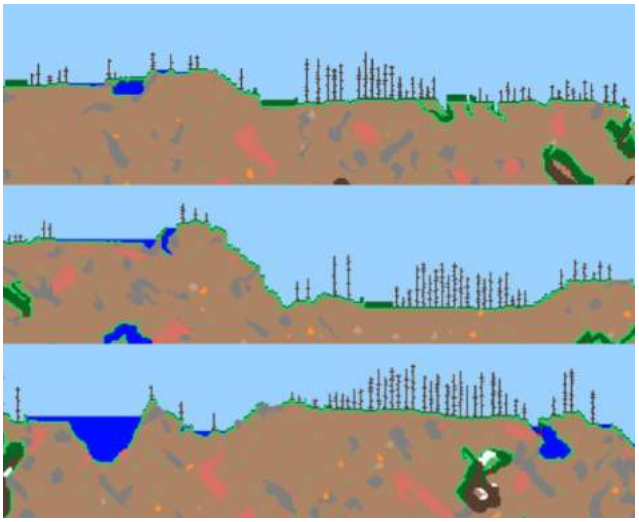

Figura 2 – Mundo bidimensional criado de forma procedural do jogo Terraria, fonte: chriskronen (2)

A geração procedural não se limita apenas a criação de mundos. Hendrikx et al. (8) comenta que jogos como Gears of War 2 e The Elders Scorlls IV: Oblivion geram a vegetação de forma procedural. O jogo da Valve, Left4Dead, gera os encontros com os inimigos de forma dinâmica baseado no stress do jogador medido pelo próprio computador. Diablo II, da Blizzard, gera masmorras (dungeons) de forma procedural.

1.2 PROBLEMÁTICA

A demanda por produtos de maior qualidade é esperada, visto que cada vez mais consumidores exigem um produto com gráficos, história e jogabilidade melhorada se comparada com os títulos anteriores. Isso pode levar a algumas consequências, como o tempo de criação do jogo ser aumentado, o que pode causar adiamento do lançamento do título ou contratação de uma equipe maior. Ambos resultam, invariavelmente, em mais custos financeiros.

Esse é um balanço complexo. O título Red Dead Redemption 2, que para muitos é considerado o melhor jogo já feito, levou 8 anos para ser desenvolvido. Analistas estimam que o orçamento, incluindo produção e divulgação, ficou entre os 370 e 540 milhões de dólares. O tempo sempre foi um inimigo das empresas, e conforme citado anteriormente, quanto mais ele

passa, mais gastos a empresa tem. Por conta disso, é preciso encontrar formas de automatizar algum tipo de serviço ou buscar caminhos para cortar esses gastos.

Existem tarefas que os computadores são melhores na execução do que humanos. Gerar números de forma aleatória, por exemplo, não é a especialidade das pessoas. Apesar de ser simples, posicionar flores por todo um mapa de proporções gigantes pode ser trabalhoso e inviável. Com um algoritmo utilizando métodos que geram esses números dentro de um contexto, por exemplo, utilizando pares de valores como coordenada x y, seria possível posicionar essas flores de forma praticamente instantânea.

Isso é apenas um detalhe de uma infinidade de outras questões que devem ser feitas dentro da criação de um jogo. Por conta disso, o trabalho manual, quando se tratando desses jogos de grande escala, acaba se tornando inviável pois possuirá um custo de tempo muito alto.

Em luz a esse contexto, o presente trabalho apresenta um estudo do algoritmo de geração procedural Fractional Brownian Motion (FBm) no contexto de jogos eletrônicos. Mostra-se as potencialidades e limitações desse através da recriação procedural de uma cena do jogo comercial Valheim.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

Analisar, sob a ótica de métricas de eficiência e qualidade de conteúdo, a capacidade do algoritmo de geração procedural Fractional Brownian Motion em replicar uma cena de um jogo digital.

2.2 OBJETIVOS ESPECÍFICOS

- Revisar na literatura os conceitos de Geração procedural;
- Avaliar os aspectos de geração online/offline do algoritmo FBM;
- Avaliar os aspectos da geração de Conteúdo Adicional x Conteúdo Necessário;
- Avaliar os aspectos da geração Construtiva x Gera e Testa;
- Detalhar e discutir as capacidades e limitações de geração de conteúdo de forma iterativa do algoritmo FBM

3 JUSTIFICATIVA

A Geração Procedural visa facilitar a criação de ambientes detalhados reduzindo-se ou eliminando o trabalho manual. Ela permite a obtenção de um produto de alta qualidade de uma forma barata e relativamente rápida utilizando de ferramentas matemáticas como permutações e interpolações. Além desses, técnicas de customização podem ser utilizadas, como vetores de parâmetros, sementes aleatórias, agentes inteligentes entre diversas outras técnicas.

Nesse contexto, um modelo de algoritmo que ganha muito destaque são os algoritmos de ruídos. Diferentemente da geração puramente aleatória, ele permite que sejam gerados números de forma pseudo-aleatória e interpolados, ou seja, os números não possuirão um padrão mas estarão diretamente relacionados. Essa característica permite a criação de comportamentos, terrenos, imagens, sons entre outras infinidades de opções.

O mercado de jogos possui um preço relacionado com a demanda e fornecimento. Caso o preço seja muito alto, as pessoas não comprarão o produto. Com isso, as empresas enfrentam o desafio de entregar um produto com uma qualidade maior, mas vendendo a preços acessíveis ao público. Com isso, o título precisa vender mais para recuperar o dinheiro investido. Uma forma de poupar o trabalho manual e, consequentemente, o tempo de produção de um jogo que implica no custo final, é utilizando a geração procedural. Dessa forma, é importante que tais algoritmos sejam eficientes para os cenários que forem encontrados.

O entendimento das limitações e potencialidades de algoritmos de geração procedural é muito importante. Por ser uma técnica relativamente simples de implementar, existe uma gama considerável de algoritmos e variações desses. Sendo assim, o processo de escolha do algoritmo não é trivial, visto que é necessário que diversos elementos sejam considerados na forma de classificar ou avaliar os algoritmos existentes.

Nesse contexto, o algoritmo de geração procedural Fractional Brownian Motion mostra-se com características e qualidades para o contexto de desenvolvimento de jogos. Assim como outras técnicas, as diferenças entre os algoritmos podem parecer sutis, por exemplo a diferença de segundos no tempo de execução. Na mesma linha, há diferenças em um resultado mais "suave", ou seja, os números gerados pelas funções possuírem uma variação menor em comparação com o outro algoritmo em questão. Por conta desses detalhes, é necessário um cuidado na escolha desses algoritmos. Durante o planejamento de um jogo, por exemplo, pode-se decidir que o terreno precise ser gerado de forma online, isso é, com um tempo baixo para que o usuário não tenha de esperar em uma tela de carregamento. Logo, deve-se escolher um algoritmo que possua um resultado mais rápido. Por outro lado, supondo que exista a necessidade de adicionar parâmetros durante a geração, deve-se, então, optar por um algoritmo que permita uma personalização nos dados de entrada.

