Universidade Estadual Vale do Acaraú
Curso de Ciências da Computação
Disciplina: Computação Gráfica

Lista 2 de Programação Gráfica com OpenGL – Operações com Câmera

Data de Entrega: 22/07/2019

Pontos possíveis: até 2 pontos na primeira avaliação parcial

Forma de Entrega: crie um repositório no github e coloque o código produzido e os documentos gerados neste repositório até o dia 22 de julho. No dia 22 de julho, envie um email para gilzamir_gomes@uvanet.br com o link para o repositório.

Todas as questões nesta lista, ao mencionarem exemplo capX_exY, referem-se ao conteúdo disponível em http://github.com/professorgilzamir/pgp .

1. É costume em aplicações clássicas de OpenGL separarmos três tipos matrizes de transformação: transformação de modelo, transformação de visão ou de câmera e transformação de projeção. A transformação de modelo (normalmente chamada de *model* muda as coordenadas dos objetos para coordenadas do mundo – por isso, geralmente temos uma matriz *model* para cada objeto. A matriz de câmera (normalmente chamada de *view*) muda as coordenadas de mundo para obtermos as coordenadas correspondentes da câmera – por isso, normalmente temos uma matriz *view* para cada câmera na aplicação. Finalmente, a matriz de projeção (normalmente chamada de *projection*) projeta as coordenadas dos objetos em relação à câmera no plano de projeção da câmera (é normal esta matriz já realizar a normalização necessária). Do mesmo modo que a matriz *view*, tem-se que há uma matriz de projeção para cada câmera na aplicação. Por isso, aplicações geralmente utilizam um padrão de três matrizes que normalmente chamamos de *model-view-projection.* Identifique no exemplo cap2_ex5 como este processo está sendo realizado e que operações envolvem. Observe que a cena tem apenas uma câmera e apenas um objeto. Se tivéssemos uma cena complexa com mais de um objeto, seria interessante termos que organizar o nosso código com técnicas de programação orientada a objetos. Nesta conceituação, quais atributos e membros mínimos teriam que ter as classes que representariam os objetos tridimensionais da cena e a câmera?
2. Verifique o código do exemplo cap2_ex5. Na função *initializeMatrix*, são utilizadas as funções *identity*, *perspective*, *scaleMatrix4*, *translationMatrix4* e *multMatrix4*. Descreva o que significa e como utilizar cada uma destas funções. Dê exemplos de entrada e a saída correspondente de cada função.
3. Verifique o código do exemplo cap2_ex5. Na função *updateMatrix*, são utilizadas as funções *rotationXMatrix4* e *rotationYMatrix4*. Descreva o que significa e como utilizar cada uma destas funções. Dê exemplos de entrada e a saída correspondente de cada função.

4. O exemplo cap2_ex5 mostra um cubo visto por uma câmera. A câmera pode ser girada com as teclas I (para cima), K (para baixo), J (para a direita) e L (para a esquerda). Identifique a parte do código que realiza esta funcionalidade e as transformações envolvidas (matrizes de transformação).
5. Altere o exemplo cap2_ex5 de modo que a cena contenha dois cubos (um azul e um vermelho) lado a lado (com um pequeno espaço entre eles) e uma câmera visualizando estes objetos alternando o foco entre o objeto vermelho e o objeto azul em intervalos regulares de tempo.
6. Altere o exemplo cap2_ex5 de movo que o movimento da câmera seja feito por meio do mouse (de forma parecida como é feito nos jogos do tipo FPS – *First Person Shotter*).