Relatório Exercício Prático 6

Gustavo Ciotto Pinton 117136
EA979 - Processamento de Imagens

1 Explicação

A biblioteca *OpenGL* utiliza, em seu processamento, um modelo contendo duas pilhas de matrizes, chamadas de *model-view* e *projection*, sendo que, após a inicialização da aplicação, cada pilha contém apenas uma matriz identidade. Tais matrizes representam um sistema de coordenadas normalizadas, chamadas, em inglês, de *normalized device coordinates*, cujos eixos X, Y e Z variam no intervalo de [-1,1]. A fim de mapear tais eixos em coordenadas da janela de visualização, é necessário aplicarmos a transformação *viewport transform* a partir da chamada `glViewport()`, passando como parâmetro a largura e altura da respectiva janela. No caso do exerício proposto, essa função é chamada dentro do escopo de `reshape()`, que é, por sua vez, chamada toda vez que a janela é redimensionada.

As transformações, chamadas de *projection transform*, realizam a conversão de cenas 3D, renderizadas pela biblioteca, em projeções 2D, a fim de que tais cenas sejam possíveis de serem visualizadas em uma tela de monitor. Este tipo de transformação deve, portanto, ser capaz de lidar com as idéias de profundidade e redimensionamento de objetos que estejam em planos distintos. Duas variedades de *projection transform* são implementadas: *orthographic* e *perspective*, sendo esta última a mais importante e a que foi utilizada na execução deste exercício. A *perspective transform* imita o funcionamento do olho humano e é chamada através da função `gluPerspective`, que recebe como parâmetro as posições do planos no eixo Z `zNear` e `zFar`. Essa função define uma pirâmide cujo topo é posicionado nas coordenadas do *olho* do visualizador da cena, que está olhando na direção negativa de Z. `zNear` e `zFar` definem, portanto, as coordenadas de Z consideradas pelo visualizador como perto e longe, respectivamente. No caso do exerício proposto, a função `gluPerspective` recebe os parâmetros 1 e 20 para `zNear` e `zFar` e também é chamada a cada redimensionamento da janela (`resize()`).

A última transformação, chamada de *model-view transform*, é responsável por mover as **coordenadas do sistema**, de maneira que o resultado seja conforme à *pirâmide* posicionada no *olho* do observador. Esta transformação é capaz de simular, por exemplo, translações, rotações e redimensionamentos. Faremos o uso justamente deste tipo de tranformação para desenharmos diversas esferas ao redor da esfera maior, que encontra-se no centro. Para desenharmos uma esfera, cujo centro encontra-se em um ponto (x, y), realizamos as seguintes operações dentro da função `display()` do programa:

i. Modificamos a cor a partir do comando `glColor3f()`, passando como parâmetro as frações de cada cor RGB;

ii. Deslocamos o **sistema de coordenadas** de $(\Delta x, \Delta y)$ a partir da função `glTranslatef()`, que recebe os *offsets* de cada coordenada;

iii. Desenhamos a esfera no centro do novo sistema de coordenadas, obtido da translação realizada no passo anterior. A função `glutWireSphere()` é responsável por fazê-lo, recebendo como parâmetro o raio da esfera e a quantidade de traços a serem desenhados.

É possivel, a partir do passos anteriores, desenharmos 4 esferas posicinadas nas coordenadas $(0, 0, 1.6)$, $(0, 0, -1.6)$, $(0, 1.6, 0)$ e $(0, -1.6, 0)$. Para a primeira, realizamos as seguintes etapas:

```
glTranslatef(0, 0, 1.6);          /* Move coordenada em Z */
glColor3f(1.0f,1.0f,1.0f);        /* Muda cor */
glutWireSphere(0.4, 20, 16);      /* Desenha esfera no centro desta nova coordenada.*/
glTranslatef(0, 0, -1.6);         /* Reverte translacao  */
```

Repete-se tal processo, modificando a coordenada desejada e obtém-se finalmente a figura 1, logo abaixo. As esferas de cor rosa e azul escuro foram obtidas a partir da translação do eixo **Y**, sendo que as esferas branca e azul clara, a partir da translação do eixo **Z**. Estas últimas não são claramente visíveis, visto que estão, respectivamente, à frente e atrás da esfera amarela de raio unitário.

