DESENVOLVIMENTO DE TRANSIÇÕES SUAVES ENTRE BIOMAS EM AMBIENTES PROCEDURAIS TRIDIMENSIONAIS UTILIZANDO O ALGORITMO DE MARCHING CUBES

LUIZ PHILIPPE PEREIRA AMARAL

DESENVOLVIMENTO DE TRANSIÇÕES SUAVES ENTRE BIOMAS EM AMBIENTES PROCEDURAIS TRIDIMENSIONAIS UTILIZANDO O ALGORITMO DE MARCHING CUBES

Projeto Orientado em Computação, elaborado como requisito para conclusão do curso de Ciência da Computação da Universidade Federal de Minas Gerais

Orientador: Renato Antônio Celso Ferreira

Belo Horizonte

Julho de 2024

Pereira Amaral, Luiz Philippe

Desenvolvimento de Transições Suaves entre Biomas em Ambientes Procedurais Tridimensionais Utilizando o Algoritmo de Marching Cubes / Luiz Philippe Pereira Amaral. — Belo Horizonte, 2024
viii, 18 f. : il. ; 29cm

Projeto Orientado a Computação — Universidade Federal de Minas Gerais
Orientador: Renato Antônio Celso Ferreira

I. Título.

Para Rita, Michelle, Elza, família e amigos.

Resumo

Este projeto avança na exploração da geração procedural de ambientes tridimensionais, centrando-se na implementação e otimização de transições suaves entre múltiplos biomas usando o algoritmo de Marching Cubes em GPU. O objetivo foi criar uma metodologia que permitisse a representação de variados biomas com transições visuais coerentes e naturalísticas, melhorando a imersão e a diversidade dos ambientes gerados. Utilizamos Unity 3D com programação em C para a CPU e HLSL para a GPU, desenvolvendo um sistema que emprega mapas de ruído e interpolação de características para definir e integrar biomas de maneira dinâmica. O projeto destacou-se pela introdução de um método inovador de codificação binária para biomas, facilitando a geração de regiões de transição com base em valores interpolados. Os resultados demonstraram melhorias significativas na fluidez e na qualidade visual dos ambientes procedurais, oferecendo novas possibilidades para aplicações em jogos, simulações educativas e experiências de realidade virtual. Este estudo não só confirmou a viabilidade técnica das técnicas propostas mas também abriu caminhos para futuras investigações sobre otimizações adicionais e aplicações práticas.

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A demanda por ambientes virtuais ricos em detalhes e dinâmicos está crescendo, impulsionada pelos avanços no desenvolvimento de jogos de vídeo, simulações educacionais e aplicações de realidade virtual. As técnicas de geração procedural, que criam ambientes dinamicamente baseados em princípios algorítmicos, desempenham um papel essencial nesses campos ao oferecer uma solução escalável para o problema de criar manualmente mundos complexos e de grande escala. Este projeto constrói sobre a implementação bem-sucedida anterior do algoritmo Marching Cubes, que se concentrou na geração de terreno, para aprimorar ainda mais a capacidade e o realismo dos ambientes procedurais.

1.1 Contextualização

O algoritmo Marching Cubes, introduzido pela primeira vez por Lorensen e Cline em 1987, revolucionou a criação de ambientes tridimensionais a partir de dados volumétricos, fornecendo uma base para o desenvolvimento de terrenos detalhados e interativos em tempo real. A capacidade deste algoritmo de gerar malhas complexas de maneira eficiente o tornou uma ferramenta valiosa para criar paisagens realistas e formas arquitetônicas em configurações virtuais.

A fase inicial do projeto focou na otimização do algoritmo Marching Cubes para execução em GPU, alcançando melhorias significativas em velocidades de processamento e detalhes visuais. O algoritmo foi adaptado para lidar com várias funções de densidade dinamicamente, o que permitiu a geração de terrenos mais intrincados e variados. Este trabalho estabeleceu a base para uma exploração mais aprofundada de técnicas avançadas de geração procedural.

1.2 Definição da tarefa

Iniciaremos definindo o que trataremos como um bioma neste projeto. O dicionário de Cambridge apresenta a seguinte definição para *biome*:

> a region of the Earth's surface and the particular combination of weather conditions, plants, and animals that are found in it

Buscaremos representar essas variações de combinações climáticas com elementos visuais disintos na atmosfere, solo e relevo de uma região do ambiente virtual gerado. Uma região com seu conjunto de características compreenderá um bioma.

