Computação gráfica
Trabalho 2

Lucas Kutner Amin

[1]DAINF – Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brazil

lucasamin@alunos.utfpr.edu.br

1. Introdução

A imagem é lida utilizando a biblioteca opencv para python, extraindo a matriz que a representa, assim tornando possível extrair vértices em três dimensões seguindo a função (x, y, f(x,y)). Para gerar a malha triangular, o buffer é preenchido com trios de pontos, indicando a posição dos vértices dos triangulos então renderizado na função display com a flag GL_TRIANGLES. Para mudar a visualizacao para malha de pontos, é preciso apenas modificar a flag GL_TRIANGLES usada na função display para GL_POINTS ou GL_LINES já que a array de triangulos possui todas as linhas e pontos usados.

2. Desenvolvimento

2.1. Estrutura de Dados para Terreno

O terreno é carregado a partir de uma imagem, representada por uma matriz. Com isso, cada vértice é adicionado em uma estrutura de dados chamada Structure, com linhas, vértices e polígonos.

A classe structure é utilizada para armazenar cada vértice, linha e polígono registrados na malha construída. Os vértices são guardados em um vetor, com suas posições, as linhas são guardadas com os pares de vértices e os triangulos são guardados em um vetor, com os trios dos segmentos de retas.

2.2. Método para computação das normais

O cálculo das normais utilizadas foram divididas em duas partes, primeiro, as normais dos triangulos e depois a normal dos vértices.

2.2.1. Cálculo das normais dos triangulos

Para cada triângulo, extrai-se dois dos segmentos de reta e realiza-se um produto vetorial. Por ultimo, o vetor normal calculado pelo produto vetorial é normalizado e então colocado no vetor que vai para o shader.

2.2.2. Cálculo das normais dos vértices

Para cada triângulo, extrai-se os três dos segmentos de reta e realiza-se um produto vetorial entre cada um dos segmentos com a normal do triangulo, calculada anteriormente. Por ultimo, os vetores normais são normalizados e então colocados em um vetor final, carregado na estrutura que representa a malha triangular e nos buffers.

2.3. Escolhas para gerar uma boa visualização

A cor de cada posição da malha foi gerada a partir de sua posição xyz, tornando a malha colorida e facilitando a observação dos triangulos da malha.

- cor da luz - glm.vec3(1, 1, 1)
- cor do objeto - glm.vec3(1, 0.4, 0.25)
- matriz de perspectiva - glm.perspective(glm.radians(110), 16 / 9, 0.1, 100)
- Angulo utilizado em rotações - 3.0
- Fator de escala - 1.09
- Fator de translação - 25
- Tamanho dos pontos utilizados - 2.5
- Cor de fundo utilizada - (0.2, 0.2, 0.2, 0.2)

3. Resultados

Com o cálculo da normal dos triangulos, foi possível carregar o shader e efetivar o uso da luz ambiente, difusa e especular em uma só visualização. Porém, a maior dificuldade encontrada foi a utilização das normais dos vértices, o que ficou fora do resultado esperado e não foi implementado até o final no trabalho.

Além disso, encontrar as modificações para utilizar um shader com os atributos necessários para a utilização de luz ambiente, difusa e specular foi difícil, pois a implementação foi feita em python, diminuindo o número de exemplos disponíveis.

Outra dificuldade encontrada foi efetivar os cálculos necessários para gerar as normais de cada triangulo da malha em tempo aceitável, quando se trata das imagens exemplo maiores como crater e crater2.