Análises de funcionamento e características de algoritmos de geração procedural, como o Fractional Brownian Motion, foco do presente trabalho, são importantes. Elas apresentam as potencialidades e limitações do algoritmo, contextualizados em um projeto. No caso desse

trabalho, o contexto é o desenvolvimento de jogos digitais.

4 REVISÃO BIBLIOGRÁFICA

4.1 TRABALHOS RELACIONADOS

4.1.1 Panorama do campo de geração procedural de conteúdo

Hendrikx et al. (8) apresenta uma visão sobre todos os dados que podem ser gerados de forma procedural. São citados métodos para a geração procedural de conteúdo como Geradores de Números Pseudo-Aleatórios (PRNG), Gramáticas Gerativas (GG), Filtragem de Imagem (IF), Algoritmos Espaciais (SA), Modelagem e Simulação de Sistemas Complexos (CS) e Inteligência Artificial (IA).

Para sustentar a sua tese, Hendrikx et al. (8) aponta alguns estudos que mostram o custo de produção de conteúdo. Por exemplo, Takatsuki (20) em uma entrevista para a BBC News em 2007 menciona a produção de conteúdo de forma manual como sendo muito cara. Similarmente, Iosup (9) menciona que essa produção não é escalável. O autor também separa a geração procedural em tópicos que contém Game Bits, Espaços do Jogo, Sistemas de Jogo, Cenários de Jogo e Design do Jogo.

Os Game bits, segundo o autor, são as unidades elementares do conteúdo jogo, que normalmente, quando abordadas de forma separada, não envolvem o usuário. O autor também identifica os principais game bits: sons, texturas, vegetação, construções, comportamento (Behaviour) e um outro tópico onde aborda-se Fogo, Água, Pedras e Nuvens.

Ao definir o Espaço do Jogo (Game Space), comenta-se que é onde o ambiente toma o seu lugar, é onde há o preenchimento com os Game Bits nos quais os jogadores podem navegar. São considerados espaço as áreas mono, bi, ou tridimensionais onde os objetos residentes tem posição e direção relativa. Essa classificação contempla elementos como Mapas Internos, Externos e os Corpos D'Água. Os Mapas Internos podem ser vistos de uma forma geral como uma caverna ou um castelo, estando conectada ao mundo exterior por uma porta de entrada por exemplo. Os Mapas Externos, por sua vez, segundo o autor, são representações da elevação e da estrutura de um terreno externo. É comum que mapas com terrenos externos tenham mapas internos, e a sua transição é frequentemente feita de forma discreta. Por fim, os Corpos D'Água são rios, lagos e mares, normalmente são obstáculos que o jogador não consegue passar.

Flake (4) menciona que ecossistemas podem ser modelados como um sistema produtor ou consumidor, através de uma relação que pode ser competitiva ou cooperativa. Essas interações podem fazer surtir comportamentos abrangentes entre os membros de um espécie, sendo que quando há uma variedade de espécies, técnicas CS e PRNG podem ser utilizadas para representar essas variações.

As Redes de Estradas formam a estrutura básica de um mapa externo, servindo para diferentes propósitos como transporte entre pontos de interesse e estruturação e transporte entre cidades. A 3 mostra uma rede de estradas gerada de forma procedural. A maior dificuldade,

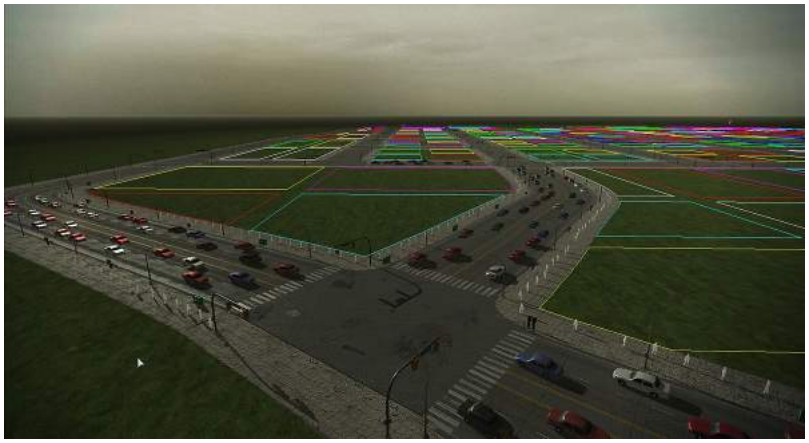

Figura 3 – Rede de Estradas Gerada de Forma Procedural, fonte: Dgreen (3)

segundo o autor, é encontrar o equilíbrio correto entre aleatoriedade e estrutura.

Os Ambientes Urbanos podem ser definidos como grandes aglomerados de construções onde várias pessoas vivem juntas e interagem com os seus arredores. Como no mundo real, as cidades levam séculos para crescer e evoluem por influência das pessoas que moram lá. Algoritmos procedurais frequentemente usam abordagens diferentes para a criação de conteúdo. Por exemplo, primeiro gerar redes de estradas, então dividir o terreno entre estradas e loteamentos de construções e, por último, gerando as cidades dentro desses loteamentos.

O Comportamento das Entidades envolve a necessidade de fazer com que a experiência do jogador seja a mais próxima possível da realidade. Por conta disso, entidades como NPCs (Non-Playable Characters) que interagem com o jogador são uma ferramenta para atingir essa necessidade. Gerar proceduralmente comportamento de entidades baseadas nas ações de jogadores e interações têm o potencial de criar experiências imersivas e realistas.

A História de um jogo também é frequentemente mencionada como a chave para criar uma boa experiência. Ela mantém o jogador motivado, apresenta uma base lógica para os eventos que acontecem ao desenrolar do jogo, e dá um objetivo para o jogador completar. Além disso, ela deve ser inteiramente embasada no universo do jogo para dar uma sensação de imersão. A história, geralmente, está ligada a atividades de game design. Essas são compreendidas como um conteúdo de regras, que é basicamente o que pode ser feito no jogo. Também contemplam os objetivos, que são o que o jogador está tentando alcançar. Um componente estético como arco dramático ou tema gráfico são também elementos importantes no design Norman (14). Um jogo pode ser visto como uma instância de um design, no qual os parâmetros das regras e os objetivos ainda não foram definidos.

O design de sistema de um jogo implica na criação de padrões matemáticos subjacentes ao jogo e suas regras Brathwaite; Schreiber (1). Um exemplo para gerar regras de jogo é o gerador de Pell (16) que gera jogos simétricos semelhantes a xadrez.

O design do mundo de um jogo engloba seu cenário, história e tema (1). Um exemplo inclui a geração de novos jogos baseados em estruturas de história desconhecidas (7). Nessa linha, a Figura 4 mostra um mundo gerado de forma procedural do jogo Valheim.