Com as capacidades básicas de geração de terreno estabelecidas, o projeto atual visou introduzir múltiplos biomas com transições suaves entre eles, aumentando assim a diversidade ambiental e o apelo visual das paisagens geradas. Esta fase teve como alvo o desenvolvimento de um sistema que pudesse não apenas manejar várias zonas ecológicas, mas também mesclá-las de maneira que imita as transições naturais encontradas em ecossistemas do mundo real. Os objetivos incluíam:

1. Desenvolver um método para definir e integrar múltiplos biomas usando algoritmos procedurais.
2. Implementar transições suaves entre esses biomas para garantir uma experiência visual sem emendas.
3. Otimizar o algoritmo Marching Cubes expandido para processar eficientemente essas transições de biomas em GPUs, garantindo desempenho em tempo real.
4. Ao abordar esses objetivos, este projeto busca expandir os limites do que pode ser alcançado com a geração procedural, criando ambientes virtuais mais imersivos e realisticamente variados que podem ser adaptados para uma ampla gama de aplicações.

1.3 Motivação

A motivação para este projeto surge da necessidade crescente de criar ambientes virtuais que não apenas capturam a imaginação do usuário, mas também oferecem uma experiência imersiva e realista. A geração procedural oferece uma solução potente para essas demandas, permitindo a criação de mundos extensos e detalhados sem a necessidade de modelagem manual intensiva, que é tanto demorada quanto cara.

Além disso, a capacidade de integrar múltiplos biomas com transições realistas entre eles pode significativamente enriquecer a experiência do usuário, proporcionando ambientes mais vibrantes e variados que refletem mais fielmente a diversidade do mundo natural.

Esta fase do projeto, portanto, busca não apenas avançar a tecnologia por trás da geração procedural, mas também responder a uma demanda de mercado para ambientes mais sofisticados e envolventes. Ao fazê-lo, espera-se abrir novas possibilidades para desenvolvedores e designers em várias indústrias, incentivando a inovação e elevando o padrão de qualidade para simulações digitais e experiências virtuais.

Capítulo 2

Fundamentação e trabalhos relacionados

Este capítulo é dedicado à exploração dos fundamentos essenciais que sustentam o projeto de geração procedural de ambientes tridimensionais, com foco no algoritmo de Marching Cubes. Além disso, serão revisadas as pesquisas relacionadas que delineiam o estado atual do campo, contextualizando a relevância e contribuições do presente trabalho.

2.1 Conceitualização

Iniciaremos discutindo os princípios fundamentais da geração procedural, destacando as técnicas e algoritmos-chave que constituem a base teórica deste projeto. Serão abordados conceitos relacionados à representação de terrenos, malhas tridimensionais e a influência das funções de densidade na criação de ambientes realistas.

Posteriormente, a análise de pesquisas relacionadas será apresentada, explorando trabalhos recentes e significativos no âmbito da geração procedural de ambientes tridimensionais. Esta revisão da literatura visa identificar lacunas no conhecimento existente e ressaltar as inovações e desafios enfrentados pelos pesquisadores na área.

Ao unir os fundamentos teóricos com as pesquisas relacionadas, este capítulo estabelece um contexto sólido para o desenvolvimento do projeto, fundamentando as escolhas metodológicas e destacando a contribuição única que este trabalho propõe no campo da geração procedural de ambientes tridimensionais.

2.1.1 Natureza da Geração Procedural

A geração procedural é uma técnica fundamental na criação de ambientes tridimensionais dinâmicos e autênticos. Neste projeto, exploramos a natureza intrínseca desse processo, compreendendo como algoritmos e técnicas podem ser empregados para criar cenários que não apenas mimetizam a complexidade da natureza, mas também oferecem interatividade e adaptabilidade. O cerne da geração procedural é elaborar um procedimento, geralmente matemático, para criar conteúdo de forma continua baseado em um conjunto de entradas.

