



University of Minho
School of Engineering
Master's degree in Software Engineering

Visualização em Tempo Real

2023/2024

Terrain Generation - Trabalho Prático

Grupo nº 3

João Braga(PG53951) Rafael Correia(PG54162) Robert Szabo(PG54194)

June 20, 2024

VTR

Summary

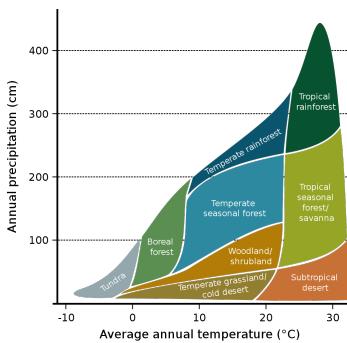
1	Introdução	1
2	Biomas	2
2.1	Biome Map	2
3	Terreno	4
3.1	Introdução	4
3.2	Processo de Geração de Terreno	4
3.2.1	Input da Função	4
3.2.2	Configuração de Parâmetros Específicos do Bioma	4
3.2.3	Parâmetros Independentes do Bioma	5
3.3	Função fbm	5
3.4	Definição de Biomas	5
3.5	Tesselação	6
3.6	Recalcular das Normais	7
3.7	Terreno Infinito	7
3.8	Funcionalidades extra	8
3.8.1	Água	8
3.8.2	Shadow Mapping	9
3.8.3	Novos padrões	10
4	Melhorias futuras	11
5	Comparação	12
6	Conclusão	13

1 Introdução

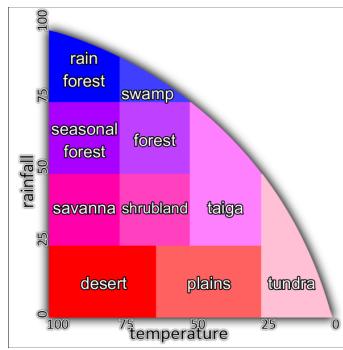
Para o trabalho de VTR, decidimos escolher o tema da geração de terreno. Para o projeto, considerámos terreno não planetário, ou seja, terreno "normal". O projeto baseia-se na utilização de ruídos. Para a escolha dos ruídos, baseámo-nos num artigo que analisa a qualidade versus a rapidez. Optámos por utilizar o Perlin para a geração de biomas, pois é rápido e suficientemente bom, e o Simplex, devido aos seus resultados superiores.

2 Biomas

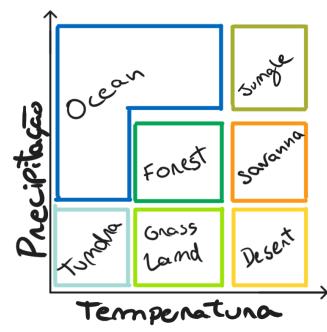
Desde o início do projeto, consideramos a ideia de criar um terreno extensivo e aplicar diferentes parâmetros de elevação e geração de terreno para exprimir características de terreno baseadas em biomas reais. Para isto, utilizamos o conceito de Biome Map, com inspiração do gerador de mundos de Minecraft, numa forma mais simplificada. Procuramos explorar o conceito, com recurso a várias fontes e decidimos retirar inspiração do modelo de classificação de biomas de Whittaker, em que a classificação depende da precipitação e da temperatura. Uma versão anterior do Minecraft utiliza um modelo semelhante para gerar o seu biome map. Note-se que ambos os modelos apresentam regiões de valores que não criam correspondência a um bioma. Com esta informação, decidimos gerar um modelo mais simplificado para gerar os nossos biomas e em caso de não correspondência de valores, geramos um oceano.



