PROJETO 1

MÓDULO BÁSICO (40%)

VISUALIZAÇÃO DE MODELOS

OPENGL

O projeto deve ser realizado utilizando a biblioteca de renderização OpenGL.

CARREGAMENTO DE .OBJ

Exemplo de arquivo .obj:

```
# cubo.obj
# cubo representado por triangulos centrado na origem
# Linhas que iniciam com # sao comentarios
# Autor: Marcelo Walter
# Lista de vertices
v -0.5 0.5 0.5
v -0.5 -0.5 0.5
v 0.5 -0.5 0.5
v 0.5 0.5 0.5
v -0.5 0.5 -0.5
v -0.5 -0.5 -0.5
v 0.5 -0.5 -0.5
v 0.5 0.5 -0.5
# Lista de faces
f 1 2 3
f 8 7 6
f 4 3 7
f 5 1 4
f 5 6 2
f 2 6 7
f 1 3 4
f 8 6 5
f 4 7 8
f 5 4 8
f 5 2 1
f 2 7 3
# fim do arquivo
```

Alguns arquivos .obj como o acima não possuem as normais calculadas, nesses casos as normais podem ser calculadas no código após o carregamento do modelo para permitir o uso de algoritmos de iluminação como Gourard. Em http://cin.ufpe.br/~marcelow/Marcelow/calculovetornormal.html o professor Marcelo Walter dá detalhes de como calcular as normais. Também existem modelos que contêm as normais calculadas (diretiva vn). Também existem arquivos .obj com informação de textura (diretiva vt). É possível conferir a especificação de arquivos .obj em http://cin.ufpe.br/~marcelow/Marcelow/mcompl_files/obj_file.html. O projeto deve ser capaz de carregar todos os arquivos .obj disponibilizados em http://cin.ufpe.br/~marcelow/Marcelow/arquivos_obj.html.

VISUALIZAÇÃO EM JANELA DO GLUT

Renderizar o resultado final em uma janela gerenciada pela biblioteca auxiliar GLUT.

INTERAÇÃO COM .OBJ – ROTAÇÃO, TRANSLAÇÃO, ESCALA

Usar as funções de *callback* do GLUT para usar o teclado como entrada para aplicar transformações nos objetos. Assumindo que sempre existe um modelo ou fonte de luz selecionada, seguem as funções associadas a cada tecla:

- ',' ou '<': seleciona o modelo anterior (ao chegar no primeiro modelo, passa para a última fonte de luz)
- '.' ou '>': seleciona o modelo seguinte (ao chegar no último modelo, passa para a primeira fonte de luz)
- '1': translada o objeto selecionado no eixo X no sentido negativo em coordenadas de mundo
- '2': translada o objeto selecionado no eixo X no sentido positivo em coordenadas de mundo
- '3': translada o objeto selecionado no eixo Y no sentido negativo em coordenadas de mundo
- '4': translada o objeto selecionado no eixo Y no sentido positivo em coordenadas de mundo
- '5': translada o objeto selecionado no eixo Z no sentido negativo em coordenadas de mundo
- '6': translada o objeto selecionado no eixo Z no sentido positivo em coordenadas de mundo
- '7': gira o objeto selecionado em relação ao eixo X
- '8': gira o objeto selecionado em relação ao eixo Y
- '9': gira o objeto selecionado em relação ao eixo Z
- '-' ou '_': modificar a escala do objeto decrementando seu tamanho em 1%
- '+' ou '=': modificar a escala do objeto incrementando seu tamanho em 1%

MUDANÇAS NA CÂMERA – ROTAÇÃO, TRANSLAÇÃO

A câmera virtual deve ser interativa, mover-se ao longo da cena de acordo com o input do usuário. A modificação deve ser realizada diretamente na matriz de parâmetros extrínsecos. Não é autorizado o uso da função *gluLookAt* tampouco de combinações de *glRotate* e *glTranslate*. As alterações dos parâmetros extrínsecos devem ser aplicadas diretamente na matriz que vai ser carregada como *modelview* no OpenGL, para isso se usam as funções *glMatrixMode(GL_MODELVIEW)* e *glLoadMatrixf(extrinsic)* em que *extrinsic* é a matriz com 16 *floats* definindo os parâmetros extrínsecos da sua câmera. A seguir a descrição de como deve ocorrer a interação:

- 'w' ou 'W': move para frente de acordo com o vetor diretor (eixo Z em coordenadas de câmera)
- 's' ou 'S': move para trás de acordo com o vetor diretor (eixo Z em coordenadas de câmera)
- 'a' ou 'A': move para a esquerda de acordo com o vetor lateral (eixo X em coordenadas de câmera)
- 'd' ou 'D': move para a direita de acordo com o vetor lateral (eixo X em coordenadas de câmera)
- Botão esquerdo do mouse pressionado seguido de movimento do mouse: usar o delta de movimento (diferença entre X e Y antigos e atuais do ponteiro) para rotacionar o vetor diretor da câmera para esquerda/direita de acordo com o delta X e cima/baixo de acordo com o delta Y. A rotação lateral deve considerar o eixo Y do mundo virtual, enquanto a rotação vertical deve considerar o eixo lateral X da câmera. Dica: para aplicar rotação em torno de um eixo arbitrário $\boldsymbol{w}$ usar a matriz de rotação de Rodrigues (http://mathworld.wolfram.com/RodriguesRotationFormula.html):

$$\begin{bmatrix} \cos\theta + \omega_x^2(1-\cos\theta) & \omega_x\omega_y(1-\cos\theta) - \omega_z\sin\theta & \omega_y\sin\theta + \omega_x\omega_z(1-\cos\theta) \\ \omega_z\sin\theta + \omega_x\omega_y(1-\cos\theta) & \cos\theta + \omega_y^2(1-\cos\theta) & -\omega_x\sin\theta + \omega_y\omega_z(1-\cos\theta) \\ -\omega_y\sin\theta + \omega_x\omega_z(1-\cos\theta) & \omega_x\sin\theta + \omega_y\omega_z(1-\cos\theta) & \cos\theta + \omega_z^2(1-\cos\theta) \end{bmatrix}$$

ILUMINAÇÃO DOS OBJETOS

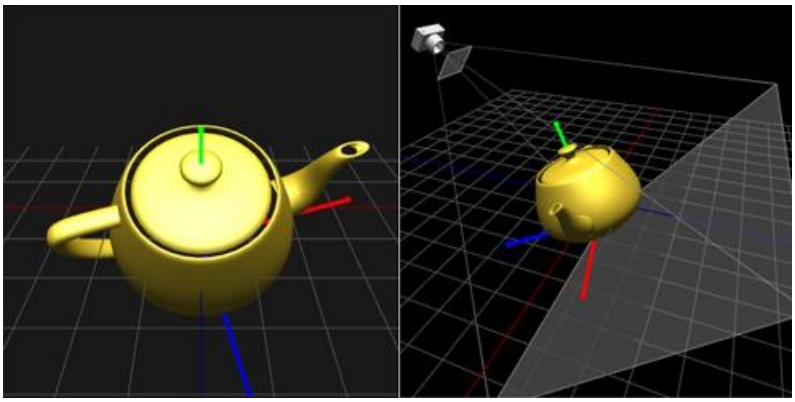
Deve existir pelo menos duas fontes de luz de cores diferentes (uma branca e outra de cor à sua escolha) e posicionadas em pontos distantes da cena. Estas fontes de luz serão manipuladas da mesma forma que os modelos, com exceção das modificações da rotação já que as fontes de luz nesse caso são pontuais.

MÓDULOS ESPECÍFICOS (20% CADA)

VISÃO DO DIRETOR (VER A CÂMERA SE MOVENDO A PARTIR DE OUTRA VISTA)

O objetivo é dividir a tela verticalmente em duas partes mostrando a cena a partir de dois pontos de vista. O primeiro ponto de vista é o mesmo do módulo básico do projeto. O segundo ponto de vista é de uma nova câmera virtual que fica parada observando a cena à distância. Na visão do diretor, a câmera deve aparecer representada por um objeto composto (um paralelepípedo e um cilindro):

Segue um exemplo de como deve ficar a visualização:



DOLLY ZOOM EFFECT

Permitir que o usuário experimente o *Dolly Effect* (encontrei referências citando efeito similar com a nomenclatura *Frantic Zoom/Trombone Effect*). O efeito garante que a distância focal e a posição da câmera sejam modificados em sentidos invertidos de forma que a distância dos objetos da cena até o plano de projeção da câmera permaneça a mesma. O efeito deve ser ativado pela interação com roda do mouse ao mesmo tempo que se pressiona a tecla 'd'. Segue o link de um vídeo ilustrando o comportamento do efeito na cena: http://upload.wikimedia.org/wikipedia/commons/5/5f/DollyZoomTest.ogv. Como curiosidade adicional, segue o link de um vídeo explicitando o uso do efeito no cinema ao longo dos anos: http://vimeo.com/84548119.