2.1.2 Algoritmo de Marching Cubes

O algoritmo de Marching Cubes, proposto por Lorensen e Cline em 1987, é a espinha dorsal deste projeto. Exploraremos a fundo os princípios que fundamentam esse método, compreendendo como ele converte dados volumétricos em malhas tridimensionais. Serão abordadas as nuances da implementação do Marching Cubes, particularmente no contexto do ambiente Unity e da linguagem HLSL.

2.1.3 Funções de Densidade

As funções de densidade desempenham um papel crucial na execução do Marching Cubes, moldando a aparência e a estrutura dos ambientes gerados. Este projeto busca conceituar e explorar diferentes tipos de funções de densidade, desde aquelas que modelam terrenos até as que simulam elementos específicos da natureza. A compreensão detalhada dessas funções é vital para ajustar e personalizar a geração procedural conforme as necessidades específicas da aplicação.

2.1.4 Ambiente Unity e Linguagem HLSL

O ambiente de desenvolvimento Unity oferece uma plataforma robusta para a criação de ambientes virtuais, integrando gráficos avançados e física realista. A escolha de incorporar a linguagem de shading HLSL para programação na GPU visa otimizar o desempenho computacional, permitindo uma execução eficiente das funções de densidade, essenciais para a geração em tempo real.

Esses conceitos fornecem a base teórica essencial para o desenvolvimento do projeto, abrindo caminho para uma análise mais aprofundada e uma implementação informada na geração procedural de ambientes tridimensionais.

2.2 Trabalhos relacionados

A geração procedural de ambientes tridimensionais tem sido extensivamente abordada na literatura especializada, refletindo avanços significativos e pesquisas inovadoras. No que diz respeito à natureza da geração procedural, autores como Parish e Müller exploram amplamente a aplicação dessa técnica em terrenos realistas em seu artigo seminal "Procedural Modeling of Cities"Parish & Müller [2001]. Este trabalho destaca a importância da flexibilidade algorítmica na criação de ambientes urbanos complexos.

No âmbito do algoritmo de Marching Cubes, a pesquisa de Lorensen e Cline, intitulada "Marching Cubes: A High-Resolution 3D Surface Construction Algorithm"Lorensen & Cline [1987], oferece os fundamentos essenciais deste método. Abordagens mais contemporâneas, como o trabalho de Chang et al. em "Interactive marching cubes algorithm for intraoral scanners"Chang et al. [2017], discutem otimizações cruciais para a execução eficiente do algoritmo, particularmente em ambientes GPU.

A influência das funções de densidade na geração procedural é destacada no trabalho de Musgrave et al., "The Synthesis and Rendering of Eroded Fractal Terrains"Musgrave et al. [1989]. Esta pesquisa explora a aplicação de funções de densidade fractal na modelagem de terrenos, fornecendo insights valiosos sobre a relação entre a formulação da função e as características do ambiente gerado.

No contexto do Unity e da linguagem HLSL, trabalhos como "GPGPU programming for games and science"de Eberly e David H. Eberly [2014] oferecem uma compreensão prática da implementação de shaders na Unity utilizando HLSL. Essa referência prática é valiosa para a otimização e integração eficiente de algoritmos de geração procedural no ambiente Unity.

Esses trabalhos representam uma base sólida de conhecimento, fornecendo suporte teórico e prático para os fundamentos e as pesquisas relacionadas ao projeto de geração procedural de ambientes tridimensionais. A revisão dessas referências contribui para uma compreensão abrangente e informada do estado atual do campo.

Capítulo 3

Execução

Este capítulo descreve detalhadamente a execução prática do projeto, desde a concepção inicial até a implementação efetiva das técnicas de geração de biomas e suas transições. A execução abrange o desenvolvimento de algoritmos, a aplicação de tecnologias e a solução de desafios técnicos.

3.1 Algoritmo

O projeto utilizou o algoritmo de Marching Cubes, adaptado para suportar a geração procedural de múltiplos biomas e suas transições suaves. A execução foi dividida em várias etapas principais:

1. Algoritmo Base: Desenvolvimento do algoritmo de Marching Cubes tradicional realizado no trabalho anterior.

2. Definição de Biomas: Cada bioma foi definido por um conjunto de parâmetros numéricos que representam propriedades ambientais como temperatura, precipitação e topografia. Estes parâmetros foram codificados em uma forma binária possibilitando a identificação de um conjunto de características com um número inteiro. Temos 4 propriedades, portanto utilizamos 4 bits para representar uma combinação de características, resultando assim em 16 possíveis biomas.