(a) Tipos de Biomas de Whittaker



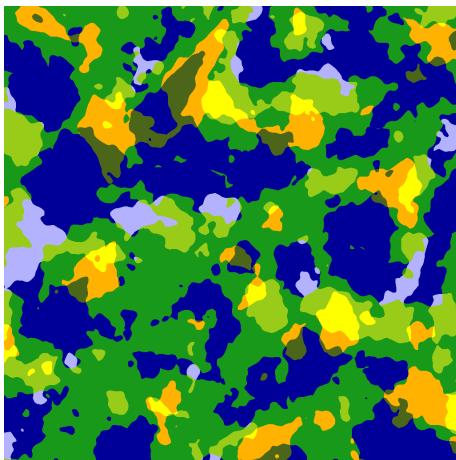
(b) Tipos de biomas de uma versão do Minecraft



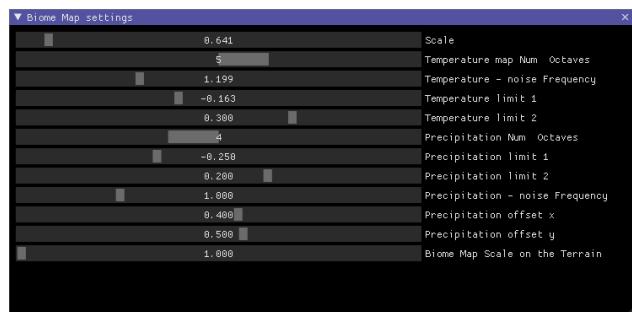
(c) Biomas implementados

2.1 Biome Map

O Biome Map implementado consiste na sobreposição de dois Perlin Noise maps, sendo que um representa a precipitação e o outro a temperatura do bioma. Ambos os mapas definem limites de valores, que representam a transição entre estado do mapa, isto é, no mapa de temperatura existem dois limites de valores que separam o *quente* do *moderado* e do *moderado* do *frio* e no mapa de precipitação existem dois limites de valores que separam o *molhado* do *moderado* e do *moderado* do *seco*. Com estes limites, diferentes combinações geram biomas diferentes, com casos em que as combinações de precipitação e temperatura que não fazem sentido geram oceano. É permitido alterar vários parâmetros de geração do biome map como os limites para experimentação



(a) Mapa gerado



(b) Características do biome map

Com o mapa de biomas gerado, agora passado para o processo de geração de terreno como uma textura, é utilizado para geração do terreno segundo diferentes características definidas para cada bioma.

3 Terreno

3.1 Introdução

Para gerar o terreno, começámos com uma grelha quadrangular, onde para cada ponto geramos uma altura definida por funções de ruído. A função que utilizamos é chamada de fbm (Fractional Brownian Motion), e será explicada posteriormente.

3.2 Processo de Geração de Terreno

3.2.1 Input da Função

Recebemos como input as coordenadas x e z do ponto na grelha e um vetor que descreve o bioma. Estas coordenadas são em worldSpace pois necessitamos que esta função dê um resultado consistente para a posição respetiva do mundo.

3.2.2 Configuração de Parâmetros Específicos do Bioma

Configuramos uma série de parâmetros específicos do bioma através de uma função que ajusta essas configurações dependendo do bioma que recebe. Esses parâmetros são os seguintes:

- **Amplitude:** Determina a altura máxima das variações no terreno. Uma maior amplitude resulta em maiores diferenças de elevação, enquanto uma menor amplitude gera um terreno mais plano.
- **Frequência:** Controla o número de variações no terreno por unidade de área.
- **BaseHeight:** Define a altura mínima do terreno.
- **Escala:** Ajusta a dimensão dos padrões de ruído aplicados ao terreno. Uma escala maior cria variações amplas e suaves, enquanto uma escala menor resulta em detalhes mais pequenos.
- **Persistência:** Determina como a amplitude do ruído muda em cada iteração. Alta

persistência mantém variações menores significativas.

- **Lacunaridade:** Define a mudança na frequência em cada iteração. Alta lacunaridade aumenta rapidamente a frequência das variações, criando um efeito que parece uma textura, enquanto baixa lacunaridade mantém o terreno com uma frequência mais constante.

3.2.3 Parâmetros Independentes do Bioma

Além dos parâmetros específicos do bioma, temos parâmetros que afetam o terreno de forma independente do bioma, como:

- **Altura da Água:** Nível da água.
- **Número de Octaves:** Quantidade de iterações no loop que constrói o ruído. Mais octaves resultam em um ruído mais detalhado.

