INF01047 Fundamentos de Computação Gráfica
Pratica 2

Objetivos:

- Reforço do entendimento da estrutura de programas com OpenGL
- Uso de menus
- Transformações geométricas
- Transformações Hierárquicas

Material Suporte

O arquivo pratica2_source_files.zip disponível no moodle

Exercício 01

a) Crie um projeto para o arquivo **BasicoMenu.cpp**.
Estude o programa, compile e execute.

b) Observe como menus foram criados. Ao clicar com o botão direito do mouse, você pode interagir com as opções do menu.

c) Observe a nova função introduzida no programa para tratamento de opções selecionadas através do menu pop-up controlado pelo botão direito do mouse: TrataMenu. De que forma ela são informadas para a OpenGL na construção do menu na função *main* do programa? Diferentes funções podem ser especificadas para diferentes menus, embora neste exemplo tenhamos centralizado o tratamento de todas as opções.

Observando alguns trechos do código:

```
// constantes usadas para identificar objetos
#define TRIANGULOCHEIO 1
#define TRIANGULOARAME 2
#define LINHA 3
#define QUADRILATERO 4

// guarda o tipo de objeto que deve ser  exibido
int objeto;

-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
// Callback de menu
void TrataMenu ( int valor )
{
    switch(valor)
      {
      case 1: // cor vermelha
         glColor3f (1.0f, 0.0f, 0.0f);
         break;

      case 2: // cor azul
         glColor3f (0.0f, 0.0f, 1.0f);
         break;

      case 3: // cor verde
         glColor3f (0.0f, 1.0f, 0.0f);
         break;
      }

    glutPostRedisplay();
}

-------------------------------------------------------------------------------------------------------
```

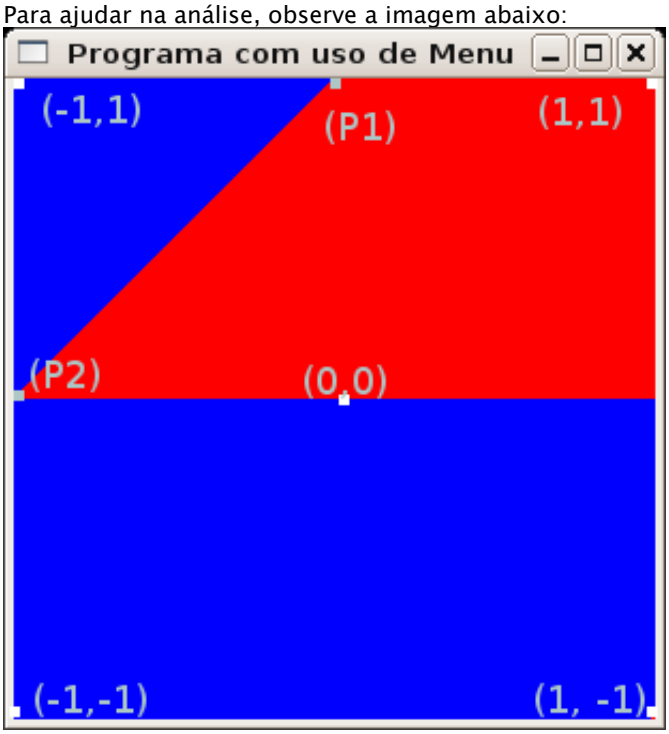
```
-----------------------------------------------------------------------------------------------
// Cria menu para escolher a cor corrente e o objeto a ser exibido
   int menuCor;        // sub-menu para cor do objeto
   int menuObjeto;     // sub-menu para tipo de objeto
   int mainMenu;

   menuCor = glutCreateMenu(TrataMenu);
   glutAddMenuEntry("Vermelho",1);
   glutAddMenuEntry("Azul",2);
   glutAddMenuEntry("Verde",3);
   glutAddMenuEntry("Branco",4);
   glutAddMenuEntry("Preto",5);
   glutAddMenuEntry("Amarelo",6);

-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
```

glutAttachMenu(GLUT_RIGHT_BUTTON); //Ao clicarmos o botão direito do mouse, o menu ser mostrado

d) Acrescente uma opção no menu, ao final de todas, para reduzir o tamanho do objeto. Esta mudança deve ser ativada através da chamada de uma função que reduza as coordenadas dos pontos à metade de seu valor original. Uma opção de "reset" disparada pela tecla "r" do teclado deve devolver as coordenadas para seu tamanho original. Observação: neste exercicío você não deve utilizar a função de OpenGL que possibilita uma transformação de escala.