3. Geração de Mapas de Ruído: Utilizamos mapas de ruído sobrepostos para determinar a distribuição dos atributos dos biomas no terreno. Isso permitiu uma variação realista que foi crucial para as áreas de transição entre biomas.

4. Elaboração das Características dos Biomas: As propriedades dos biomas devem resultar em ambientes visualmente distintos em termos de atmosfera, relevo e texturas. Para isso é necessário implementar funções de densidade e *shaders* específicos para cada um dos biomas que por fim serão combinados.

5. Implementação de Transições: Desenvolvemos um método para calcular transições suaves entre biomas, onde as áreas de transição foram geradas pela combinação das funções geradores de características baseados nos valores dos mapas de ruído.

3.2 Implementação

A implementação inclui elaboração de códigos C que serão executados em CPU e códigos HLSL para GPU. Temos como ponto de partida o projeto desenvolvido em "Geração de ambientes complexos com algoritmos procedurais executados em GPU"Pereira [2023].

3.2.1 Combinação de Características

As propriedades de um dado ponto no ambiente gerado vêm da amostragem de uma sobreposição de 4 mapas de ruído em tons de cinza. Essa amostragem resultará em 4 valores entre 0 e 1. O objetivo desse algoritmo é mapear esses quatro valores para uma combinação de biomas, sendo cada um desses biomas representado por 4 bits. Assim, podemos definir a função C que realiza a combinação das propriedades embientais para o algoritmo como:

$$\begin{aligned} C(p) &= t \\ p \in \mathbb{R}^4 &: 0 \geq p_i \geq 1 \\ t \in \mathbb{R}^{16} &: \sum t = 1 \end{aligned} \tag{3.1}$$

Seja $D_i(x, y, z) = d$ a função de densidade para o bioma i na posição (x, y, z). Seja $S_i(x, y, z, n) = c$ a função do *shader* de calcula a cor c (em formato RGBA) de uma superficie do bioma i na posição (x, y, z) em um plano de normal n. E por fim $A_i(y) = c_a, c_w, r$ a função que calcula as cores c_a e c_w da atmosfera e da água respectivamente e a densidade do efeito de neblina r do bioma i em altitude y.

O resultado da soma de dois biomas $B_a = \{D_a, S_a, A_a\}$ e $B_b = \{D_b, S_b, A_b\}$ consiste em um bioma $B_y = \{D_a + D_b, S_a + S_b, A_a + A_b\}$. A multiplicação de um bioma $B_a = \{D_a, S_a, A_a\}$ por um escalar t_a resulta em um bioma $B_y = \{D_a \times t_a, S_a \times t_a, A_a \times t_a\}$.

Podemos definir o conjunto de regras do bioma i com as definições de D,S e A, portanto, $B_i = \{D_i, S_i, A_i\}$. Podemos realizar a combinação linear de dois ou mais biomas utilizando $t = C(p)$ como coeficientes. Assim, nosso bioma B_x pode ser definido com a combinação de todos biomas baseado em um dado conjunto de propriedades p com

$$B_x = \sum B_i \times t_i$$

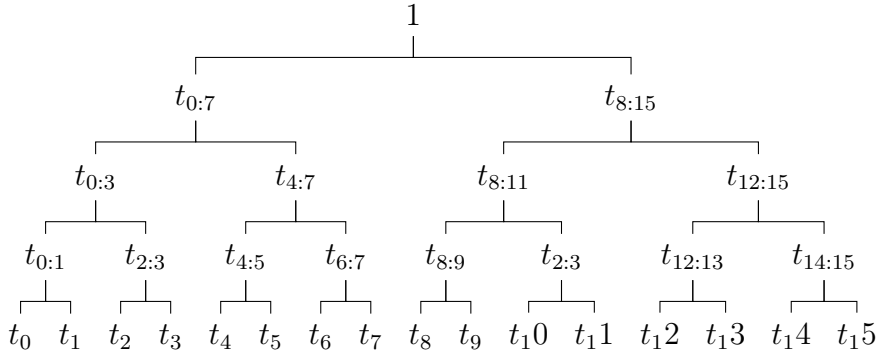
Um bioma formado pela combinação de um ou mais biomas apresenta traços de características de todos biomas utilizados na sua composição cujo coeficiente é maior que zero. Um bioma B_y formado por uma combinação de biomas $B_0..B_n$ apresenta mais características de um bioma B_i quando seu coeficiente t_i se aproxima de 1. Essas propriedades fazem com que a combinação linear dos biomas seja uma estratégia promissora para gerar a distribuição dos biomas pelo ambiente bem como as transições entre eles. Para realizar a implementação dessa estratégia precisamos definir a função C e determinar entradas apropriadas. Discutiremos ambos os pontos a seguir.