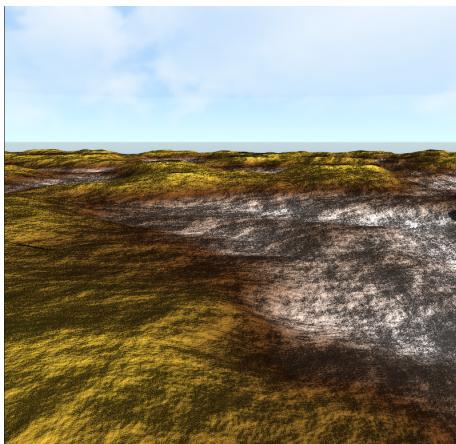
3.3 Função ffbm

A função ffbm utiliza um loop de várias iterações. Em cada iteração, calculamos amostras de ruído para as coordenadas ajustadas pela escala e frequência, somando esses valores ao ruído acumulado. Durante o loop, ajustamos a amplitude e a frequência de acordo com a persistência e a lacunaridade. No final, combinamos o valor acumulado do ruído com a altura base e aplicamos um fator de ajuste para obter a altura final do terreno.

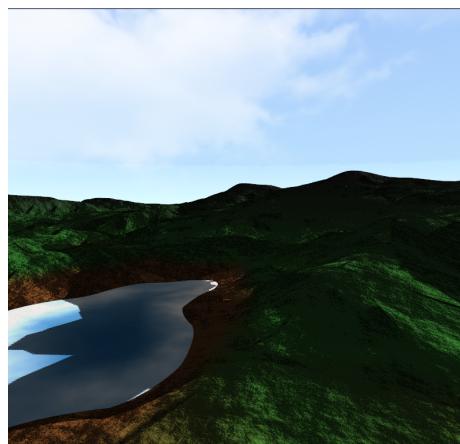
Esta abordagem permite criar paisagens detalhadas e variadas, simulando diferentes tipos de terrenos e biomas. Também temos a possibilidade de ter outros padrões a partir de domain-warping.

3.4 Definição de Biomas

Para definir os biomas, utilizamos sliders para ajustar os parâmetros e selecionámos manualmente alguns padrões que considerámos esteticamente agradáveis. Esses padrões foram "guardados" na função que define os parâmetros específicos de cada bioma. As cores do biomas são definidas noutra função dentro do fragment shader da geração de terreno. Temos os biomas Tundra, grassland, desert, jungle, forest e savanna. Este biomas precisam de bastante mais para serem considerados o bioma cujo lhe foi atribuido, mas é mais para podermos ter terrenos diferentes.



(a) Bioma da Savanna

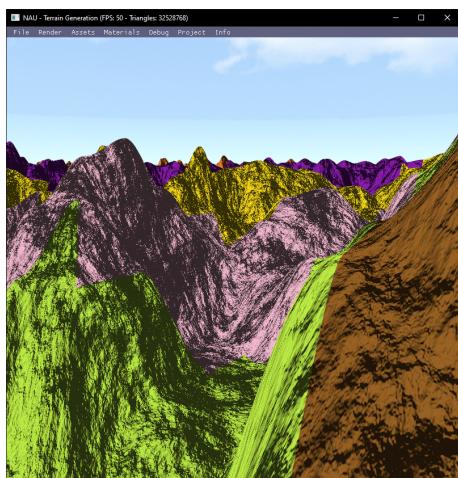


(b) Bioma da selva

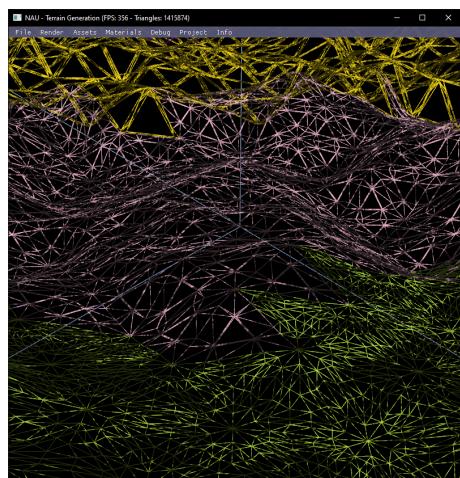
3.5 Tesselação

Um problema que o nosso projeto tinha é que queríamos ocupar o ecrã com o nosso terreno, para isto precisavamos de uma grelha enorme. Mas se tivermos uma grelha enorme e com muito detalhe, teríamos demasiados triângulos e a performance não seria ideal. Isso levou-nos a implementar a nossa tesselação utilizando PN Triangles, conforme descrito nos slides disponíveis.

