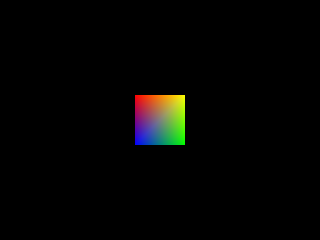
**Tutorial 02 - Matrizes e coloração de poligonos.**



Neste tutorial, vamos configurar um sistema de coordenadas de 640x480 e, enquanto estivermos nisso, vamos dar alguma cor ao nosso polígono.

**if(!glfwInit()){**

**System.err.println("Falha ao inicializar GLFW!");**

**System.exit(1);**

**}**

Essa função glfwInit() inicializa a biblioteca GLFW.

Antes que a maioria das funções do GLFW possa ser usada, o GLFW deve ser inicializado

E antes de um aplicação terminar, o GLFW deve ser encerrado, a fim de liberar quaisquer recursos alocados durante ou após a inicialização.

Se esta função falhar, chama glfwTerminate antes de retornar. Se for bem-sucedida, você deve chamar o glfwTerminate antes que o aplicativo seja encerrado.

Retorna TRUE se for bem-sucedida, e FALSE se ocorrer algum erro

Caso glfwInit() retornar FALSE, mostra na tela a mensagem "Falha ao inicializar GLFW!".  System.exit(int) encerra o programa, o parametro int quando for diferente de 0 indica que o programa foi finalizado devido a algum erro.

**long win = glfwCreateWindow(640, 480, "Window", 0, 0);**

**Função e parâmetros:**

**glfwCreateWindow ( int width, int height, const char \* title, GLFWmonitor \* monitor,**

**GLFWwindow \*share)**

Essa função cria uma janela e com seu contexto do OpenGL associado.

Antes que você possa começar a desenhar coisas com o OpenGL, precisa inicializa-lo. Isso é feito pela criação de um contexto, que é essencialmente um estado de máquina que armazena todos os dados relacionados à renderização de sua aplicação. Quando a sua aplicação é fechada, o contexto é destruído e tudo que ele armazena é apagado da memória.

Parâmetros

width :  A largura desejada, em coordenadas de tela, da janela. Isso deve ser maior que zero.

height:  A altura desejada, em coordenadas de tela, da janela. Isso deve ser maior que zero.

title:   O título da janela inicial.

monitor: O monitor a utilizar para o modo de tela cheia, ou NULL para o modo de janela.

share:   A janela cujo contexto para compartilhar recursos com, ou NULL para não compartilhar recursos.

**glfwMakeContextCurrent(win);**

A criação bem-sucedida da janela não muda o contexto atual. Antes de poder usar o contexto recém-criado, você precisa torná-lo atual. Para isso utilizamos essa função.

**glfwMakeContextCurrent(win);**

**Função e parâmetros:**

**void glfwShowWindow ( GLFWwindow \* window )**

Essa função torna a janela especificada visível se ela estava anteriormente oculta. Se a janela já estiver visível ou estiver no modo de tela cheia, esta função não faz nada.

Parâmetros

window: A janela para tornar visível.

**GL.createCapabilities();**

Para que LWJGL realmente saiba sobre o contexto OpenGL e se inicialize usando esse contexto, temos que chamar GL.createCapabilities ()

**while(!glfwWindowShouldClose(win)){**

**glfwPollEvents();**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**//limpa todos os pixels para a cor preta**

**glViewport( 0, 0, 640, 480 );**

**glBegin(GL\_QUADS);**

**glColor4f(1,0,0,0);**

**glVertex2f(-0.5f, 0.5f);**

**glColor4f(0,1,0,0);**

**glVertex2f(0.5f, 0.5f);**

**glColor4f(0,0,1,0);**

**glVertex2f(0.5f, -0.5f);**

**glColor4f(0,0,0,1);**

**glVertex2f(-0.5f, -0.5f);**

**glEnd();**

**glfwSwapBuffers(win);**

**//Precisa de 2 contextos(buffers) para trocar de um para outro, um é mostrado na tela, enquanto o OpenGL desenha no outro**

**}**

**while(!glfwWindowShouldClose(win))**

**//Loop até que o usuário feche a janela**

**glfwPollEvents();**

O GLFW precisa se comunicar regularmente com o sistema de janelas tanto para receber eventos quanto para mostrar que o aplicativo não está bloqueado. O processamento de eventos deve ser feito regularmente enquanto você tem janelas visíveis e normalmente é feito em cada frame após o buffer swapping.

Há dois métodos para processar eventos pendentes; polling ou waiting. Este exemplo usará polling de eventos, que processa apenas os eventos que já foram recebidos e, em seguida, retorna imediatamente.

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**//Limpa todos os pixels para a cor preta**

**glViewport( 0, 0, 640, 480 );**

**//Função e parâmetros:**

**void glViewport( GLint x, GLint y, GLsizei width, GLsizei height);**

glViewport: Define a janela de visualização

x, y: Especifica o canto inferior esquerdo do retângulo da janela de visualização, em pixels. O valor inicial é (0,0).

width, height: Especifica a largura e a altura da janela de visualização. Quando um contexto GL é primeiro anexado a uma janela, largura e altura são definidas para as dimensões dessa janela.