ILUMINAÇÃO COM SOMBRAS E NÉVOA

Dispor 8 fontes de luz diferentes e projetar a sombra dos objetos da cena no chão (plano horizontal). Estas fontes de luz também devem ser transladáveis da mesma forma que as do módulo básico. Adicionalmente deve ser implementado o efeito de névoa, deixando os pontos mais distantes da câmera mais acinzentados enquanto que os pontos próximos continuam bem definidos. A intensidade da névoa deve ser modificada pela interação com roda do mouse ao mesmo tempo que se pressiona a tecla 'n'.

MAPEAMENTO DE TEXTURAS

Renderização de quatro objetos diferentes com mapeamento de texturas. Os quatro devem ser apresentados na cena, cada um contendo uma das seguintes formas de mapeamento: texturização de cor (com informações de cor a partir de um arquivo de imagem) e três tipos de distorções: *Bump mapping*, *Normal mapping*, *Reflection mapping*.

NURBS

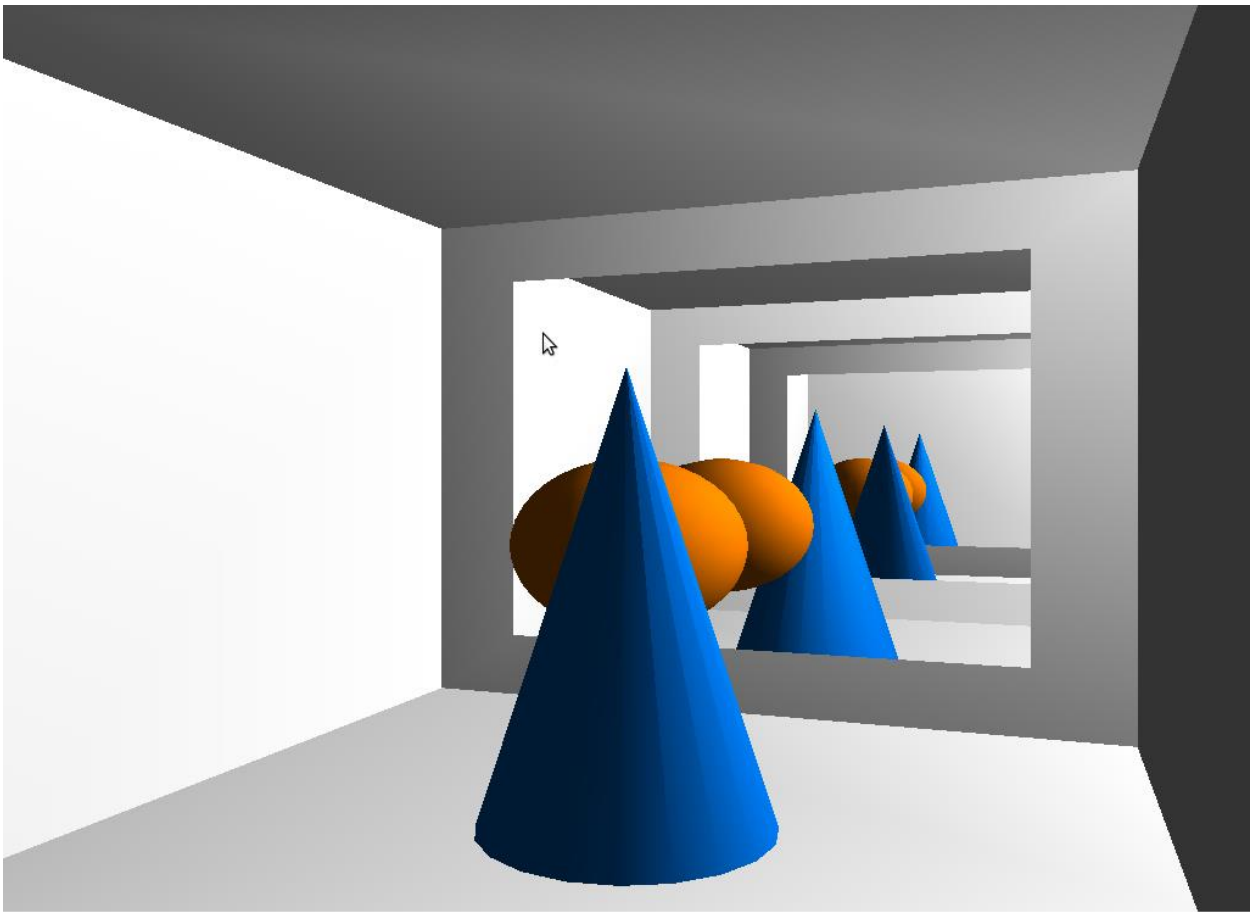
Ler um arquivo que descreve um grid de pontos de controle no formato .nurb (formato criado para o projeto) como descrito abaixo:

```
# cilindro.nurb
# cilindro representado por superfície racional paramétrica
# Linhas que iniciam com # sao comentarios
# Autor: Lucas S. Figueiredo
# N é o número de pontos de controle do grid em um dos sentidos (relacionado ao variação paramétrica de u)
# M é o número de pontos de controle do grid no outro sentido (relacionado ao variação paramétrica de v)
N 9
M 3
# Lista de pontos de controle no formato X Y Z W
# W é o peso do ponto
1       0       -1      1
1       1       -1      0.7071067811865475
0       1       -1      1
-1      1       -1      0.7071067811865475
-1      0       -1      1
-1      -1      -1      0.7071067811865475
0       -1      -1      1
1       -1      -1      0.7071067811865475
1       0       -1      1

1       0       0       1
1       1       0       0.7071067811865475
0       1       0       1
-1      1       0       0.7071067811865475
-1      0       0       1
-1      -1      0       0.7071067811865475
0       -1      0       1
1       -1      0       0.7071067811865475
1       0       0       1

1       0       1       1
1       1       1       0.7071067811865475
0       1       1       1
-1      1       1       0.7071067811865475
-1      0       1       1
-1      -1      1       0.7071067811865475
0       -1      1       1
1       -1      1       0.7071067811865475
1       0       1       1

# fim do arquivo
```

