Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Ciência da Computação

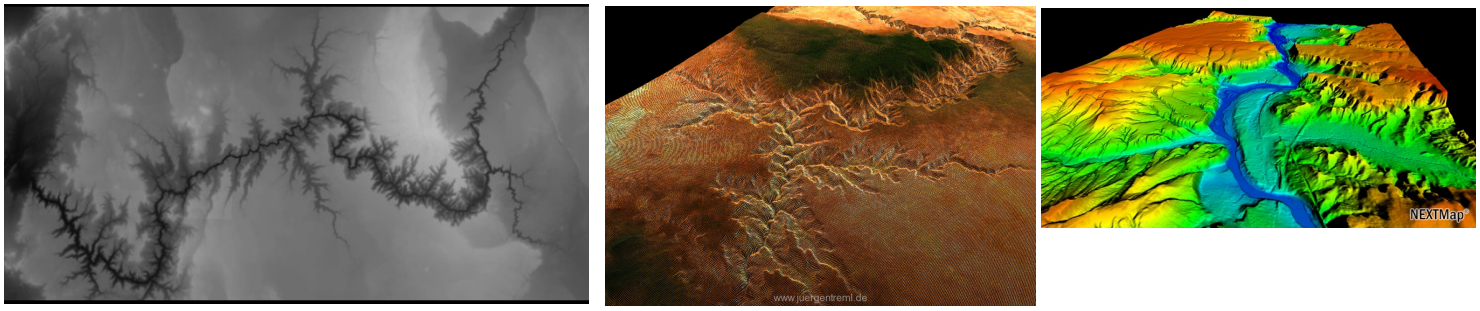
MATA65 - Computação Gráfica Período: 2014-1 Data: 10/06/2014
Prof. Antonio L. Apolinário Junior

Segundo Trabalho

Motivação:

O uso de modelos de terreno sempre foi vital para várias atividades humanas. Atualmente com o uso de imagens de satélite e o auxilio da Ciência da Computação sua confecção tornou-se mais simples e seu acesso muito mais fácil. Um dos Modelos Digitais de Terreno (MDT) [1] mais simples é o *Digital Elevation Model* (*DEM*), ou Modelo Digital de Elevação. Um *DEM* nada mais é que uma imagem digital onde cada pixel representa uma altura relacionada ao terreno. A figura 1 mostra um *DEM*, como uma imagem digital e sua representação como a superfície de um terreno visualizada em 3D.

Figura 1: Modelo Digital de Terreno do *Grand Canyon*: representação como imagem e como mapa de elevações no espaço 3D com textura e cores baseadas na elevação.



Jogos eletrônicos também vem utilizando de modelos digitais de terreno para criar cenários externos com maior grau de realismo. Para tanto esses modelos de terreno tem que estar em uma escala correta para que os personagens do jogo possam vê-lo de forma proporcional. Além disso, os objetos que se locomovem sobre esse terreno devem ter a sensação de estarem caminhando no local, ou seja, as irregularidades do terreno devem permitir ou impedir seu avanço. Na figura 2 podemos ver 2 exemplos de paisagens criadas para jogos onde o terreno se destaca como parte do importante do cenário.

Figura 2: Exemplos de cenários onde o terreno se destaca em jogos eletrônicos. Da esquerda para a direita: Lord of The Rings, Witcher 2 e Star Wars - The Old Republic.

Objetivo do trabalho:

Construir uma aplicação em WebGL que permita visualizar qualquer um dos modelos digitais de terreno do tipo DEM, fornecidos pelo professor.

Sua aplicação deve ser capaz de:

1. Carregar um DEM como uma imagem e montar sua representação tridimensional, como uma superfície (valor 1,5)
2. Configurar uma câmera que permita a visualização da superfície do terreno de um ponto de vista sobre a superfície, de modo que a visão seja igual a da imagem digital inicial. (valor 1,0)
3. A visualização do terreno deve ser feita no formato *wireframe* ou utilizando uma escala de cores semelhante a de mapas topográficos, onde a altura em um ponto do terreno determinar a sua coloração (Figura 1). (valor 2,0)
4. Configurar uma segunda câmera que permita o caminho sobre o terreno. A sensação que se deve ter é de um personagem caminhando sobre o terreno. A escala do do movimento do observador deve ser levada em conta nesse processo. O usuário deve ser capaz de controlar a câmera pelo teclado e/ou mouse e poder se movimentar de acordo com o terreno. (valor 2,0)
5. O usuário não pode “entrar" dentro do terreno. Da mesma forma o usuário deverá parar seu caminhar caso encontre um obstáculo no terreno. Caso o obstáculo seja menor que um certo “degrau” ele passa sobre o obstáculo. Obstáculos maiores que um “degrau" porém menores que um “pulo" podem ser ultrapassados com um “salto" (comando especial). Obstáculos maiores que um pulo devem impedir o personagem de prosseguir mesmo que ele tente “saltar”. (valor 1,5)
6. Configurar uma terceira câmera que possa voar pelo terreno, ou seja, que não fique presa ao chão. Essa câmera deve fazer movimentos semelhantes aos de um avião, ou seja, *tilt*, *pan* e *roll*. A escala do movimento dessa câmera deve ser maior que a câmera do personagem. (valor 2,0).

A Implementação:

A implementação deve ser desenvolvida em linguagem Javascript/WebGL/HTML5 [3] [4].

Os trabalhos deverão ser desenvolvidos individualmente ou em duplas. Os códigos gerados deve ser comentados e legíveis. Acompanhando os códigos fonte um breve relatório técnico (**em formato pdf** com 2 ou 3 paginas) deve ser entregue, descrevendo brevemente o que foi implementado e como a aplicação deve ser utilizada.

A interface para o controle das operações propostas deve ser feita em HTML5/ Javascript.

A Entrega:

O trabalho deverá ser submetido via *Moodle*, respeitando a data e hora limite para entrega. **Em caso de atraso, será aplicado um fator de penalização de 1,0 ponto por dia de atraso**. Qualquer problema de arquivos corrompidos ou similar o trabalho será considerado não entregue. Portanto, verifique bem o que for entregar.

Os arquivos devem ser enviados seguindo o seguinte padrão: arquivo compactado (zip, rar, tgz ou gzip apenas) contendo um diretório com o nome do(s) aluno(s) e seu arquivos. Arquivos fora desse padrão sofrerão penalização de 0,5 pontos na nota final.

Códigos com caminhos absolutos ou qualquer outra pendência que impeça a execução não serão avaliados.

A cooperação entre alunos e grupos é considerada salutar. No entanto, trabalhos com alto grau de similaridade serão tratados como “plágio”, o que resultará em avaliação ***zero*** **para todos os envolvidos**.

Qualquer dúvida adicional, evite problemas: não presuma nada, procure o professor para esclarecimentos.

Referencias Bibliográficas:

[1] Polack, T., **Focus On 3D Terrain Programming** (Focus on Game Development), Course Technology PTR; 1st edition, 2002.

[2] Hughes , J.F., van Dam, A., McGuire, M., Sklar, D.F., Foley, J.D., Feiner, S.K., Akeley, K., **Computer Graphics: Principles and Practice**, Addison-Wesley Professional, 3rd Edition, 2013.

[3] Matsuda,K., Lea, R., **WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL**, Addison-Wesley Professional, 1st edition, 2013.

[4] Parisi, T., **WebGL: Up and Running**, O'Reilly Media, 1st edition, 2012.