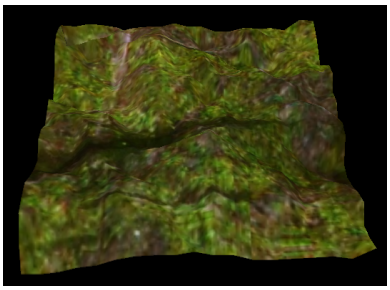
Geração de Terrenos

Mayara Marques da Rosa *

[1]Universidade Federal Rural do Rio de Janeiro, Departamento de Ciência da Computação, Brasil



RESUMO

Este relatório tem o objetivo de evidenciar as etapas de desenvolvimento do trabalho proposto na disciplina de computação gráfica no período 2018-1. O trabalho em questão se trata da geração de terrenos utlizando OpenGl e bibliotecas relacionadas. Este relatório traz conceitos ,abordagens e ferramentas utilizadas durante o trabalho bem como os resultados obtidos.

Keywords: Terrenos, malha poligonal.

1 INTRODUÇÃO

A geração de terrenos aparece em diversos contextos como jogos, filmes e sistemas de realidade virutal, motivada pelo desejo de simular de forma cada vez mais natural superfícies como relevos e montanhas[9]. Por meio de conceitos de matemática e de computação gráfica a tarefa de representar terrenos se torna possível.

2 FERRAMENTAS

Para o desenvolvimento deste trabalho foram utilizados:

- Linguagem C++
- OpenGl
- Glut
- Glu
- SOIL

3 REPRESENTAÇÃO DO TERRENO

Segundo [7], uma malha polígonal é uma coleção de vértices, arestas e faces semelhante a uma grade desestrutura que define a forma de um objeto tridimensional. Para a representação do terreno foi construída uma malha polígonal. Todos os vértices dessa malha foram dispostos em um espaçamento uniforme e conectados de maneira a formar triângulos.

*e-mail: mmrosatab@hotmail.com

3.1 Triangle strips

Um triangle strip permite criar uma sequência de triângulos conectados. Após três vértices serem desenhados na tela e conectados por arestas, cada novo vértice inserido será conectado aos dois vértices anteriores [8].

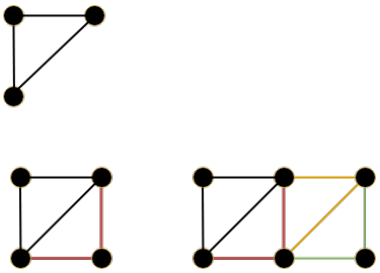


Figura 1: Exemplo ligação dos vértices

No OpenGL este conceito pode ser aplicado através da primitiva *GL_TRIANGLE_STRIP*.

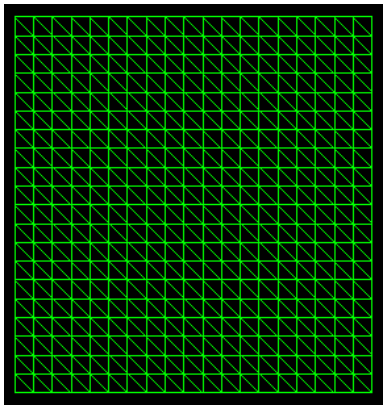


Figura 2: Malha triangular

4 OBSERVAÇÃO

Após a construção da malha polígonal, houve-se a necessidade de mudar a posição da câmera/observador. A função *GluLookAt* define

posição da câmera e direção em que a câmera está "observando"a cena.

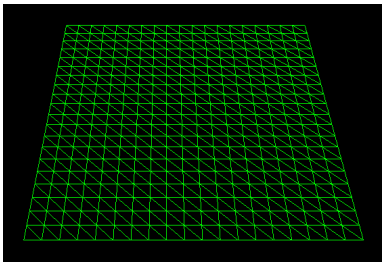


Figura 3: Malha triangular

5 Ruído

5.1 Ruído

O ruído diz respeito a padrões aleatórios que interferem a qualidade de sinais de áudio, vídeo e imagem[6]. Em computação gráfica o ruído é empregado em diversas aplicações para fins diferentes. Neste trabalho serão usadas duas formas de geração de ruído para definir os pontos de altura em Z na malha que representa o terreno.