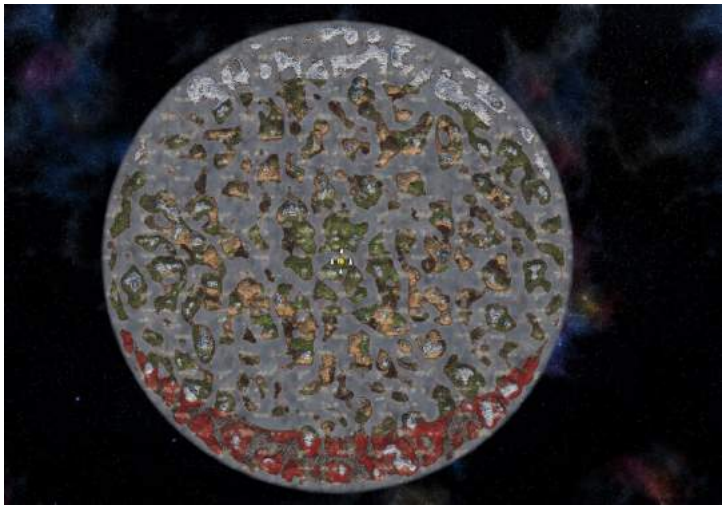

Figura 4 – Mundo gerado proceduralmente no jogo Valheim (12).

A geração de conteúdo pode aumentar o sentimento de imersão que o jogador tem com o mundo do jogo. Um jogo pode mostrar aos jogadores novidades com base em suas ações e outras mudanças no universo do jogo. Brathwaite; Schreiber (1) comenta quais métodos e quais algoritmos são mais interessantes para utilização durante a geração procedural. Para texturas, é mais comum que seja utilizado Perlin Noise.

A geração procedural de construções tem o objetivo de gerar o máximo de construções diferentes. Por exemplo, os sistemas de Lindenmayer, usado pela CityEngine de Parish; Müller (15), inicia de um símbolo representando os limites de uma caixa para a construção. Em seguida, interativamente, transforma a atual construção ou gera novas construções. Essas construções pertencem a um mesmo estilo de arquitetura.

A geração procedural de uma variedade de elementos complexos, em particular água, fogo, nuvens e pedras pode ser gerada a partir do Perlin noise e outras técnicas. Um desses elementos, por exemplo, são masmorras, conforme ilustrado na Figura 5. Dentro do Espaço do Jogo, nos Mapas Internos, as masmorras (dungeons) geradas proceduralmente foram introduzidas no mundo dos videogames em 1992. Dentro delas, existem labirintos, salas, quebra-cabeças foram introduzidos usando uma variedade de algoritmos.

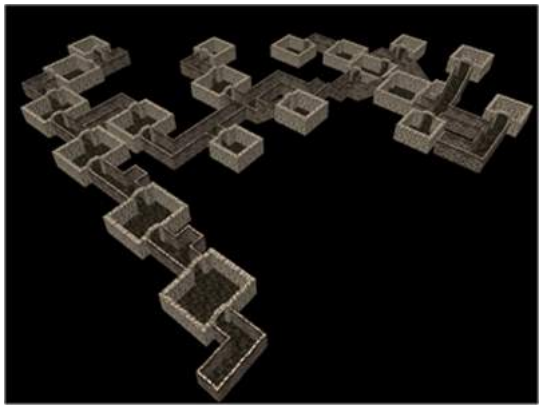

Figura 5 – Masmorra Gerada de Forma Procedural. Reproduzido de Unzaga (23)

Em termos de Mapas Externos, muitos são feitos em estruturas baseadas em redes (grids) no qual cada célula representa uma altura diferente do terreno. Muitos são baseados em técnicas de PRNG, IF, SA e CS, majoritariamente baseadas em técnicas escritas por Musgrave et al. (13) e Smelik et al. (19). Diferentes abordagem foram sugeridas para dar ao designer controle sobre o ambiente produzido.

Usando técnicas CS, um Ecossistema pode ser modelado, segundo Flake (4), como um sistema produtor e consumidor, no qual vários produtores e consumidores interagem de forma competitiva e de forma cooperativa. Comportamentos complexos podem ser modelados para representar a interação entre membros da mesma espécie. Quando múltiplas espécies existem, as interações das inter-espécies obedecem a regras muito mais simples. Portanto, para múltiplas espécies, técnicas CS e PRNG podem ser combinadas para dar mais variação Lintermann; Deussen (11). Hammes (6) desenvolveram algoritmos de simulação de ecossistemas para gerar mapas de posicionamento de vegetação, que combina um ecossistema com o espaço do jogo.

Kelly; McCabe (10) apresentam um visão geral dos diferentes métodos que podem ser utilizados para gerar ambientes urbanos. A geração desses ambientes, segundo eles, normalmente é iniciada com uma densa redes de estradas, onde os disponíveis espaços próximos às ruas estão subdivididos em loteamentos. Vetorização pode ser usada para encontrar os limites desses blocos malhados Sexton; Watson (18). Então, as construções podem ser geradas de acordo com as suas regras pré definidas. Para modelar o comportamento de entidades foi usado IA para que as entidades do jogo reajam às várias ações que os jogadores podem executar em um mundo virtual de uma forma que pareça inteligente. A Figura 6 mostra um possível exemplo de um ambiente urbano gerado de forma procedural.

Figura 6 – Cidade Gerada de Forma Procedural, fonte Greuter et al. (5)

4.1.2 Importância de algoritmos de ruído para geração procedural

Dentro da geração procedural, os ruídos ganham bastante atenção, principalmente na geração procedural de terreno. Isso deve-se ao fato de serem práticos, rápidos e gerar resultados

satisfatórios. Eles possuem um campo de estudo significativo dentro da computação.

Tatarchuk (21) comenta sobre as propriedades ideais dos ruídos. As mais relevantes são aquelas que o algoritmo aparenta ser aleatório, mas não é, sendo a função repetitiva capaz de entregar o mesmo valor para um ponto de entrada, pseudo-aleatório sem uma periodicidade óbvia. Também comenta-se sobre alguns tipos de ruídos como o Lattice noise e tabelas de permutações. Segundo a autora, esse é o método mais simples para se introduzir aos ruídos pois é baseado em algoritmos de repetição.

O Value Noise, segundo a autora, é a forma mais simples de gerar uma função estocástica de baixa passada e filtrada. A ideia é basicamente utilizar valores entre $[-1, 1]$ e fazer uma função de ruído baseada na interpolação desses pontos. O problema dessa técnica é que a derivada de uma função interpolada não é contínua, e por conta disso, pode gerar um resultado mais serrilhado em comparação com outros algoritmos.

O Gradiente (Perlin Noise Clássico) gera um vetor gradiente pseudo aleatório para cada ponto. Dado uma entrada P de n dimensões, para cada um dos seus pontos vizinhos, pega-se um vetor de direção pseudo aleatório e computa-se uma função linear, com isso combina-se os resultados com uma soma com pesos. O resultado é uma função suave (possui derivadas de todas as ordens) que tem um padrão regular.