No tessellation control shader, ajustamos dinamicamente o nível de detalhe com base na distância dos triângulos ao clipSpace. Isso significa que áreas próximas à camera têm uma mais tesselação, garantindo detalhes nítidos, enquanto áreas distantes têm uma tesselação reduzida para optimizar o desempenho. A imagem abaixo ilustra como diferentes níveis de tesselação são aplicados ao longo da cena, adaptando-se conforme necessário:



(a) Tesselação níveis



(b) Tesselação triangulos

Cada cor na imagem representa um nível diferente de tesselação. Escolhemos cores distintas para garantir mais que 3 níveis de detalhes (r,g,b). Dependendo do tamanho da grelha e

do espaçamento dos triângulos, ajustamos o tamanho de cada nível para cobrir uma área adequada, especialmente em casos onde os triângulos são maiores. Se a grelha original tivesse mais triângulos, poderíamos reservar os níveis mais altos de tesselação para os triângulos mais próximos à câmara.

Incorporamos também a capacidade de ajustar globalmente os níveis de tesselação, o que permite melhorar o desempenho em computadores menos potentes ou em situações onde uma complexidade gráfica reduzida é suficiente. Como por exemplo, aqui ajustamos a tesselação para 20% do nível original para poder claramente observar a redução na quantidade de triângulos gerados à medida que a distância à camera aumenta.

3.6 Recalcular das Normais

Um outro problema que enfrentamos foi garantir que as normais fossem corretamente calculadas mesmo com a tesselação aumentada, pois as normais da grelha plana original não eram adequadas. Para resolver isso e garantir maior detalhe, optamos por calcular as normais para cada pixel do terreno. Para isso, criamos quatro pontos adicionais à volta de cada ponto original: frente, trás, direita e esquerda. Supondo que o ponto original seja $(0, \text{altura}, 0)$, os componentes x e z desses pontos foram ajustados por uma variável OFFSET. De seguida, calculamos quatro vetores: um para cada ponto adicional para o ponto original. Finalmente, utilizando o cross product entre os vetores "frente" e "direita", assim como entre "esquerda" e "trás", somamos os resultados e normalizamos o vetor resultante. Tendo assim, uma média dos cross products, garantindo uma maior precisão no cálculo das normais do terreno.

Este cálculo poderia ser feito só com um triângulo, mas não teria resultados agradáveis em comparação ao que fizemos por causa da quantidade de detalhe que colocamos por defeito.

3.7 Terreno Infinito

Uma funcionalidade que desejávamos ter no nosso terreno era torná-lo infinito. Para isso a ideia principal era, mover a grelha de uma maneira a ser sempre visível para a câmera e recalcular a altura a partir da nossa função fhm que dada uma coordenada no espaço global, devolve sempre o mesmo valor. Na primeira tentativa, geramos uma grelha que fazia translate para acompanhar a posição da câmera e rodava na direção para onde a câmera estava apontada. No entanto, encontramos um problema: ao gerar os pontos do terreno, surgiam erros de aproximação que causavam um efeito de movimento inconsistente do terreno enquanto a câmera se movia, parecia que o terreno tinha vida. Para melhorar isso, removemos a rotação, que não era essencial, mas permitia que o terreno cobrisse mais ecrã (pois podíamos fazer um translate mais generoso a meter a camera numa ponta).

A nossa solução final foi mais simples: dividimos a posição da câmera por um valor arbitrário

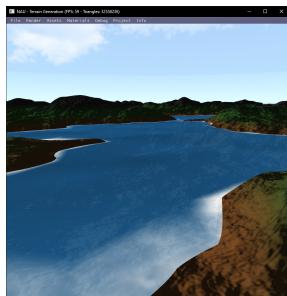
e arredondamos para o valor inteiro mais próximo. De seguida, fizemos translate do terreno com esse valor multiplicado pelo valor arbitrário. Isso permitiu-nos ajustar essa variável para evitar pequenos valores que pudessem causar erros de aproximação no movimento do terreno, além de evitar que o fim da grelha fosse visível na tela.