3.2.2 Função de Combinação

Foram implementadas duas versões do algoritmo de combinação de propriedades, uma delas utilizando uma árvore (1) e a outra com auxílio de um vetor de valores de combinação (2). Como o bioma de uma região afeta diretamente suas características topográficas, é importante que os coeficientes de combinação estejam disponíveis para o código HLSL que, efetivamente, é responsável por calcular a função de densidade que determina o relevo. Esse requisito favorece a versão do algoritmo baseada em vetores, já que árvores são, por definição, estruturas recursivas, que não são suportadas por HLSL devido a natureza da arquitetura das GPUs. Por esses motivos a versão final da implementação aplica o algoritmo 2, no entanto, ainda discutiremos o algoritmo 1 como um caminho para visualizar a intuição por trás da nossa estratégia, já que ele possui vantagens em termos de visualização e intuitividade.

A primeira versão implementada do algoritmo de combinação de propriedades ilustra a intuição do processo com um árvore binária. Dado um conjunto de propriedades $p =$

$\{p_0, p_1, p_2, p_3\}$, podemos elaborar onde as folhas representam coeficientes dos biomas. A árvore a seguir nosso cenário com quatro propriedades de ambiente.



A raiz da árvore é inicializada com o valor 1, a partir daí, para cada propriedade definiremos seu coeficiente c determinando um intervalo de transição $[p_{min}, p_{max}]$ entre zero e um, isso serve para evitar que os biomas percam sua identidade, diremos que o coeficiente de valores abaixo desse intervalo são categoricamente zero, enquanto o coeficiente de valores acima dele são categoricamente 1. Caso o valor da propriedade esteja dentro do intervalo de transição realizamos a interpolação inversa da propriedade nesse intervalo, que resultará em um valor para o coeficiente entre zero e um. O filho da esquerda do nó recebe o valor resultante desse processo multiplicado pelo próprio valor do nó. O filho da direita recebe $1 - c$ multiplicado pelo coeficiente pai. As folhas possuem o valor dos coeficientes finais. O algoritmo (1) demonstra essa implementação.

Embora o algoritmo (1) funcione, sua implementação recursiva dificulta sua tradução para HLSL, por isso, é utilizado um algoritmo semelhante que utiliza vetor de coeficientes no lugar do árvore. Fazendo uma analogia com o algoritmo anterior, podemos imaginar que nessa implementação temos um vetor cujo tamanho é sempre um quadrado perfeito igual a $2^{|p|}$, nesse cenário cada subdivisão em fatias iguais do vetor atua como uma sub-árvore e armazenamos apenas o último nível calculado pelo algoritmo. O algoritmo (2) representa essa implementação.

Uma vez que esses coeficientes são calculados na GPU, podemos renderizá-los em uma textura que será utilizada tanto pelas funções em GPU quanto em CPU.

3.2.3 Mapeamento de Propriedades

As propriedades apresentadas por um ponto qualquer no espaço do ambiente gerado devem seguir uma coesão geográfica para que o resultado obtido seja realista, por esse motivo, utilizamos um mapa de ruído de Perlin para cada propriedade estabelecida.