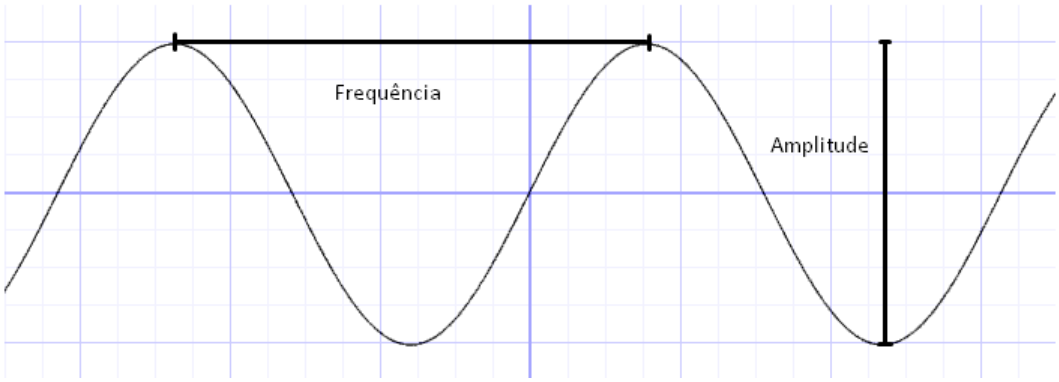

Figura 7 – Relação Entre Frequência e Amplitude

O algoritmo Fractorial Brownian Motion (FBm) soma vários ruídos, cada um com amplitude e frequência diferentes. A Figura 7 mostra a relação entre frequência e amplitude. As frequências e amplitudes estão relacionadas com a lacunaridade e ganho, respectivamente. Basicamente, a lacunaridade controla a mudança de frequências entre cada banda e o ganho controla a mudança de amplitude entre as mesmas. Ela é uma medida de como uma curva fractal enche o espaço. De forma mais contextualizada, é a proporção entre os tamanhos de sucessivas escalas de oitavas.

A Figura 8 mostra a diferença no gráfico ao se utilizar 3 lacunaridades diferentes. Utilizando-se os valores 1, 1.5 e 2 pode-se perceber que quanto maior a lacunaridade mais serrilhado o resultado aparentará. O mesmo serve para o ganho, que possui o efeito porém alterando a amplitude.

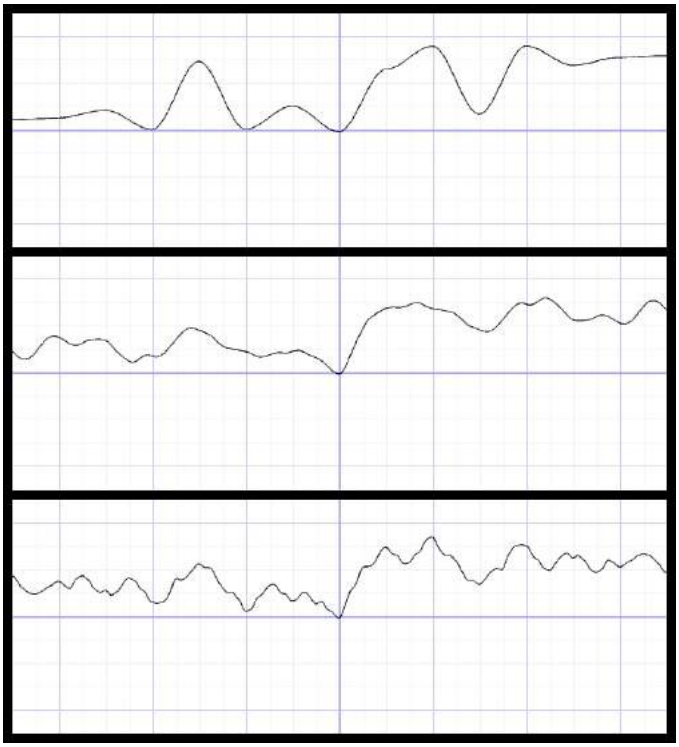

Figura 8 – Imagem Mostrando o FBm usando 3 lacunaridades Diferentes

4.1.3 Geração procedural baseada em buscas

Togelius et al. (22) comenta sobre o campo de procedimentos baseados em pesquisa na geração de conteúdo utilizando técnicas evolutivas, pesquisa estocástica e meta-heurística para a geração de conteúdo para jogos. Dentro da sua obra, o autor comenta sobre a taxonomia utilizada para analisar os exemplos de conteúdos gerados proceduralmente. No total, são apresentadas cinco métricas:

- Online ou Offline
- Conteúdo Necessário x Conteúdo Adicional
- Sementes Aleatórias x Vetor de Parâmetros
- Geração Estocástica x Geração Determinística
- Construtivo x Gera e Testa

Online ou Offline: essa métrica diz a respeito se o conteúdo é gerado durante o decorrer do jogo, ou antes, durante o carregamento. Os requerimentos para uma técnica de geração procedural ser considerada online são uma velocidade de geração rápida, que seja possível prever o tempo de execução juntamente com a qualidade do resultado. Um exemplo pode ser caso o usuário entre em uma construção, e o jogo gere o interior dessa construção instantaneamente.

Conteúdo Necessário x Conteúdo Adicional: essa métrica menciona que o conteúdo necessário pode ser definido. Por exemplo, se o jogador estiver atravessando uma masmorra, para ele concluir, ele deverá concluir tal puzzle ou derrotar determinado inimigo. O conteúdo adicional seria, então, um elemento não relacionado diretamente a esse contexto, como um baú com recompensas, que não precisam necessariamente ser coletado para o término do nível.

Sementes Aleatórias x Vetor de Parâmetros: essa métrica foca na diferença de configuração baseado em um vetor de parâmetros ou em sementes aleatórias. Nesse caso, tem-se a possibilidade de regular o aleatório. Dentro de um vetor de entrada, pode-se definir quantas casas tal cidade terá, ou como é a arquitetura das mesmas. Em contrapartida, na utilização de sementes aleatórias, a quantidade de casas e o estilo das mesmas é totalmente aleatória.

Geração Estocástica x Geração Determinística: essa métrica cobre o quão determinístico é o resultado. A geração estocástica é uma geração que possui a sua variação aumenta com o tempo. No modelo determinístico, a sua variação já está pré-determinada, produzindo sempre os mesmos resultados.

Construtivo x Gera e Testa: por fim, essa métrica versa sobre a forma de criar/evoluir o conteúdo. Um algoritmo construtivo pode ser exemplificado através da utilização de camadas. Por exemplo, supondo-se que o algoritmo gere uma cidade, primeiramente ele irá gerar o terreno, após isso se o resultado for satisfatório, ele gerará as ruas, as habitações e assim por diante. Na modalidade gera e testa, todo o conteúdo é gerado de uma única vez. O operador pode checar o conteúdo e, se esse não for plausível, gerá-lo novamente (em sua totalidade).