Os pontos de controle devem ser selecionáveis e transladáveis pelo mesmo modo de interação utilizado para modificar os modelos .obj descritos no módulo básico.

REFLEXÃO (JOGO DE ESPELHOS)

Mostrar 4 espelhos na cena. 2 destes devem estar um de frente para o outro. Os demais podem estar dispostos em qualquer ponto da cena. Dica: é preciso simular uma câmera adicional para cada espelho, projetar a imagem destas em uma textura (em vez de projetar diretamente na tela) e daí então aplicar esta textura aos retângulos (GL_QUADS) que representam os espelhos na cena. Abaixo segue um exemplo demonstrando o efeito:



Uma das formas para alcançar o efeito em OpenGL é necessário criar uma câmera adicional para cada um dos espelhos, esta câmera adicional deve ter como plano de projeção (*near*, *right*, *left*, *top* e *bottom* do *frustum*) fixado nas coordenadas do espelho no mundo virtual, seu ponto central deve ser oposto ao ponto do observador e sua direção (vetor diretor da câmera nos parâmetros extrínsecos) deve ser a mesa da normal do espelho. Feito isso utiliza-se opção em OpenGL de renderizar em texturas (*render to texture*) para usar essa textura no espelho virtual.

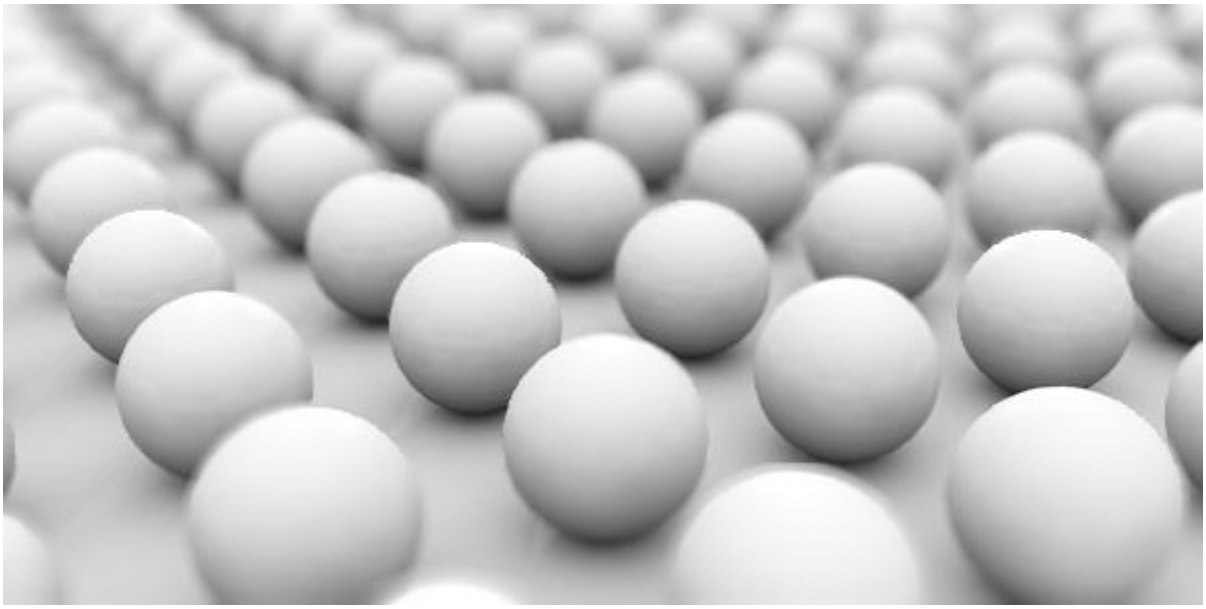
PINTOR DE OBJETOS

Mostrar uma paleta de cores no canto inferior direito da tela como mostrado na figura ao lado. Ao clicar em uma das cores da paleta e clicar em seguida em uma das faces do objeto, a face clicada assume a nova cor. Alternativamente, ao apertar a tecla 'p' o pintor de objetos é modificado para pintar o modelo clicado por completo ao invés de uma única face. Se a tecla 'p' for apertada novamente, o pintor de objetos retorna ao modo de pintar faces.