5.1.1 Pontos de altura em Z

Para dar um efeito de rugosidade a malha, de forma pseudo-aleatória números em um intervalo entre -10 e 10 foram gerados e atribuídos ao pontos em Z na malha.

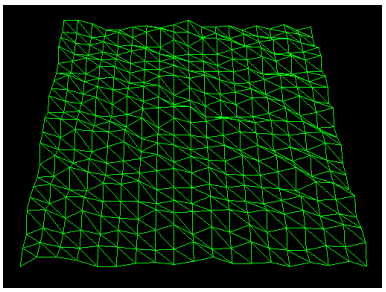


Figura 4: Pontos de altura em Z definidos entre -10 e 10

5.1.2 Perlin Noise

Com o intuito de permitir que o terreno gerado por meio da malha polígonal tenha o efeito mais realista, foi empregado a técnica desenvolvida por Ken Perlin, o Perlin Noise. O Perlin noise possibilita a geração de números aleatórios suaves[1].

O Perlin Noise possui diversas aplicações tais como geração de terrenos, renderização de nuvens, texturas, animações e etc[4]. Foi originalmete construído nos anos 80 para o filme Tron com o objetivo de obter texturas contínuas para objetos tridimensionas[5]. Desde então vem sendo utilizada em diversas aplicações e servindo como referência para vários trabalhos.

Dado um ponto (x,y), Perlin noise retorna um valor Z. O valor retornado está entre 0 e 1. Na abordagem da etapa anterior os números para altura em Z eram gerados de forma aleatória entre o intervalo de -10 a 10 e não possuiam relação nenhuma uns com os outros. O algoritmo de Perlin faz com que cada ponto escolhido em torno de um ponto Z seja semelhante a ele[1].

5.1.3 Passos para Perlin noise:

- Definição da grade
- Produto escalar entre gradiente e a distância de vetores
- Interpolação linear dos valores do produto escalar

5.1.4 Demonstração:

- Obtendo ponto e vizinhança:

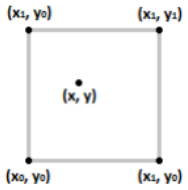


Figura 5: Obtendo ponto e vizinhança. [6]

- Obter vetor de distância entre ponto x,y e seus vizinhos na grade:

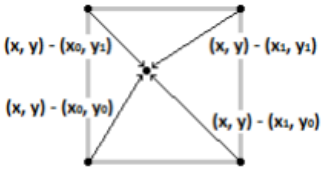


Figura 6: Calculo do vetor de distância entre ponto x,y e seus vizinhos[6]

- Calcular o produto escalar entre o gradiente neste no ponto e o vetor de distância obtido:

$$v1 = gradiente(x_0, y_0) \cdot ((x, y) - (x_0, y_0))$$
$$v2 = gradiente(x_1, y_0) \cdot ((x, y) - (x_1, y_0))$$
$$v3 = gradiente(x_0, y_1) \cdot ((x, y) - (x_0, y_1))$$
$$v4 = gradiente(x_1, y_1) \cdot ((x, y) - (x_1, y_1))$$

Figura 7: Produto escalar entre gradiente e vetor de distância[6]

6 Textura

O uso de textura consiste em basicamente "aplicar"uma imagem sobre as faces de um objeto 3D[2]. O mapeamento da textura possui uma representação que varia de zero até um, tanto no eixo x quanto y. Desse modo para aplicar a imagem a malha foi feita uma correspondência entre os pontos da malha polígonal e a textura. Cada valor de ponto x,y,z da malha foi multiplicado por 0,001 para que os pontos da malha ficassem mapeados em um intervalo de 0 a 1.

O carregamento da imagem de textura foi feito através da biblioteca SOIL.

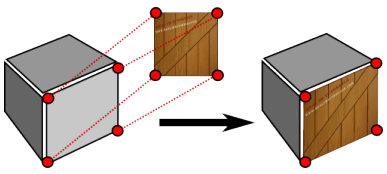


Figura 8: Conceito aplicação textura OpenGl[3].