5 METODOLOGIA

O presente trabalho tem um cunho investigativo baseado em um experimento. Esse foi baseado no intuito de reproduzir duas cenas de um jogo comercial (Valheim) fazendo uso do algoritmo Fractional Brownian Motion (FBm). O algoritmo foi testado em dois contextos diferentes: a geração de uma cena mais simples de jogo e outra mais complexa. Dentro de cada um desses contextos, o resultado gerado pelo algoritmo foi avaliado sob a ótica de três métricas (conforme mencionado anteriormente). Essas métricas são a velocidade de geração do terreno, que está relacionado ao fato dele ser online ou offline. Outra métrica é se o algoritmo, nesse contexto, é construtivo ou se ele "gera e testa". Por fim, foi avaliada a métrica se o algoritmo permite a geração de conteúdo adicional, ou apenas o conteúdo necessário.

5.1 EXPERIMENTO

A partir da implementação do algoritmo, esse foi utilizado para a geração de duas cenas do jogo, uma mais simples e outra mais complexa, conforme ilustrado nas Figuras 9 e 10.

Figura 9 – Contexto simples que foi replicado pelo algoritmo.

Figura 10 – Contexto complexo que foi replicado pelo algoritmo.

A geração do terreno foi feita na forma que o algoritmo de ruído gere os valores em uma matriz chamada *height map* (mapa de alturas), e no mesmo foi aplicada a técnica *FBM*, entregando assim, um *noise map* (mapa de ruído) demonstrado na Figura 11. Após isso, cada quadrado da matriz é colorido baseado na sua altura. Feito isso, os quadrados e são divididos em triângulos sendo que cada vértice possui uma altura diferente gerando assim, um mapa tridimensional. Com isso, foi posicionada a grama, as árvores, as pedras e a água. Sendo assim, concluída a etapa de implementação.

A análise da primeira métrica do experimento tem foco no teste se o algoritmo possui um tempo de geração online ou offline. O algoritmo foi por 10 vezes, calculando-se a média de tempo de geração. Se o valor é inferior à 1 segundo, o algoritmo foi classificado como online, visto que não existe a necessidade de levar um jogador para uma tela de carregamento, por exemplo.

Após isso, foi verificada a característica construtiva ou a "gera e testa", isso é, caso ele gere o terreno por camadas. Por exemplo, primeiro o relevo, vegetação e assim por diante, ou se a geração leva um passo apenas. foi testada a possibilidade de gerar primeiramente apenas o relevo, em sequencia, foi selecionado o nível da água, alagando-se as partes inferiores. Por fim, foi inserido os detalhes do ambiente, como grama e árvores.

5.2 IMPLEMENTAÇÃO

O primeiro passo para o início da implementação foi fazer o download e instalação do software Unity e criar um novo projeto em 3D.

Feito isso, foram criados alguns arquivos em *C#*, sendo um deles, responsável pela criação do Noise Map e após criar o script responsável pelo retorno do *height map* junto com algumas configurações dentro da Engine, o Noise Map estava pronto.

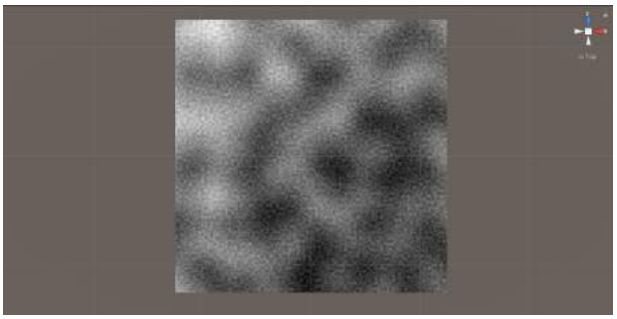

Figura 11 – Noise Map

Após isso foi criada uma lista de "regiões"para atualizar a cor do terreno baseada na posição e valor do ruído de cada quadrado

Feito isso, o terreno foi dividido em triângulos para que o terreno pudesse ficar tridimensional

Com o terreno pronto, foram posicionados os detalhes do ambiente.

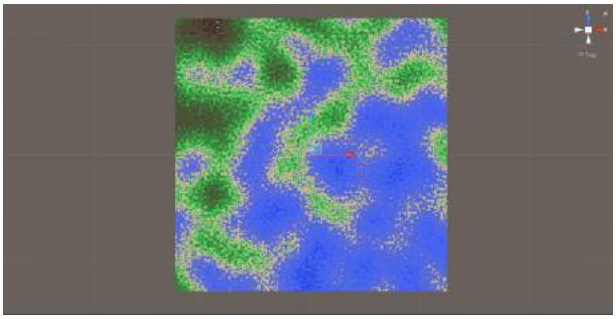

Figura 12 – Colour Map

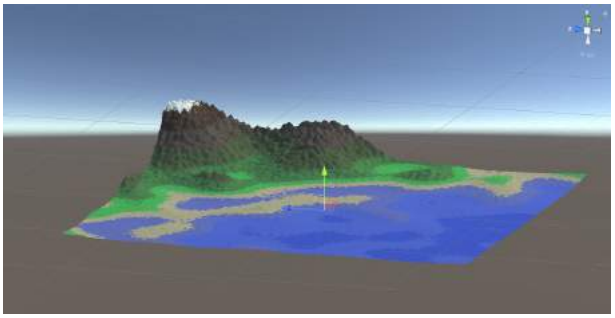

Figura 13 – Mapa 3D

Primeiramente as árvores, que para cada posição seria rodada uma probabilidade, caso verdadeira, um objeto de jogo (Game Object) é posicionado nesse quadrante.

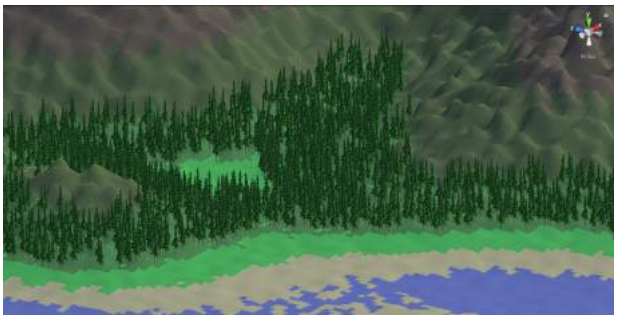

Figura 14 – Mapa 3D com Árvores

Para a grama, foi necessário percorrer cada quadrante internamente por mais um loop, pois o modelo da mesma é menor, fazendo com que assim, a aparição das gramas seja mais frequente.

Para as pedras, o modelo que foi utilizado era muito pequeno, por isso, foi necessário utilizar uma matriz de multiplicação com valores aleatórios, para que o modelo ficasse deformado fazendo com que a cena fique com um tom de diversidade.

Finalmente, a água foi feita utilizando um plano com uma textura ondulada azul clara com uma baixa transparência para que seja possível ver o que está abaixo dela. Foi mantida a pintura da água por baixo para simular a profundidade.

Para facilitar a criação das cenas, dentro da Engine, conforme ajustes na implementação foram feitos, foi-se criado variáveis de apoio para facilitar na manipulação da cena, podendo

Figura 15 – Adicionando Grama à imagem anterior

Figura 16 – Adicionando Pedras à imagem anterior

Figura 17 – Implementação dá água

editar os limites e probabilidades da geração e posicionamento dos detalhes (árvores, pedras e grama). Podendo também alterar as características do terreno, aumentado a sua persistência, lacunaridade, multiplicador de alturas, offset e semente. Podendo também alterar entre alterar automaticamente entre a atualização automática do mapa, baseada nas mudanças das variáveis acima, ou não. Conforme a Figura 18.