3.8 Funcionalidades extra

3.8.1 Água

Para tornar o terreno mais realista, decidimos adicionar água. Como mencionado anteriormente, a água influencia a aparência do terreno. Embora não tenhamos aplicado erosão hidráulica, a água altera as cores do terreno. Para criar o efeito de água, utilizamos duas texturas para normal mapping que se movem em direções opostas. Além disso, uma dessas texturas foi usada para aplicar um displacement mapping, dando a impressão de movimento na água.

Outro detalhe que implementamos foi a espuma nas bordas nas costas. Para isso, para cada ponto na água, calculamos se ele está abaixo ou acima do terreno. Se estiver abaixo, aplicamos uma textura de espuma sobre a água; se estiver acima, a superfície permanece como água normal. Este cálculo tem um pormenor que é um pequeno offset para aparecer espuma só nas bordas. Nas seguintes imagens podemos ver a água em si e a água debaixo do terreno:

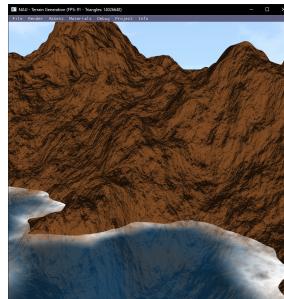


(a) Água na superfície

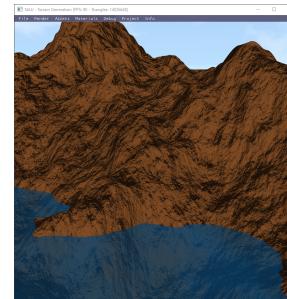


(b) Água de baixo do terreno

Como o terreno não é suposto ser visível por dentro, não faz mal a água ser só espuma, mas mesmo assim temos a hipótese de retirar a espuma caso não seja de agrado para o utilizador:



(a) Espuma

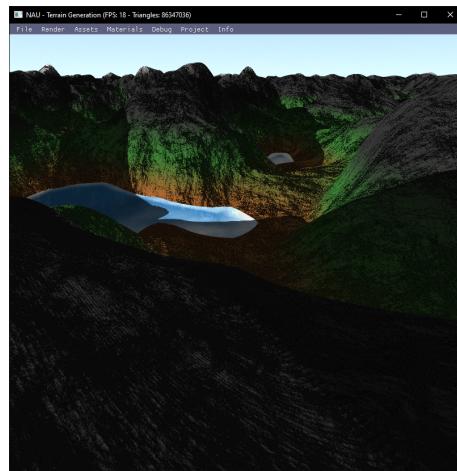


(b) Sem espuma

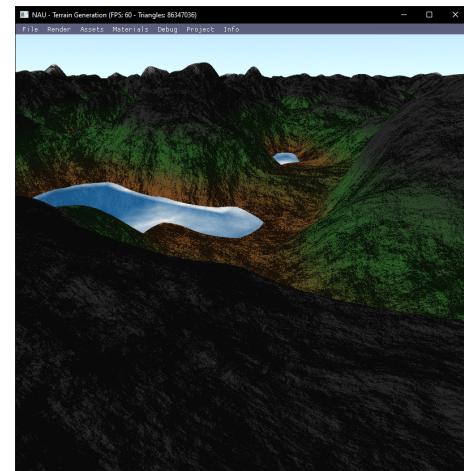
3.8.2 Shadow Mapping

Outra feature que foi implementada foi shadow mapping, mais especificamente cascade shadow mapping. Para isso tivemos que "dividir" a camera, e optamos por fazer 4 divisões. Criamos 4 passos na nossa pipeline que calculam a profundidade a cada divisão. Para cada divisão, é gerada um shadow map, por outras palavras, texturas que guardam a profundidade dos objetos a partir da perspectiva da luz.

A nossa implementação é custosa na performance porque temos que reconstruir o terreno outra vez para calcular as sombras (isto é necessário porque o modelo que nos estamos a basear é uma grelha plana) e se isso já não fosse custoso o suficiente, também temos que calcular as sombras com triângulos tesselados, porque se não "tesselassemos as sombras" iríamos ter sombras não representativas do terreno. Por estas razões decidimos ter uma opção para ligar e desligar as sombras para poupar cálculos. O resultado da nossa implementação pode-se ver nas seguintes imagens:



