Aula Prática 4 (OpenGL)

Disciplina: Computação Gráfica
Professora: Deller James Ferreira

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

```
#include<GL/glut.h>
#include<stdlib.h>
#include<math.h>
/* Janela inicial */
GLsizei  winWidth = 600,  winHeight = 600;
/* limites para as coordenadas do mundo */
GLfloat xwcMin = 0.0, xwcMax = 255.0;
GLfloat  ywcMin = 0.0, ywcMax = 255.0;
class wcPt2D {
  public:
       GLfloat x, y;
};
typedef GLfloat Matrix3x3  [3][3];
Matrix3x3 matComposite;
const GLdouble pi = 3.14158;
void init (void)
{
```

```
/* determina cor da janela de branca*/


glClearColor (1.0,1.0,1.0,1.0);

}

/*Construa matriz identidade */

void matrix3x3SetIdentity (Matrix3x3 matIdent3x3)


{

GLint row, col;

for (row=0;row<3;row++)

    for (col=0;col<3;col++)

         matIdent3x3 [row][col] = (row == col);

}

/*multiplique matriz  m1 por matriz m2 e coloque o resultado em m2 */

void matrix3x3PreMultiply (Matrix3x3  m1, Matrix3x3 m2)

{

GLint row, col;

Matrix3x3 matTemp;

for (row=0;row<3;row++)

    for (col=0;col<3;col++)

        matTemp  [row][col] = m1 [row][0] * m2 [0][col] + m1[row][1]*m2[1][col] + m1[row][2] *m2[2]
[col];

for (row=0;row<3;row++)

    for (col=0;col<3;col++)

       m2 [row][col] =matTemp[row][col];

}
```

```
void translate2D (GLfloat tx, GLfloat ty)
{
 Matrix3x3 matTrans1;
/*Inicialize a matriz de transformacao para a identidade*/
matrix3x3SetIdentity(matTrans1);
matTrans1[0][2] = tx;
matTrans1[1][2] = ty;
/*Concatene a matriz matTrans1 com a matriz de composição*/
matrix3x3PreMultiply(matTrans1, matComposite);
}
void rotate2D ( wcPt2D pivotPt, GLfloat theta)
{
Matrix3x3 matRot;

/*Inicialize a matriz derotacao para a identidade*/
matrix3x3SetIdentity(matRot);
matRot [0][0] = cos(theta);
matRot [0][1] = -sin(theta);
matRot [0][2] = pivotPt.x * (1- cos(theta)) + (pivotPt.y * sin(theta));
matRot [1][0] = sin(theta);
matRot [1][1] = cos(theta);
matRot [1][2] = pivotPt.y * (1- cos(theta)) - (pivotPt.x * sin(theta));
/*Concatene a matriz matRot com a matriz de composição*/
matrix3x3PreMultiply(matRot, matComposite);
```

```
}


void scale2D (GLfloat sx, GLfloat sy, wcPt2D fixedPt)
{
 Matrix3x3 matScale;
/*Inicialize a matriz de transformacao para a identidade*/
matrix3x3SetIdentity(matScale);
matScale[0][0] = sx;
matScale[0][2] = (1 - sx) * fixedPt.x;
matScale[1][1] = sy;
matScale[1][2] = (1 - sy) * fixedPt.y;

/*Concatene a matriz matScale com a matriz de composição*/
matrix3x3PreMultiply(matScale, matComposite);
}
/*Use a matriz composta para calcular as transformadas*/
void transformVerts2D (GLint nVerts, wcPt2D * verts)
{
GLint k;
GLfloat temp;
for ( k=0;k<nVerts;k++){
   temp = matComposite [0][0] * verts[k].x + matComposite [0][1] * verts[k].y + matComposite [0][2] ;
   verts [k].y = matComposite [1][0] * verts[k].x + matComposite [1][1] * verts[k].y + matComposite [1]
[2] ;
   verts[k].x = temp;
}
```

```
}
void  triangle (wcPt2D *verts)
{
GLint k;
glBegin (GL_TRIANGLES);
for ( k=0;k<3;k++)
      glVertex2f (verts[k].x, verts[k].y);
glEnd ();
}
void displayFcn (void)
{

/*define a posição inicial do triangulo*/
GLint nVerts = 3;
wcPt2D  verts [3] = {{50.0,25.0}, {150.0,25.0}, {100.0,100.0}};
/*calcula  a posição do centróide do triangulo*/
wcPt2D centroidPt;
GLint k;
GLfloat xSum = 0, ySum =0;
for ( k=0; k < nVerts; k++){
xSum += verts[k].x;
ySum += verts[k].y;
}
centroidPt.x = xSum / GLfloat (nVerts);
centroidPt.y = ySum / GLfloat (nVerts);
```

```
/*Estabelece parâmetros da transformação geométrica*/
wcPt2D pivPt, fixedPt;
pivPt = centroidPt;
fixedPt = centroidPt;
GLfloat tx = 0.0, ty = 100.0;
GLfloat sx = 0.5, sy = 0.5;
GLdouble theta = pi/2.0;
glClear (GL_COLOR_BUFFER_BIT);   //limpa a janela de visão (display window)
glColor3f (0.0,0.0,1.0); //estalebece a cor de preenchimento inicial com o azul
triangle(verts); //Exibe triângulo
/*inicializa a matriz de composição para a identidade*/
matrix3x3SetIdentity (matComposite);
/*Constrói a matriz de composição para a sequência de  transformações*/
scale2D(sx,sy,fixedPt);
rotate2D(pivPt,theta);
translate2D(tx,ty);
/*Aplica a matriz de transformação nos vértices do triângulo*/
transformVerts2D(nVerts, verts);
glColor3f(1.0,0.0,0.0);
triangle(verts); // Exibe triângulo transformado
glFlush();
}
void winReshapeFcn (GLint newWidth, GLint newHeight)
{
glMatrixMode (GL_PROJECTION);
```

```
glLoadIdentity();
gluOrtho2D (xwcMin, xwcMax,ywcMin, ywcMax);
glClear (GL_COLOR_BUFFER_BIT);
}
int main(int argc, char ** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowSize(winWidth,winHeight);
   glutInitWindowPosition(50,50);
   glutCreateWindow("Composicao de Transformacoes");
   init();
   glutDisplayFunc(displayFcn);
   glutReshapeFunc(winReshapeFcn);
  glutMainLoop();
}
```

2. Modifique o programa anterior para realizar a escala de um quadrado em função de seu centro.

3. Modifique o programa anterior para rotacionar um quadrado em torno de um de seus vértices.