E com isso, deu-se por concluída a etapa da implementação.

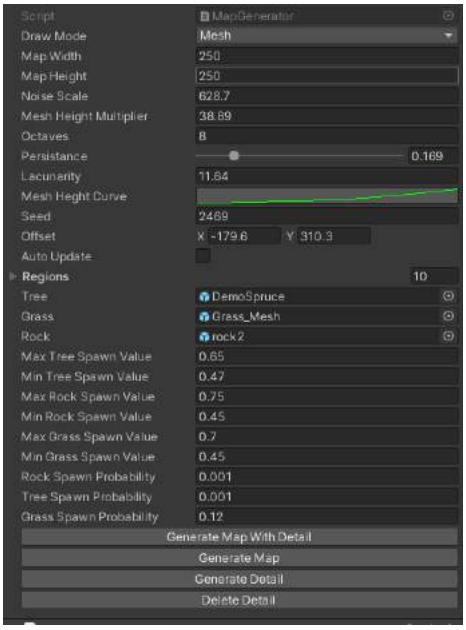

Figura 18 – GUI

6 RESULTADOS

Após o término da etapa da implementação, foi dado inicio à coleta de resultados.

6.1 GERAÇÃO DA CENA SIMPLES

Abaixo, na figura 19, é possível ver a cena simples gerada.

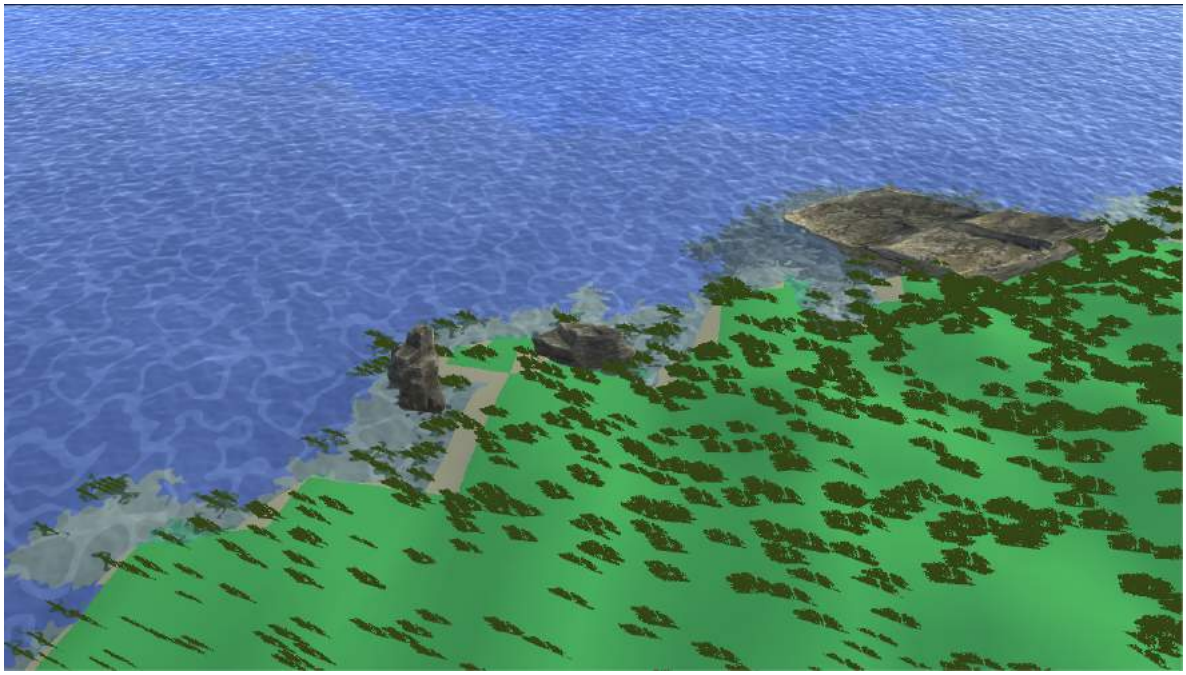

Figura 19 – Cena Simples Gerada de Forma Procedural

A cena foi gerada mais 10 vezes para se ter uma média e um desvio padrão do tempo de geração. Abaixo, na Figura 20 é possível ver os tempos de geração.

Figura 20 – Tempos de Geração da Cena Simples

O tempo médio de geração ficou em 2,47863349 segundos. O desvio padrão em 0,0708025239 segundos.

Abaixo também, é possível ver a Figura 21 mostrando lado a lado a cena proposta e a cena gerada.

Figura 21 – Comparação entre as cenas simples

6.2 GERAÇÃO DA CENA COMPLEXA

Abaixo, na Figura 22 é possível ver a cena complexa gerada.

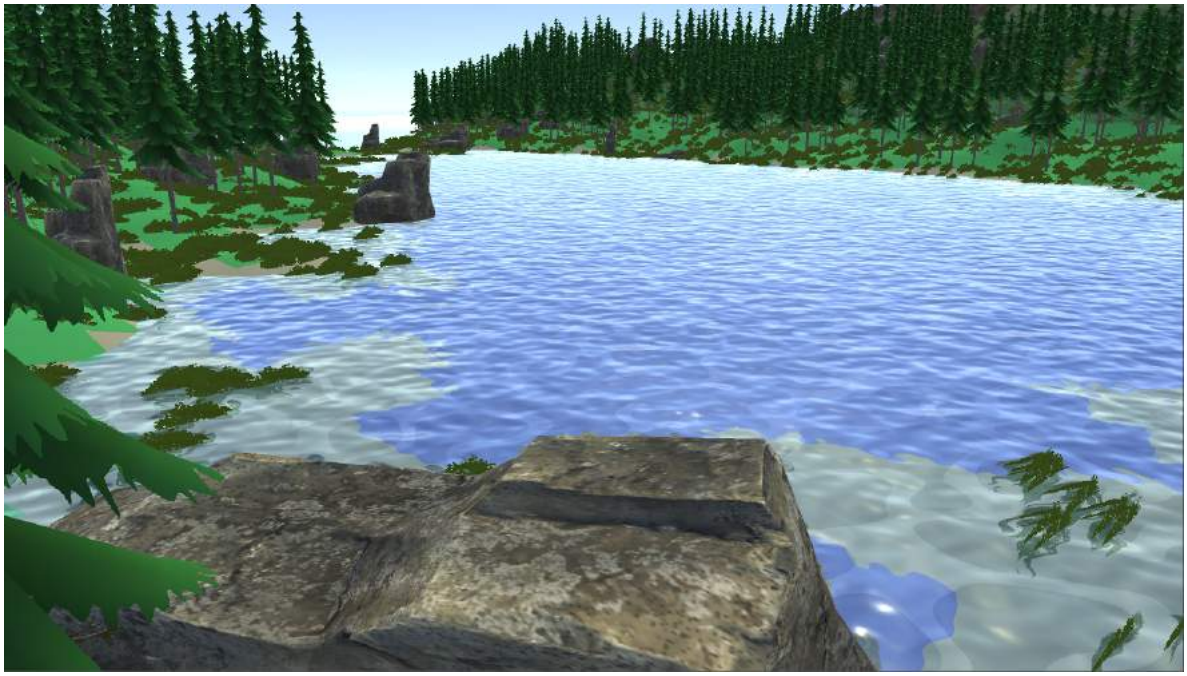

Figura 22 – Cena Complexa Gerada de Forma Procedural

Assim como na cena anterior, a mesma foi gerada 10 vezes, e obteve-se os seguintes valores conforme a Figura 23.

