

Trabalho Prático

Descrição

Desenvolver uma aplicação em qualquer uma das linguagens usadas na disciplina (Java, C++ e Python) que atenda os seguintes requisitos:

- Adicionar e remover (uma ou mais) formas:
 - Cubo, esfera (icosphere), cone e torus;
- Habilitar e desabilitar reflexão:
 - Especular, difusa e ambiente;
- Adicionar e remover fontes de luz (pontuais);
- Definir tipo de projeção e seus parâmetros associados:
 - Ortográfica ou perspectiva;
- Câmera;
 - Definir vetor de visão e posição da câmera;
- Transformar as formas:
 - Translação, Rotação, Escala e *Shear*.
- Mostrar/Esconder os eixos;
- Salvar a imagem corrente em arquivo PNG;
- Mostrar as luzes posicionadas na cena;
- Ler comandos de entrada padrão por linha de comando.

Informações sobre a Implementação

- As formas adicionadas inicialmente devem estar na posição (0,0,0) e conter as seguintes dimensões – opções do blender: *cube* (diagonais=1,1,1 e -1,-1,-1), *sphere* (icosphere, 5 subdivisões, size=1 - <https://www.youtube.com/watch?v=1nRZsWUlc5A>), *cone* (vértices=32, raio1=1, raio2=0, profundidade=2) e *torus* (major seg=48, minor seg=12, major radius=1, minor radius=0.25) – Exportar no Blender no modo *smooth* (botão direito sobre o objeto e selecionar *shade smooth*, depois exportar).
- A cada execução de um novo comando, a imagem sendo renderizada deve ser atualizada;
- É obrigatório ler comandos descritos na tabela anterior de entrada padrão por linha de comando. Caso alguma funcionalidade não tenha sido implementada, responda com a mensagem “NOT IMPLEMENTED”;
- A câmera inicialmente deve estar na posição (0, 0, 0); O *lookat* (0, 0, -1) e o vetor *up* (0, 1, 0). Parâmetros para *ortho* (-2f, 2f, -2f, 2f, 2f, -2f) e *frustum*: (-1f, 1f, -1f, 1f, 0.5f, 2f). Projeção padrão: *ortho*;
- Shading padrão: *none*;
- Valores iniciais de k para todas as reflexões= 0.2 .Não vai fazer diferença para o shading *none*.
- Parâmetros para a reflexão especular n = 32;
- Podem ser utilizadas bibliotecas externas desde que sejam apresentadas no relatório junto com o link para download e com informações sobre ela;
- Resolução da tela de desenho deve ser de 640x640;
- Qualquer parâmetro ou valor não definido neste documento pode ser fixo e deve ser escolhido pelo aluno desde que explicado no relatório;

- Todas as luzes devem ser brancas e pontuais, máximo 10 inseridas. O cálculo da intensidade de luz deve usar a atenuação de acordo com a posição da luz na cena. Parâmetros para a atenuação: $a_0=1$, $a_1=1$ e $a_2=0.25$;
- Cor de fundo padrão deve ser preta e de desenho inicial branca;
- **É obrigatória a implementação usando a API Programável da OpenGL (3.0+)** – usando *shaders* na GLSL;
- Não é necessária nenhuma interação com o mouse ou teclado, mas pode ser implementada.