Algorithm 1 FeatureCombination (Tree)

```
root ← 1
procedure expandTree(node)i
    if i = 4 then return
    end if
    feature ← getFeature(i)
    c ← inverseLerp(lowerBound, upperBound, feature)
    if feature ≤ lowerBound then
        c ← 0
    end if
    if feature ≥ upperBound then
        c ← 1
    end if
    node.addLeftChild(c × node)
    node.addRightChild((1 − c) × node)
    expandTree(node.left, i + 1)
    expandTree(node.right, i + 1)
end procedure
expandTree(root, 0)
```

Algorithm 2 FeatureCombination (Array)

```
values ← [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
featureCount ← 4
biomeCount ← featureCount^2
for depth ← 0..featureCount do
    for index ← 0..biomeCount do
        slice ← biomeCount / 2^depth
        leftIndex = index
        rightIndex = index + slice/2
        c ← inverseLerp(lowerBound, upperBound, feature)
        if feature ≤ lowerBound then
            c ← 0
        end if
        if feature ≥ upperBound then
            c ← 1
        end if
        values[leftIndex] ← (c × values[index])
        values[rightIndex] ← ((1 − c) × values[index])
        index ← index + biomeCount / 2^depth
    end for
end for
```

Podemos pensar nesses mapas como uma única textura de quatro canais (RGBA) onde cada canal se associa a um propriedade. Diferente da função de densidade, nesse caso estamos apenas interessados na posição geográfica em um plano (x, z) paralelo ao solo do ambiente. Portanto, podemos realizar uma amostragem da textura de ruídos utilizando as coordenadas (x, y) para determinar os valores dessas propriedades para cada ponto no espaço.

3.3 Resultados

Este capítulo apresenta os resultados obtidos ao longo do projeto, destacando as melhorias na geração de biomas e na transição entre eles. As imagens incluídas ilustram as capacidades visuais alcançadas e os efeitos das técnicas de interpolação implementadas.

3.3.1 Visualização dos Biomas

A figura 3.1 mostra os diferentes biomas gerados proceduralmente. Cada bioma apresenta características únicas, refletindo as variáveis ambientais definidas.