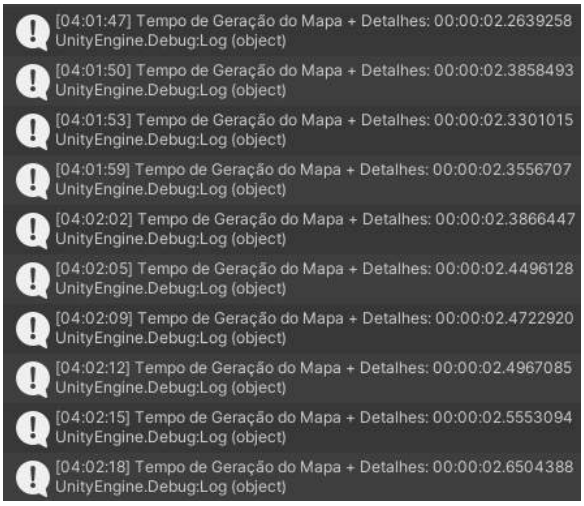

Figura 23 – Tempos de Geração da Cena Simples

O tempo médio de geração da cena ficou em 2,43465535 segundos. O valor do desvio padrão em 0,1143175149 segundos.

Finalmente, na Figura 24 é possível comparar ambas as cenas complexas.

Figura 24 – Comparação entre as cenas complexas

6.3 ESPECIFICAÇÕES DA MÁQUINA UTILIZADA

Os valores abaixo foram medidos em um computador desktop com as seguintes especificações:

- Placa de Vídeo: GTX 1050 TI 4GB VRAM;
- Processador: Intel I3 3330 3.0GHz 4 CPUs;
- RAM: 8 GB.

Por conta disso, o resultado das execuções podem variar para outras máquinas.

7 DISCUSSÃO

Nessa seção é discutido os resultados apresentados e alguns outros pontos que chamaram a atenção durante a implementação do projeto.

7.1 ANÁLISE DOS RESULTADOS

É possível ver que o tempo médio de geração das cenas, por mais que ambas possuíssem um nível de detalhe diferente, ficou com uma margem de diferença muito baixa, a cena simples, que possui um nível baixíssimo de árvores acabou tendo um tempo de geração médio de 0,01 segundos maior do que a cena complexa, que por sua vez, possui mais detalhes. O que faz total sentido, uma vez que ambas as cenas estão tendo o mesmo número de interações, com a única diferença sendo que quando o caso da probabilidade for verdadeiro ele adicionará ao mapa e a uma lista de objetos (para futura remoção dos mesmos).

O tempo médio de geração de mapa + detalhes em 2,45 segundos pode não é o suficiente para definir se a geração é online ou offline. Contudo, levando em consideração que apenas um quadrado de 255x255 foi gerado e que conforme o personagem se movimentar pelo mapa, novos quadrados deverão ser gerados caso o contrário o personagem se encontrará com apenas a opção de voltar. Além disso, supondo que isso leve no mínimo o mesmo tempo de geração do primeiro quadrado, ou seja, 2,45 segundos, o algoritmo tende a ser mais offline.

Novamente, não é simples escolher entre as técnicas Construtiva x Gera e Testa, uma vez que dependendo do contexto na qual elas forem aplicadas, uma pode ser mais eficiente do que a outra. Contudo, uma abordagem Gera e Testa é relativamente mais custosa, afinal, todos os detalhes do mapa serão posicionados e apenas então, o personagem estará livre para se locomover. Se por outro lado, uma abordagem Construtiva for implementada, após uma geração rápida do relevo, o usuário já poderia se movimentar pelo mesmo, ainda que inacabado, diminuindo o tempo de espera na tela de carregamento.

A abordagem relacionada à técnica de Conteúdo Adicional x Conteúdo Necessário por sua vez apresenta algumas questões que devem ser pontuadas. A principal delas é que o conteúdo adicional transforma o estilo de geração em Construtivo, pois você está adicionando conteúdo à um mapa que já foi gerado e levando em consideração que a adição de mais objetos na cena irá impactar no tempo de geração, acontecerá da cena ficar com características Construtivas e possivelmente offlines. Portanto, colocar Conteúdo Adicional é possível, mas removerá a característica Gera e Testa e aumentará o tempo de geração da cena, tornando-a possivelmente offline.

7.2 SOBREPOSIÇÃO DE OBJETOS

Uma questão que chamou a atenção durante a implementação é que nada foi feito para evitar que um objeto esteja "dentro"do outro. O posicionamento de objetos é feito utilizando o *height map* e calculando a probabilidade de geração dentro do intervalo para cada tipo de objeto. Por exemplo, como é possível ver na Figura 18, o intervalo de geração para as árvores (Trees) está entre (0.47;0.65), com a probabilidade de 0.001. Logo, nada impede de uma árvore estar dentro de uma pedra, por exemplo. Esses testes de colisões podem acarretar em uma carga de processamento ainda maior, e como não é crucial para o funcionamento do trabalho, não foi implementada nenhuma solução para isso.

7.3 DETALHE DA GRAMA

Na figura 19, no canto inferior esquerdo fica evidente um defeito visual do modelo da grama utilizado no projeto. A grama é um plano sem profundidade com a propriedade de "culling"(cortar superfícies que não são "vistas"pela câmera) aplicada. Com isso, supondo que a câmera se rotacione 180°, nenhuma grama seria visível. Uma solução possível seria fazer com que todos os objetos grama rotacionassem de tal forma em que a face que não é cortada ficasse de frente com a câmera. Contudo, o motivo pelo qual isso não foi implementado é que a sabendo que as dimensões do mapa são 255x255 e que a geração da grama em especial, para cada quadrado, possui um laço de repetição particular que executa mais 10 vezes, a quantidade de grama presente no mapa pode chegar a até 650250 unidades. Fazer toda grama rotacionar para cada posição da câmera reduziria consideravelmente a taxa de quadros/segundo, uma vez que sem a rotação, como possível ver nos resultados, a cena simples, que por sua vez possui a probabilidade de criação de um fardo de 0.65, ocupando a 40 por cento do terreno (o intervalo de criação está definido em 0.4 até 0.8), em comparação com a cena complexa que possui 0.30 ocupando o mesmo intervalo de criação, fez com que ambas as cenas possuíssem um tempo médio de geração muito similar, logo, apesar de não impactar no tempo de geração, reduziria a taxa de quadros por segundo caso fosse aplicada de forma iterativa.

