Aula Prática 3 (OpenGL)

Disciplina: Computação Gráfica
Professora: Deller James Ferreira

1)Compile, execute e observe o programa a seguir:

```
/* * quad.cc - Cumulative 2D transformations */
#include <GL/gl.h> // Header File For The OpenGL Library
#include <GL/glu.h> // Header File For The GLu Library
#include <GL/glut.h> // Header File For The GLut Library
#include <stdlib.h>
void draw(){
// Make background colour yellow
glClearColor( 100, 100, 0, 0 );
glClear ( GL_COLOR_BUFFER_BIT );
// modelview matrix for modeling transformations
glMatrixMode(GL_MODELVIEW);
// x-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.5,0.0);
glEnd();
// y-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.0,0.5);
glEnd();

// RED rectangle
glColor3f( 1, 0, 0 );
glRectf(0.1,0.2,0.4,0.3);
// Translate GREEN rectangle
glColor3f( 0, 1, 0 );
glTranslatef(-0.4, -0.1, 0.0);
glRectf(0.1,0.2,0.4,0.3);
// Rotate and translate BLUE rectangle
```

```
glColor3f( 0, 0, 1 );
glRotatef(90, 0.0, 0.0,1.0);
glRectf(0.1,0.2,0.4,0.3);
// Scale, rotate and translate MAGENTA rectangle
glColor3f( 1, 0, 1 );
glScalef(-0.5, 1.0, 1.0);
glRectf(0.1,0.2,0.4,0.3);
// display rectangles
glutSwapBuffers();
} // end of draw()

// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
if(key==27) exit(0);
}
int main(int argc, char ** argv)
{
glutInit(&argc, argv);
// Double Buffered RGB display
glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
// Set window size
glutInitWindowSize( 500,500 );
glutCreateWindow("Rectangles moving around: CUMULATIVE 2D
transformations");
// Declare the display and keyboard functions
glutDisplayFunc(draw);
glutKeyboardFunc(keyboard);
// Start the Main Loop
glutMainLoop();
return 0;
}
```

2) **Compile, execute e observe o programa a seguir:**

```
/* * quad.cc - Non-cumulative 2D transformations */
#include <GL/gl.h> // Header File For The OpenGL Library
#include <GL/glu.h> // Header File For The GLu Library
#include <GL/glut.h> // Header File For The GLut Library
#include <stdlib.h>
void draw(){
// Make background colour yellow
glClearColor( 100, 100, 0, 0 );
glClear ( GL_COLOR_BUFFER_BIT );
```

```
// modelview matrix for modeling transformations
glMatrixMode(GL_MODELVIEW);
// x-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.5,0.0);
glEnd();
// y-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.0,0.5);
glEnd();

// RED rectangle
glColor3f( 1, 0, 0 );
glRectf(0.1,0.2,0.4,0.3);
// Translate GREEN rectangle
glColor3f( 0, 1, 0 );
glTranslatef(-0.4, -0.1, 0.0);
glRectf(0.1,0.2,0.4,0.3);
// Rotate BLUE rectangle
glColor3f( 0, 0, 1 );
glLoadIdentity(); // reset the modelview matrix
glRotatef(90, 0.0, 0.0,1.0);
glRectf(0.1,0.2,0.4,0.3);
// Scale MAGENTA rectangle
glColor3f( 1, 0, 1 );
glLoadIdentity(); // reset the modelview matrix
glScalef(-0.5, 1.0, 1.0);
glRectf(0.1,0.2,0.4,0.3);
// display rectangles
glutSwapBuffers();
} // end of draw()

// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
if(key==27) exit(0);
}
int main(int argc, char ** argv)
{
glutInit(&argc, argv);
// Double Buffered RGB display
glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
// Set window size
```

```
glutInitWindowSize( 500,500 );

glutCreateWindow("Rectangles moving around: NON-CUMULATIVE 2D
transformations");
// Declare the display and keyboard functions
glutDisplayFunc(draw);
glutKeyboardFunc(keyboard);
// Start the Main Loop
glutMainLoop();
return 0;
}
```

3) **Compile, execute e observe o programa a seguir:**

```
/* * quad.cc - Stack-cumulative 2D transformations  */
#include <GL/gl.h> // Header File For The OpenGL Library
#include <GL/glu.h> // Header File For The GLu Library
#include <GL/glut.h> // Header File For The GLut Library
#include <stdlib.h>
void draw(){
// Make background colour yellow
glClearColor( 100, 100, 0, 0 );
glClear ( GL_COLOR_BUFFER_BIT );
// modelview matrix for modeling transformations
glMatrixMode(GL_MODELVIEW);
// x-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.5,0.0);
glEnd();
// y-axis
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2f(0.0,0.0);
glVertex2f(0.0,0.5);
glEnd();

// RED rectangle
glColor3f( 1, 0, 0 );
glRectf(0.1,0.2,0.4,0.3);
// Translate GREEN rectangle
glColor3f( 0, 1, 0 );
glTranslatef(-0.4, -0.1, 0.0);
glRectf(0.1,0.2,0.4,0.3);
// save modelview matrix on the stack
glPushMatrix();
```

```
// Rotate BLUE rectangle
glColor3f( 0, 0, 1 );
glRotatef(90, 0.0, 0.0,1.0);
glRectf(0.1,0.2,0.4,0.3);
// restore modelview matrix from the stack
glPopMatrix();
// Scale and translate MAGENTA rectangle
glColor3f( 1, 0, 1 );
glScalef(-0.5, 1.0, 1.0);
glRectf(0.1,0.2,0.4,0.3);
// display rectangles
glutSwapBuffers();
} // end of draw()

// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
if(key==27) exit(0);
}
int main(int argc, char ** argv)
{
glutInit(&argc, argv);
// Double Buffered RGB display
glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
// Set window size
glutInitWindowSize( 500,500 );
glutCreateWindow("Rectangles moving around: STACK-CUMULATIVE 2D
transformations");
// Declare the display and keyboard functions
glutDisplayFunc(draw);
glutKeyboardFunc(keyboard);
// Start the Main Loop
glutMainLoop();
return 0;
}
```

4) Implemente um programa que faça o espelhamento de um objeto com o eixo que possui inclinação de 45 graus com relação ao eixo X. Desenhe o objeto antes e depois da transformação geométrica.

5) Implemente um programa que faça a escala de um quadrado em função de seu centro.

6) Implemente um programa que faça a rotação de um objeto em torno de um ponto de pivot utilizando composição de matrizes.