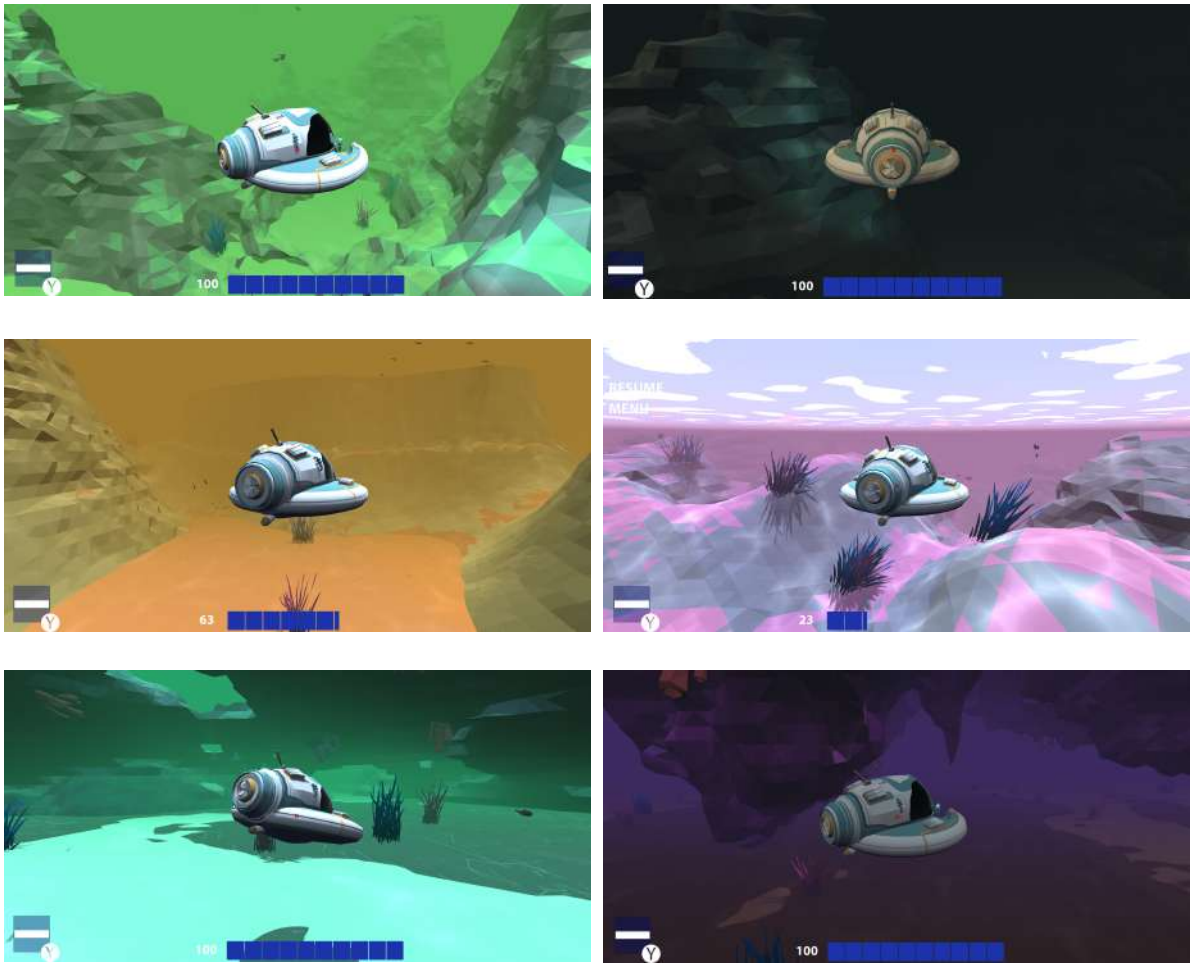


Figura 3.1. Diferentes biomas gerados com o sistema.

3.3.2 Transições entre Biomas

A figura 3.2 ilustra as transições suaves entre diferentes biomas. Note a graduação suave e naturalista nas áreas de transição, demonstrando a eficácia do algoritmo de interpolação, nessa figura, as cores do terreno representam o bioma para melhor visualização.

