Computação Gráfica
Trabalho 1

Descrição

Neste trabalho deverá ser implementada a visualização de malhas triangulares de modelos 3D. A visualização poderá ser por faces ou *wireframe* (mostrando apenas as arestas das faces). Deve ser fornecidos dois modos de visualização poligonal. Também deverão ser implementadas funções de manipulação das malhas usando transformações geométricas. A malha triangular deverá ser lida de um arquivo do tipo *object file* (OBJ).

Arquivo de Entrada

O modelo de entrada deverá esta no formato OBJ que possui extensão .obj[1]. Este tipo de arquivo define a geometria de um objeto, seus vértices e faces, e em alguns casos, outras propriedades, como normais. Um exemplo simples de arquivo é mostrado abaixo.

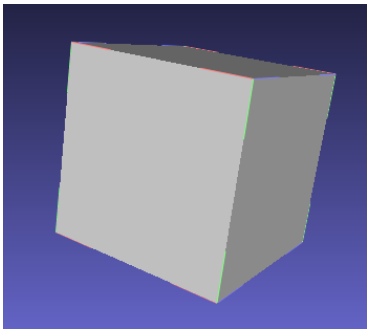
```
# cube.obj

v  0.0  0.0  0.0
v  0.0  0.0  1.0
v  0.0  1.0  0.0
v  0.0  1.0  1.0
v  1.0  0.0  0.0
v  1.0  0.0  1.0
v  1.0  1.0  0.0
v  1.0  1.0  1.0

vn  0.0  0.0  1.0
vn  0.0  0.0 -1.0
vn  0.0  1.0  0.0
vn  0.0 -1.0  0.0
vn  1.0  0.0  0.0
vn -1.0  0.0  0.0

f  1//2  7//2  5//2
f  1//2  3//2  7//2
f  1//6  4//6  3//6
f  1//6  2//6  4//6
f  3//3  8//3  7//3
f  3//3  4//3  8//3
f  5//5  7//5  8//5
f  5//5  8//5  6//5
f  1//4  5//4  6//4
f  1//4  6//4  2//4
f  2//1  6//1  8//1
f  2//1  8//1  4//1
```

[1] http://paulbourke.net/dataformats/obj/
https://people.sc.fsu.edu/~jburkardt/data/obj/obj.html

O arquivo define um cubo da seguinte maneira:

- v indica as coordenadas de um vértice (os valores nas colunas que seguem o indicador);
- vn indica as normais de um vértice;
- f indica uma face que é composta pelas propriedades dos vértices que seguem nas colunas seguintes. No exemplo os vértices possuem coordenadas x, y e z, assim como as normais. As faces são definidas como $i//j$, em que i é um vértice que forma a face e j é a normal relacionada com o vértice i.

O objeto do arquivo é mostrado na figura abaixo.



Transformações geométricas

Dada uma malha triangular, funcionalidades para a aplicação de transformações geométricas (translação, escala e rotação) devem ser disponibilizadas para mudar a visualização.

Implementação e Execução

As implementações devem ser feitas em linguagem C ou C++ e compilar, respectivamente, usando os compiladores gcc e g++. Um arquivo Makefile deve ser disponibilizado para a compilação. A compilação deve ser feita usando o comando *make* em um terminal de comandos do Linux. Alternativamente, poderá ser usada a linguagem Python para a implementação.

Em qualquer caso, o aplicativo deve executar em sistema operacional Linux usando a biblioteca "moderna" do OpenGL, com o uso de shaders para mostrar a visualização e manipular o modelo de terreno por transformações geométricas. Não devem ser usadas construções como glBegin e glEnd das versões antigas do OpenGL.

O programa deve ter o nome "mesh" e receber como argumento o caminho para um arquivo OBJ. A execução deve iniciada em um terminal de comandos do Linux, como no exemplo abaixo.

```
$ ./mesh cube.obj
```

Caso seja implementado em Python, o programa deve ter o nome "mesh.py" e receber o caminho para um arquivo OBJ como argumento. Sendo executado como mostrado abaixo em um terminal de comandos do Linux.

```
$ ./mesh.py cube.obj
```

Após abrir a janela de visualização, as seguintes funcionalidades devem estar disponíveis para que o usuário possa manipular o modelo. Os valores de incremento para as translações, rotações e escalas devem ser escolhidas de forma que as transformações não sejam nem abruptas nem imperceptíveis visualmente. Rotações e escalas devem ser feitas usando o centro do modelo como referência.

Translação

Ao digitar no teclado a letra "t" o programa deve entrar em modo de translação. As seguintes teclas então definem a direção que o modelo deve ser incrementalmente deslocado:

- seta para cima: deslocamento positivo em y;
- seta para baixo: deslocamento negativo em y;
- seta para a direita: deslocamento positivo em x;
- seta para a esquerda: deslocamento negativo em x;
- a: deslocamento positivo em z;
- d: deslocamento negativo em z.

Rotação

Ao digitar no teclado a letra "r" o programa deve entrar em modo de rotação. As seguintes teclas então definem a direção que o modelo deve ser incrementalmente rotacionado:

- seta para cima: rotação positiva em x;
- seta para baixo: rotação negativa em x;
- seta para a direita: rotação positiva em y;
- seta para a esquerda: rotação negativa em y;
- a: rotação positiva em z;
- d: rotação negativa em z.

Escala

Ao digitar no teclado a letra "e" o programa deve entrar em modo de escala. As seguintes teclas então definem a direção que o modelo deve ser incrementalmente escalado:

- seta para cima: fator de escala maior que 1 em y;
- seta para baixo: fator de escala menor que 1 e maior que 0 em y;
- seta para a direita: fator de escala maior que 1 em x;
- seta para a esquerda: fator de escala menor que 1 e maior que 0 em x;
- a: fator de escala maior que 1 em z;
- d: fator de escala menor que 1 e maior que 0 em z.

Visualização

Ao digitar no teclado a letra "v" o programa deve alternar entre as visualizações de faces (polígonos preenchidos) e *wireframe* (apenas arestas dos polígonos).

Entrega

As entregas deverão ser realizadas usando o Moodle da disciplina e não serão aceitas entregas fora do prazo.

Reúna os códigos-fonte e o relatório em um diretório com o número do seu registro acadêmico. Se o seu registro acadêmico for 1234567, o diretório deve ter o nome 1234567. Dentro do diretório deverão estar os códigos-fonte e o arquivo Makefile (se usar C ou C++). Por exemplo, o diretório poderia conter os arquivos listados abaixo (não coloque acentos ou espaços em nomes de arquivos).

```
1234567/mesh.c
1234567/Makefile
```

O arquivo de entrega deve ser um ZIP do diretório. O arquivo ZIP deve inclusive conter o diretório e ser nomeado da mesma maneira. No nosso exemplo: 1234567.zip.