e) Exiba o quadrilátero em sua posição original. Considerando o que aparece na janela, como podes explicar que o quadrilátero aparece incompleto? Quando você aplica a função de redução ao quadrilátero, qual o resultado em termos de imagem? Em OpenGL, a região visível default do Sistema de Referência do Universo (SRU) tem limites (-1,-1) e (1,1).

Para ajudar na análise, observe a imagem abaixo:



Exercício 02

a) Crie um projeto para o código fonte **Transforma.cpp**. Compile e execute o programa. Identifique, no código as transformações realizadas. Investigue o resultado na mudança de ordem das funções de transformações geométricas utilizadas.

Observações: na API OpenGL, as funções de transformação são compostas numa matriz de transformação corrente, considerando uma multiplicação na seguinte ordem:

Matriz corrente = Matriz armazenada * Transformações chamadas
Ponto transformado = Matriz corrente * Ponto passado na função *glVertex*

Isto significa que é realizada uma pré-multiplicação. Assim, as chamadas das funções de transformação devem ser feitas na ordem inversa da sua aplicação. Em outras palavras, observando o código, a função de transformação imediatamente anterior à *glVertex* é a primeira a afetar o objeto.

```
glTranslatef(0.0, 10.0, 0.0); // translação  em y
glRotatef (45,0.0, 0.0, 1.0); // rotação de 30 graus em torno do eixo Z
glBegin (GL_LINES);
   glVertex2f (0.0f, 0.0f);
   glVertex2f (1.0f, 1.0f);
glEnd();
```

No exemplo acima, a linha com vértices (0,0) e (1,1) seria rotacionada em torno de (0,0), ficando sobre o eixo y e os vértices seriam transladados de dy=10, ficando o primeiro vértice no ponto (0,10).

Observe ainda que as funções são as mesmas para 2D e 3D.

b) Utilizando a facilidade de menus da GLUT que você viu no programa BasicoMenu.cpp, crie um menu que controle a realização de transformações do seguinte modo:

Crie um menu com cinco opções:

- Translações
- Rotações
- Escala
- Translações e rotações
- Rotações e Translações

Estas opções estabelecem como devem ser interpretadas as teclas de setas:

- Se for translação, as teclas modificam a posição do objeto;
- Se for rotação, a tecla "seta para direita" deve causar uma rotação de 25 graus no sentido horário; "seta para a esquerda", 25 graus no sentido anti-horário.
- Se for escala, a tecla "seta para cima" deve causar um aumento do tamanho do objeto em 10% e "seta para baixo", diminui na mesma proporção.
- Nas opções 4 e 5 deve ser uma rotação em z. Elas retornam resultados iguais? Porque?

Neste exercício, você deve utilizar as funções de transformação fornecidas pela OpenGL e usadas no programa **Transforma.cpp.** Você pode utilizar as funções de transformação de OpenGL em qualquer ponto do programa, mas lembre-se que todas as primitivas geométricas (vértices) informadas após transformações, sofrerão as mesmas.

Lembre de concentrar as chamadas de primitivas geométricas na função *RenderScene*, pois é esta que é executada para re-exibir a imagem. Colocar funções de desenho fora da callback de display não garante que elas sejam exibidas sempre, a não ser que estejam em funções explicitamente chamada dentro da *RenderScene*.