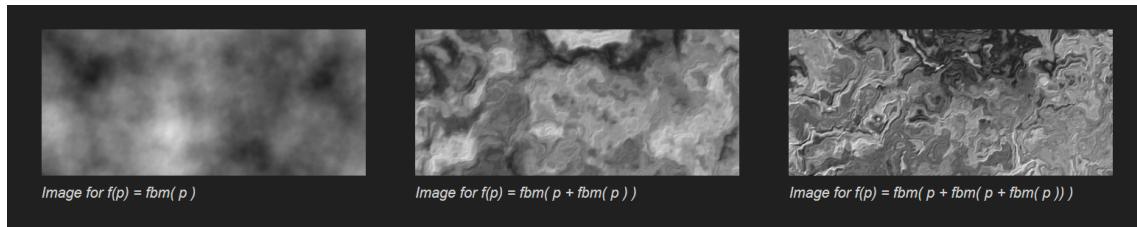
(a) Sombra



(b) Sem sombra

3.8.3 Novos padrões

Os padrões que criámos para o terreno são inspirados no ruído simplex. Para explorar novos padrões, utilizámos uma técnica denominada de 'domain warping'.



Esta abordagem permite-nos criar padrões mais variados e interessantes. Por vezes, os resultados até se assemelham a terrenos vulcânicos. No entanto, estes padrões exigem mais recursos do que o método 'fbm' comum. Na nossa implementação, optámos por usar o 'fbm' puro como base, mas mantivemos a possibilidade de mudar para estes novos padrões se desejarmos, mas a ideia de usar estes padrões para certos biomas apenas seria algo que a nossa equipa desejaría explorar.

Para uma descrição mais aprofundada, pode consultar o artigo em que nos baseámos aqui.

4 Melhorias futuras

Como acontece em qualquer projeto, tivemos que fazer alguns compromissos. Ainda assim, acreditamos que o resultado é positivo.

No futuro, gostaríamos de resolver alguns problemas identificados no nosso programa, especialmente no que diz respeito à interação entre a água e a espuma. Embora a água não tenha sido o foco principal do nosso trabalho, teríamos preferido implementar algo mais complexo. Reconhecemos que existem defeitos a serem corrigidos, principalmente a interação da água ao mover a camera, que gera um efeito visual da animação "cortar" e a falta de tesselação na grelha da água, que resulta numa espuma "triangular". Acreditamos que a tesselação na água poderia solucionar este problema.

Uma funcionalidade que gostaríamos de adicionar é a inclusão de objetos pelo mundo, como árvores e torres, para proporcionar uma noção de escala e tornar o ambiente mais interessante e também diferentes tipos de ruído e padrões.

Finalmente, gostaríamos de adicionar transições mais suaves entre os diferentes biomas.

5 Comparação

Neste trabalho, fomos inspirados por diversas fontes. Em primeiro, o videojogo Minecraft e a sua geração de biomas serviram-nos como referência. Queríamos criar um gerador de terreno com múltiplos biomas que fosse "infinito" e apresentasse variação. Também exploramos geradores de terreno como o que foi discutido numa conferência da GDC sobre o No Man's Sky, que mencionou como é que eles gerão o seu terreno, eles denominam o seu ruído de "Uber noise", que ao contrário da nossa função usa múltiplas camadas de noise diferente para ter padrões interessantes, algo que gostaríamos de implementar no futuro. Como podemos ver nas seguintes imagens:



(a) perlin



(b) Uber

Outro exemplo que seguimos foi um conjunto de vídeos do Sebastian Lague que o processo de ele tentar gerar um terreno. Tomámos algumas inspirações na implementação dele e achamos que conseguíramos construir por cima do trabalho dele.

6 Conclusão

Com este trabalho queríamos criar um gerador de terreno com biomas variados, inspiramo-nos em várias fontes, como Minecraft, No Man's Sky. Implementamos uma função de fbm para a geração de ruído para ter alturas diferenciadas, temos vários parâmetros para poder experimentar e criar biomas, aplicamos conhecimentos das aulas como tesselação, shadow maps e normal Maps e achamos que temos um trabalho com bastante conteúdo.

No entanto, reconhecemos que há espaço para melhorias. No futuro gostaríamos de as resolver, também sabemos que há sempre mais funcionalidades que podíamos adicionar, como experimentar mais combinações de ruído diferentes.

Concluindo, estamos satisfeitos com o que alcançamos até agora, mas vemos muitas oportunidades para expandir.