8 CONCLUSÃO

Este trabalho apresentou uma implementação e análise da técnica Fractional Brownian Motion (FBM) que consiste em somar oitavas de ruídos com frequências menores. O algoritmo foi avaliado dentro das três técnicas que foram propostas, Gera e Testa x Construtiva, Online x Offline e Conteúdo Adicional x Conteúdo Necessário. Foi testado experimentalmente em dois contextos de cenas de um jogo digital 3D, uma simples, com menos detalhes visuais e outra relativamente mais complexa.

Ao longo do trabalho pode-se constatar que o método de geração procedural FBM tornou viável a criação de um mapa em um tempo baixo resolvendo em partes, assim, a premissa inicial.

Percebeu-se também, que o algoritmo é muito eficiente para criar as suas paisagens próprias. Contudo, ao tentar recriar uma cena gerada de forma procedural, foi possível ver que esse não é o forte do algoritmo, muitas tentativas precisaram ser feitas até ser gerada uma cena satisfatória, o que por sua vez faz sentido a final, é uma geração procedural tentando recriar outra geração procedural.

As técnicas que foram testadas, de geração Online x Offline, Conteúdo Adicional x Conteúdo Necessário e Geração Construtiva x Gera e Testa, que no início aparentavam ter pouco em comum, acabaram se mostrando extremamente relacionadas. O tempo de geração está diretamente relacionado à quantidade de objetos presentes na tela, que por sua vez, está relacionado com o Conteúdo Adicional, além do mais, ao adicionar mais conteúdo a um mapa que já está pronto, ele perde a característica Gera e Testa e passa a ser Construtivo.

No geral, o resultado do trabalho foi satisfatório, as técnicas que foram propostas foram todas testadas. O FBM mostrou-se extremamente eficiente na geração dos mapas, permitindo que vários parâmetros fossem alterados para alcançar um terreno mais desejável, desde uma série de ilhas caso necessário até uma longa planície, as técnicas que foram avaliadas, que em um primeiro momento aparentavam estarem desconectadas, mostraram-se extremamente relacionadas, sendo assim, necessário encontrar um ponto de equilíbrio entre detalhe e tempo.

8.1 TRABALHOS FUTUROS

Uma possível continuação seria a implementação de alguma técnica que gerasse novos quadrados conforme a câmera se movimentasse em direção aos limites do mapa, dando assim, a impressão de infinidade no mesmo.

Outra possível sequencia seria corrigir a grama e o posicionamento das pedras e árvores, verificando antes de cada posicionamento se já existe algum objeto nessa posição, corrigindo assim, a sobreposição de objetos.

REFERÊNCIAS

1 BRATHWAITE, B; SCHREIBER, I. **Challenges for Game Designers . Rockland, MA: Charles River Media**. [S.l.]: Inc, 2008.

2 CHRISKRONEN. **Terraria: Map viewer**. [S.l.: s.n.], 2011. Acessado em 12 de Maio de 2021. Disponível em: <https://www.deviantart.com/chriskronen/art/Terraria-Map-viewer-210241370>.

3 DGREEN. **Rede de Estradas Gerada de Forma Procedural**. [S.l.: s.n.], 2014. Acessado em 10 de Maio de 2021. Disponível em: <http://www.radioactivesoftware.com/content/making-lots-progress-procedural-city-generation>.

4 FLAKE, Gray William. **The computational beauty of nature, A Bradford book**. [S.l.]: The MIT Press, Cambridge, 1999.

5 GREUTER, Stefan et al. Real-time procedural generation ofpseudo infinite'cities. In: PROCEEDINGS of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia. [S.l.: s.n.], 2003. 87–ff.

6 HAMMES, Johan. Modeling of ecosystems as a data source for real-time terrain rendering. In: DIGITAL Earth Moving. [S.l.]: Springer, 2001. p. 98–111.

7 HARTSOOK, Ken et al. Toward supporting stories with procedurally generated game worlds. In: IEEE. 2011 IEEE Conference on Computational Intelligence and Games (CIG'11). [S.l.: s.n.], 2011. p. 297–304.

8 HENDRIKX, Mark et al. Procedural content generation for games: A survey. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)**, ACM New York, NY, USA, v. 9, n. 1, p. 1–22, 2013.

9 IOSUP, Alexandru. POGGI: generating puzzle instances for online games on grid infrastructures. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 23, n. 2, p. 158–171, 2011.

10 KELLY, George; MCCABE, Hugh. A survey of procedural techniques for city generation. **ITB Journal**, Citeseer, v. 14, n. 3, p. 342–351, 2006.

11 LINTERMANN, Bernd; DEUSSEN, Oliver. Interactive modeling of plants. **IEEE Computer Graphics and Applications**, IEEE, v. 19, n. 1, p. 56–65, 1999.

12 MUNDO Gerado no Jogo Valheim de Forma Procedural. [S.l.: s.n.], 2021. Acessado em 10 de Maio de 2021. Disponível em: <https://re-actor.net/valheim-world-map/>.

13 MUSGRAVE, F Kenton et al. **Texturing and modeling: a procedural approach**. [S.l.]: Academic Press Professional, Inc., 1994.

14 NORMAN, DA. **The design of everyday things Basic Books**. [S.l.]: USA, 2002.

15 PARISH, Yoav IH; MÜLLER, Pascal. Procedural modeling of cities. In: PROCEEDINGS of the 28th annual conference on Computer graphics and interactive techniques. [S.l.: s.n.], 2001. p. 301–308.

16 PELL, Barney. Metagame in symmetric chess-like games. University of Cambridge Computer Laboratory, 1992.

17 RANDI ROST, Fort Collins. **Exemplo de soma de ruídos utilizando o FBm**. [S.l.: s.n.], 2003. Acessado em 10 de Maio de 2021. Disponível em: <http://www.yaldex.com/open-gl/ch15lev1sec1.html>.

18 SEXTON, Chris; WATSON, Benjamin. Vectorization of gridded urban land use data. In: PROCEEDINGS of the 2010 Workshop on Procedural Content Generation in Games. [S.l.: s.n.], 2010. p. 1–8.

19 SMELIK, Ruben M et al. A survey of procedural methods for terrain modelling. In: PROCEEDINGS of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS). [S.l.: s.n.], 2009. v. 2009, p. 25–34.

20 TAKATSUKI, Yo. Cost headache for game developers. **BBC News**, v. 27, 2007.

21 TATARCHUK, Natalya. The Importance of Being Noisy: Fast, High Quality Noise. **3D Application Research Group, AMD Graphics Products Group**, 2007.

22 TOGELIUS, Julian et al. Search-based procedural content generation: A taxonomy and survey. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v. 3, n. 3, p. 172–186, 2011.

23 UNZAGA, Brett. **Random Dungeon Generator**. [S.l.: s.n.], 2014. Acessado em 10 de Maio de 2021. Disponível em: <https://experiments.withgoogle.com/random-dungeon-generator>.