Exercício 03

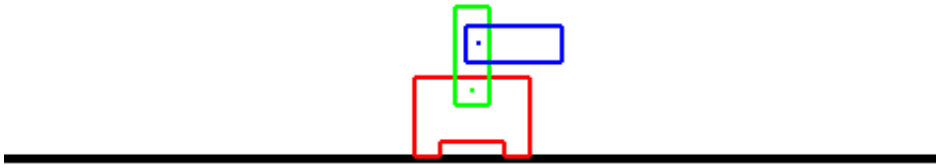
Muitas aplicaçoes utilizam modelos ou cenas para descrever graficamente entidades com propriedades geométricas herdadas. Uma estrutura hierárquica, por exemplo, pode ser introduzida por um processo de construção bottom-up: um conjunto de transformações geométricas é aplicado em um objeto de nível mais alto, que, por sua vez, é composto por outros objetos que serão influenciados por estas transformações, e assim sucessivamente. O corpo humano é um exemplo clássico de hierarquia de objetos. Por exemplo, um movimento de rotação do ombro obrigatoriamente causa um movimento de braço. Da mesma maneira, se movimentarmos o braço, a mão irá se movimentar também. Para manipular objetos hierarquicos em OpenGL é necessário acumular as transformações geométricas que vão sendo aplicadas, de maneira a combiná-las para alcançar o resultado esperado. Para isto, é fundamental compreender o escopo das transformações geométricas que pode ser manipulado com as seguintes funções:

```
void glPushMatrix(void);
Função utilizada para guardar a matriz de transformação corrente na pilha.
void glPopMatrix(void);
Função utilizada para recuperar a matriz de transformação corrente na pilha.
```

O pedaço de código abaixo exemplifica a utilização de transformações hierárquicas. Neste são construídos três objetos: uma base, modelada utilizando a função denominada de DesenhaBase(), e dois braços, ambos modelados pela função DesenhaBraco().

```
glTranslatef(tx,0.0f,0.0f);
glPushMatrix();
                    glScalef(2.5f,2.5f,1.0f);  // aplica transformação de escala apenas na base
                    glColor3f(1.0f,0.0f,0.0f);
                    DesenhaBase();
glPopMatrix();
glTranslatef(-3.0f,1.5f,0.0f);
glRotatef(ang1,0.0f,0.0f,1.0f);
glPushMatrix();
                    glScalef(1.4f,1.4f,1.0f); // aplica transformação de escala apenas em um dos braços
                    glColor3f(0.0f,1.0f,0.0f);
                    DesenhaBraco();
glPopMatrix();
glTranslatef(-5.4f,0.5f,0.0f);
glRotatef(ang2,0.0f,0.0f,1.0f);
glColor3f(0.0f,0.0f,1.0f);
DesenhaBraco();
```

A figura abaixo representa o resultado alcançado, utilizando o código apresentado acima.



Tarefas

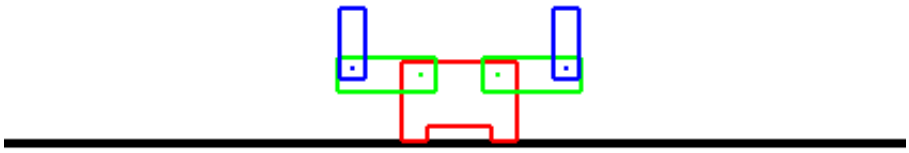
Crie um projeto para o arquivo **ObjetoComTransformacoesHierarquicas.cpp**, compile e analise a hierarquia utilizada para aplicação das transformações.
Modifique o código para que este apresente agora dois braços articulados,conforme a figura abaixo.
Utilizando os conceitos apresentados anteriormente sobre a implementação de menus em OpenGL: construa um menu para o controle de movimento dos braços. O menu poderá conter quatro opções:

- Braço direito
- Antebraço direito
- Braço esquerdo
-Antebraço esquerdo

ou ainda este poderá ser implementado utilizando sub-menus para definir entre antebraço ou braço:

- Braço direito
    - Braço
    - Antebraço
- Braço esquerdo
    - Braço
    - Antebraço

Utilize as funções GLUT_KEY_LEFT (movimentar o objeto completo para esquerda), GLUT_KEY_RIGHT (movimentar o objeto completo para direita), GLUT_KEY_UP (girar o braço selecionado no sentido anti-horário) e GLUT_KEY_DOWN (girar o braço selecionado no sentido horário);



Referências

- http://www.opengl.org
- Mason Woo et al, OpenGL Programming Guide (3rd Edition), Addison-Wesley, ISBN 0201604582. ("Red Book").