**glBegin(GL\_QUADS);**

**glColor4f(1,0,0,0);**

**glVertex2f(-0.5f, 0.5f);**

**glColor4f(0,1,0,0);**

**glVertex2f(0.5f, 0.5f);**

**glColor4f(0,0,1,0);**

**glVertex2f(0.5f, -0.5f);**

**glColor4f(0,0,0,1);**

**glVertex2f(-0.5f, -0.5f);**

**glEnd();**

glBegin e glEnd: delimitam os vértices de um primitivo ou um grupo de primitivos semelhantes

GL\_QUADS: Trata cada grupo de quatro vértices como um quadrilátero independente.

glVertex2f: Especifica um ponteiro para uma matriz de dois elementos. Os elementos de um array de dois elementos são x e y

glColor4f: Especifique novos valores de vermelho, verde, azul e alpha para a cor atual.

**glfwSwapBuffers(win);**

É necessário 2 contextos(buffers) para trocar de um para outro durante o desenho, enquanto um é mostrado na tela, o OpenGL desenha no outro. Essa função realiza essa troca de buffers, passando como parâmetro a janela cujos buffers são trocados, fazendo com que a imagem da tela atualize.

**glfwTerminate();**

Esta função destrói todas as janelas e cursores restantes e libera quaisquer outros recursos alocados.

--------------- Código ---------------

import static org.lwjgl.glfw.GLFW.\*;

import static org.lwjgl.opengl.GL11.\*;

import org.lwjgl.glfw.GLFWKeyCallback;

import org.lwjgl.opengl.GL;

import input.Input;

public class Tutorial02 {

final int SCREEN\_WIDTH = 640;

final int SCREEN\_HEIGHT = 480;

int COLOR\_MODE\_CYAN = 0;

int COLOR\_MODE\_MULTI = 1;

int gColorMode = COLOR\_MODE\_CYAN;

float gProjectionScale = 1.f;

private GLFWKeyCallback keyCallback;

public Tutorial02() {

        if (!glfwInit()) {

            System.err.println("Falha ao inicializar GLFW!");

            System.exit(1);

        }

        long win = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "Window", 0, 0);

        //Primeiro 0 serve para permitir que mudemos a tela para Tela cheia. (Caso queira, substituir 0 por glfwGetPrimaryMonitor()

       glfwShowWindow(win);

        glfwMakeContextCurrent(win);

        //Cria o contexto e permite que o open gl desenhe nele

        GL.createCapabilities();

        initGL();

       glfwSetKeyCallback(win, keyCallback = new Input());

       while (!glfwWindowShouldClose(win)) {

            glfwPollEvents();

            glClear(GL\_COLOR\_BUFFER\_BIT);

            glMatrixMode( GL\_MODELVIEW );

            glLoadIdentity();

            glTranslatef( SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f, 0.f );

            if( gColorMode == COLOR\_MODE\_CYAN )

            {

            //Cria um quadrado com a cor  CYAN

            glBegin( GL\_QUADS );

            glColor3f( 0.f, 1.f, 1.f );

            glVertex2f( -50.f, -50.f );

            glVertex2f(  50.f, -50.f );

            glVertex2f(  50.f,  50.f );

            glVertex2f( -50.f,  50.f );

            glEnd();

            }

            else

            {

                //Cria um qudado MULTICOLOR

                glBegin( GL\_QUADS );

                    glColor3f( 1.f, 0.f, 0.f ); glVertex2f( -50.f, -50.f );

                    glColor3f( 1.f, 1.f, 0.f ); glVertex2f(  50.f, -50.f );

                    glColor3f( 0.f, 1.f, 0.f ); glVertex2f(  50.f,  50.f );

                    glColor3f( 0.f, 0.f, 1.f ); glVertex2f( -50.f,  50.f );

                glEnd();

            }

      if(glfwGetKey(win, GLFW\_KEY\_Q) == GL\_TRUE)

        {

        //Muda a cor entre CYAN e MULTICOLOR

        if( gColorMode == COLOR\_MODE\_CYAN )

        {

                  gColorMode = COLOR\_MODE\_MULTI;

        }

               else

               {

                   gColorMode = COLOR\_MODE\_CYAN;

               }

        }

        else if( glfwGetKey(win, GLFW\_KEY\_E) == GL\_TRUE )

        {

            //Muda entre as escalas de projeções

            if( gProjectionScale == 1.f )

            {

                //Zoom out

                gProjectionScale = 2.f;

            }

            else if( gProjectionScale == 2.f )

            {

                //Zoom in

                gProjectionScale = 0.5f;

            }

            else if( gProjectionScale == 0.5f )

            {

                //Regular zoom

                gProjectionScale = 1.f;

            }

            //Atualiza a matriz de projeção

            glMatrixMode( GL\_PROJECTION );

            glLoadIdentity();

            glOrtho( 0.0f, SCREEN\_WIDTH \* gProjectionScale, SCREEN\_HEIGHT \* gProjectionScale, 0.0f, 1.0f, -1.0f );

    }

        //Precisa de 2 contextos(buffers) para trocar de um para outro, um é mostrado na tela, enquanto o OpenGL desenha no outro

        glfwSwapBuffers(win);

    }

            }

boolean initGL() {

//Define o viewport

   glViewport( 0, 0, SCREEN\_WIDTH, SCREEN\_HEIGHT);

//Initialize Projection Matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho( 0.0, SCREEN\_WIDTH, SCREEN\_HEIGHT, 0.0, 1.0, -1.0 );

//Inicializa a Matriz Modelview

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//Especifica valores para os buffers de cores

glClearColor(0.f, 0.f, 0.f, 1.f);

//Checa possíveis erros

int error = glGetError();

if (error != GL\_NO\_ERROR) {

System.err.println("Falha ao inicializar OpenGL  initGL");

        return false;

}

return true;

}

public static void main(String[] args) {

new Tutorial02()

} }