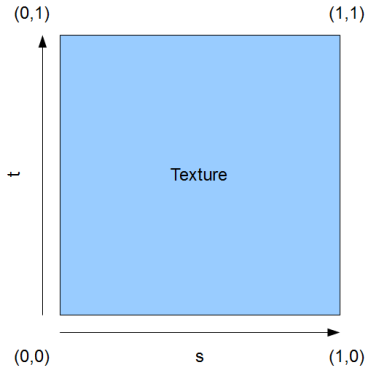


Figura 9: Coordenadas da textura[3].

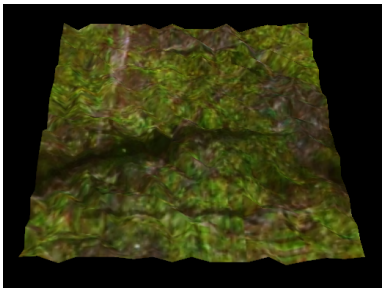


Figura 10: Aplicação de textura na malha.

7 RESULTADOS

Segue abaixos resultados obtidos:

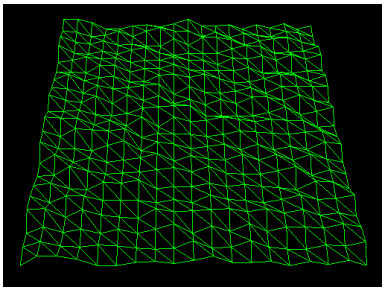


Figura 11: Antes da aplicação textura

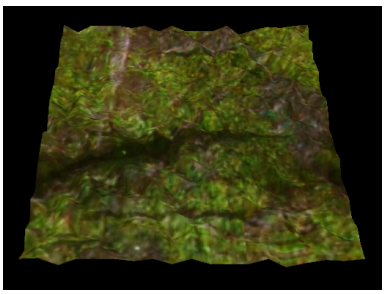


Figura 12: Depois aplicação textura

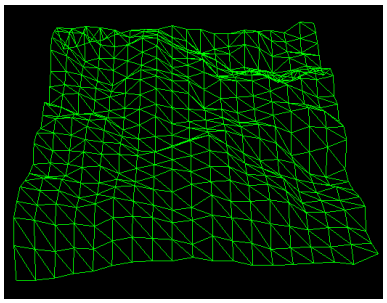


Figura 13: Perlin noise - Antes da aplicação textura

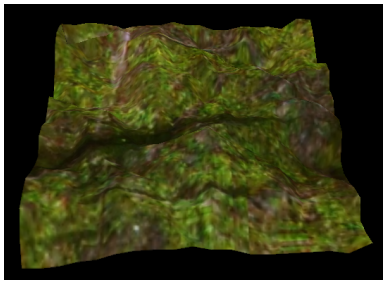


Figura 14: Perlin noise - Depois da aplicação textura

REFERÊNCIAS

[1] 3d terrain generation with perlin noise in processing. https://www.youtube.com/watch?v=IKB1hWWedMk. Accessed: 2011-07-01.

[2] Mapeamento de texturas. https://www.inf.pucrs.br/ pinho/CG/Aulas/OpenGL/Texturas/MapTextures.html. Accessed: 2011-07-11.

[3] Mapeamento de texturas. http://www.real3dtutorials.com/tut00005.php. Accessed: 2011-07-11.

[4] Perlin noise and its application. https://hendhyhutomoere.wordpress.com/2013/11/14/per noise-and-its-application/. Accessed: 2011-07-01.

[5] Tron. https://mrl.nyu.edu/ perlin/tron/. Accessed: 2011-07-11.

[6] G. T. da Cunha Coelho. Geração procedimental de ambientes para jogos eletrônicos, October 2002.

[7] Z. L. Faxin Yu, Hao Luo and P. Wang. *Three-Dimensional Model Analysis and Processing*. Springer, first edition, 2010.

[8] J. J. McConnell. *Computer Graphics: Theory Into Practice*. Manning, first edition, 2006.

[9] R. L. Saundersh. Terrainosaurus - realistic terrain synthesis using genetic algorithms. 1(2):1–19, Dec. 2006.