Segue um link com um exemplo de como recuperar o ponto 3D da cena sob o ponteiro do mouse: http://nehe.gamedev.net/article/using_gluunproject/16013/#5

DESAFIO: CAMPO DE PROFUNDIDADE

Forçar o foco da imagem renderizada para que seja em uma distância específica, desfocar/borrar os demais pontos à medida que eles se distanciam da distância ideal para o foco. Dica: renderizar os pixels em uma textura ao invés de mostrá-los diretamente na tela, daí então aplicar um algoritmo de borramento nos pixels que representam pontos da cena com a profundidade (distância entre o pixel e a câmera) não ideal usando técnicas de processamento de imagem. Quão mais distante da profundidade ideal do foco, mais borrada deve ficar a região da imagem. Segue um exemplo de como o efeito deve se comportar em uma cena:



CRITÉRIOS DE AVALIAÇÃO

ASPECTO VISUAL

- ☐ Resolução da imagem de saída (meta é HD ou Full HD)
- ☐ Rodar em tela cheia (*fullscreen*)
- ☐ Percepção clara dos efeitos implementados

PERFORMANCE

- ☐ Taxa de atualização em tempo real (meta é 30 quadros por segundo)
- ☐ Adicionar diversas instâncias de modelos 3D (a meta é 10 instâncias de 5 modelos diferentes, tendo um deles pelo menos 1000 polígonos)

IMPLEMENTAÇÃO

- ☐ Clareza no código implementado (código limpo, livre de códigos comentados e com comentários explicando as principais classes e funções)
- ☐ Código modularizado (separar em classes ou pelo menos em arquivos distintos cada uma das responsabilidades do projeto)
- ☐ Clareza na explicação da implementação (cada integrante do grupo deve explicar sua participação no projeto de forma consistente e clara)

DESAFIO: APLICABILIDADE

Resultados devem ter potencial de aplicação real no mercado, responder perguntas como:

Em que cenário seu resultado é útil? _____

Que empresas se interessariam pela tecnologia? _____