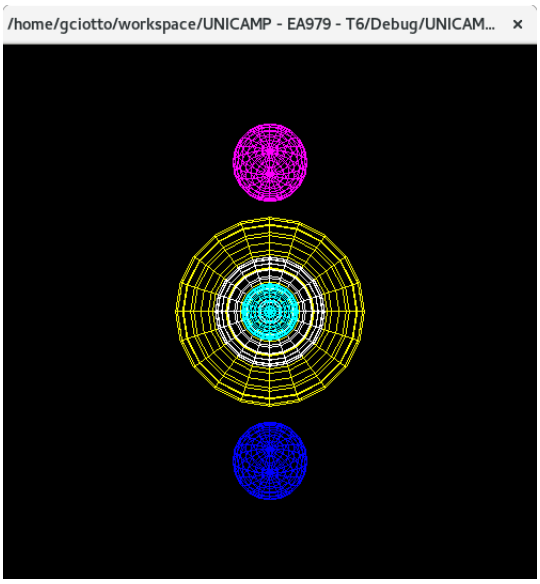


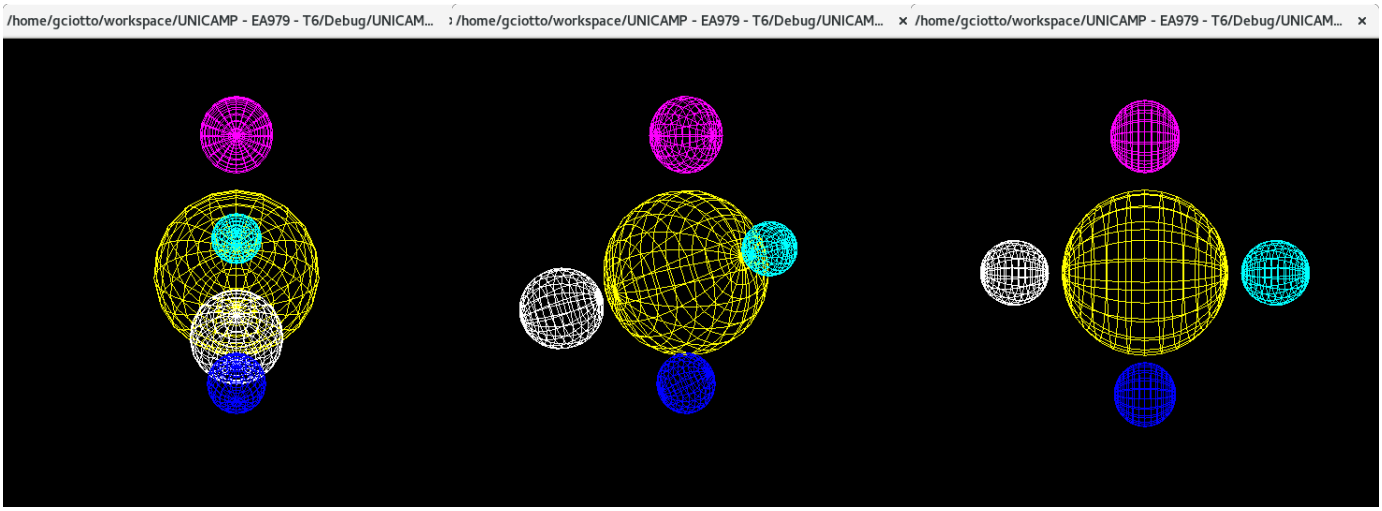
Figura 1: Janela obtida após lançamento do programa.

Por fim, a documentação das funções nos aconselha a englobar todas as tranformações *model-view* dentro de um bloco `glPushMatrix()` - `glPopMatrix()`, a fim de preservarmos o sistema de coordenadas anterior à realização das operações.

A última etapa consistiu na rotação das esferas pela função `glRotatef`. Além da rotação em torno do eixo **X**, já fornecido no modelo, adicionei também a rotação ao redor do eixo **Y**. Para isso, criei uma nova variável global, chamada de `cameraX`, que é incrementada/decrementada a cada pressionamento da tecla X/x do teclado na função `keyboard()`. O trecho abaixo, adicionado logo antes dos comandos que desenham as esferas, representa tal funcionalidade:

```
glRotatef ((GLfloat) camera, 1.0, 0.0, 0.0);  /* Roda em torno de X */
glRotatef ((GLfloat) cameraX, 0.0, 1.0, 0.0); /* Roda em torno de Y */
```

As figuras abaixo representam algumas rotações realizadas pelo programa.



(a) Exemplo 1. (b) Exemplo 2. (c) Exemplo 3.

Figura 2: Exemplos da execução do programa.

Referências

1. *What exactly are eye space coordinates?*, disponível em `http://stackoverflow.com/questions/15588860/what-exactly-are-eye-space-coordinates`.