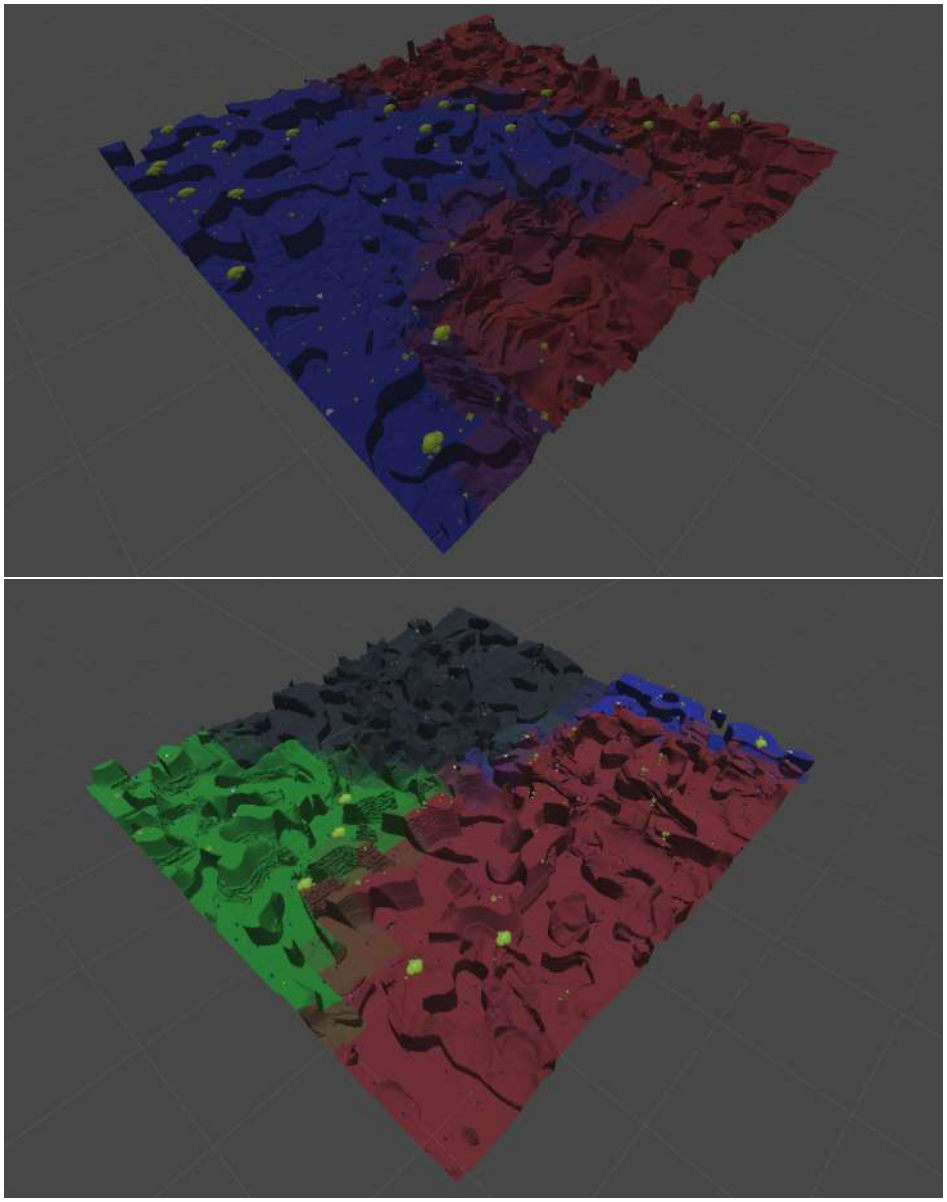


Figura 3.2. Transições suaves entre biomas.

O sistema também emite uma saída detalhando os biomas utilizados na composição da localização atual do usuário exibida na figura 3.3.

Figura 3.3. Descrição de composição dos biomas.

Capítulo 4

Conclusão

Este projeto alcançou avanços na geração procedural do ambientes em relação a iteração anterior, especificamente no desenvolvimento e na implementação de múltiplos biomas com transições suaves entre eles. Os objetivos estabelecidos foram atendidos com sucesso, demonstrando a viabilidade de técnicas avançadas de interpolação e otimização de algoritmos para a execução em GPUs.

4.0.1 Reflexões sobre os Objetivos Alcançados

Os resultados obtidos confirmam que a introdução de um método robusto para definir e integrar biomas variados, e para gerenciar suas transições de maneira suave, enriquece significativamente a qualidade visual e a imersão dos ambientes gerados. A implementação bem-sucedida dessas técnicas em uma plataforma como o Unity 3D e a otimização para desempenho em tempo real são conquistas notáveis que oferecem novas possibilidades para aplicações em jogos, simulações e realidade virtual.

4.0.2 Impacto e Aplicações Práticas

Este projeto não apenas avança o conhecimento técnico na área de geração procedural, mas também fornece uma base sólida para futuras inovações que podem ser aplicadas em um espectro amplo de indústrias. A capacidade de criar ambientes mais realistas e responsivos tem implicações diretas para a melhoria da experiência do usuário em contextos interativos e imersivos.

4.0.3 Direções Futuras

Olhando para o futuro, há várias áreas promissoras para continuação deste trabalho:

- **Exploração de Novas Técnicas de Otimização:** A busca por métodos ainda mais eficientes para a execução de algoritmos complexos em GPUs pode levar a melhorias adicionais no desempenho e na qualidade.
- **Adaptação para Diferentes Plataformas:** Expandir a aplicabilidade das técnicas desenvolvidas para plataformas móveis e de baixa potência pode ampliar significativamente o alcance e a utilidade das soluções geradas.
- **Integração com Sistemas de Inteligência Artificial:** Utilizar IA para aprimorar a geração procedural e a adaptação dos ambientes pode resultar em sistemas ainda mais dinâmicos e adaptativos.

Em conclusão, este projeto demonstrou que o desenvolvimento cuidadoso de técnicas procedural de geração de biomas pode resultar em avanços substanciais na forma como os ambientes digitais são construídos e experimentados.

Referências Bibliográficas

Chang, M.; Oh, J. W.; Chang, D. S. & Park, S. C. (2017). Interactive marching cubes algorithm for intraoral scanners. *International Journal of Advanced Manufacturing Technology*, 89(5-8):2053 – 2062. Cited by: 2.

Eberly, D. H. (2014). *GPGPU programming for games and science*. Cited by: 10.

Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. Em *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, p. 163–169, New York, NY, USA. Association for Computing Machinery.

Musgrave, F. K.; Kolb, C. E. & Mace, R. S. (1989). The synthesis and rendering of eroded fractal terrains. p. 41 – 50. Cited by: 284.

Parish, Y. I. H. & Müller, P. (2001). Procedural modeling of cities. Em *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, p. 301–308, New York, NY, USA. Association for Computing Machinery.

Pereira, L. (2023). Geração de ambientes complexos com algoritmos procedurais executados em gpu.