Tabela de comandos de entrada

| Comando | Descrição |
|---|---|
| add_shape <u>shape</u> <u>name1</u> | Adiciona a forma correspondente de tamanho centrada na origem com o nome <u>name1</u> a cena. Formas possíveis: cube, sphere, cone e torus. Ex.: add_shape cube cube1 |
| remove_shape <u>name1</u> | Remove a forma com o nome <u>name1</u> da cena. Ex.: remove_shape cube1 |
| add_light <u>name1</u> <u>float float float</u> | Adiciona uma nova luz na posição (x, y, z) na cena. Máximo 10. Ex. add_light light1 0.5 0.3 1.0 |
| remove_light <u>name1</u> | Remove a luz com o nome <u>name1</u> da cena. Ex.: remove_light light1 |
| reflection_on <u>type</u> <u>float</u> | Habilita o tipo de reflexão da luz nos objetos da cena e define o coeficiente de reflexão associado (k). Coeficiente de brilho da reflexão especular fixo. Tipos de reflexão possível: specular, diffuse e ambient. Ex.: reflection_on specular 0.3 |
| reflection_off <u>type</u> | Desabilita o tipo de reflexão da luz nos objetos da cena. Tipos de reflexão possível: specular, diffuse e ambient. Ex.: reflection_off diffuse |
| shading <u>type</u> | Define o tipo de shading usado para renderizar a cena. Tipos de shading possíveis: none, flat, smooth e phong. Ex.: shading smooth |
| projection <u>type</u> <u>float float float</u> <u>float float float</u> | Define o tipo de projeção. <u>type</u> pode ser: ortho ou perspective. Tamanho do frustum definidos pelos parâmetros (left, right, bottom, top, nearVal, farVal) Ex.: projection perspective -1 1 -1 1 0.5 2 |
| translate <u>name1</u> <u>float float float</u> | Translada a forma correspondente de acordo com os três parâmetros em (x, y, z). Ex.: translate cube1 1.0 0.0 0.0 |
| scale <u>name1</u> <u>float</u> <u>float float</u> | Redimensiona a forma correspondente de acordo com os três parâmetros em (x, y, z). Ex.: scale cube1 0.5 0.5 0.5 |
| shear <u>name1</u> <u>float</u> <u>float float float</u> <u>float float</u> | Inclina a forma correspondente de acordo com os seis parâmetros em (syx, szx, sxy, szy, sxz, syz). Ex.: shear cube1 0.5 0 0 0 0 0 |
| rotate <u>name1</u> <u>float</u> <u>float float float</u> | Rotaciona a forma correspondente de acordo com o ângulo e o vetor (x, y, z). Ex.: rotate cube1 45 1.0 0.0 0.0 |
| lookat <u>float float</u> <u>float</u> | Faz a câmera olhar para o ponto definido pelos parâmetros (x,y,z). Ex.: lookat 0.0 0.0 0.0 |
| cam <u>float float</u> <u>float</u> | Define a posição da câmera por meio dos parâmetros (x,y,z). Ex.: cam 0.0 0.0 0.0 |
| color <u>name1</u> <u>float</u> <u>float float</u> | Define a cor da forma com nome <u>name1</u> usando três parâmetros (r, g, b). Ex.: color cube1 1.0 0.0 0.0 |
| save <u>filename</u> | Salva a visualização corrente em uma imagem png com o nome <u>filename</u> . Ex.: save test.png |
| axis_on | Mostra os eixos x, y e z. x: Eixo x vermelho, y verde e z azul. |
| axis_off | Esconde os eixos x, y e z. x: Eixo x vermelho, y verde e z azul. |

| | |
|------------|---|
| lights_on | Mostra as luzes posicionadas na cena (um ponto amarelo para cada luz). |
| lights_off | Esconde as luzes posicionadas na cena (um ponto amarelo para cada luz). |
| wire_on | Mostra todos os objetos no formato wireframe (cor das linhas deve ser a cor do objeto). |
| wire_off | Mostra todos os objetos no formato normal (não wireframe). |
| quit | Sai do programa. |

Relatório

Entregar um **relatório em PDF** apresentando:

- Nomes dos alunos;
- Linguagem de programação e versão;
- Versão da OpenGL e GLSL;
- Parâmetros fixos (constantes) utilizados na implementação;
- Informações sobre os arquivos no zip (código do projeto enviado);
- Funções (implementações) adicionais da aplicação,
- Outras informações que achar pertinente.

Não entregar o relatório impresso. O relatório deve **obrigatoriamente** ter entre 1 e 5 páginas **no total**. Não deve ter capa, contracapa, bibliografia ou índice.

Entrega

Entregar o projeto zipado com todos os códigos desenvolvidos e o PDF do relatório. O projeto deve ser desenvolvido **em dupla ou individualmente**. No caso da dupla, a nota vai ser a mesma para ambos os alunos. Basta que apenas um aluno da dupla faça a entrega no AVA. A data de entrega será definida no AVA.

Avaliação

A avaliação do trabalho vai ser realizada de acordo com a implementação e o relatório entregue. No total a nota do trabalho é no máximo 10 (verificar o peso do trabalho na nota final no SISCAD). Todo o código deve ser comentado.

Se eu desconfiar que o trabalho foi copiado de outro colega ou da internet (ou de alguma maneira não foi você que fez) eu vou convidá-lo a me explicar o código na minha sala. Em caso de confirmação, é zero para o trabalho.

Quaisquer dúvidas no desenvolvimento do trabalho devem ser enviadas para o e-mail andvicoso@facom.ufms.br ou pessoalmente.

Bom trabalho!