# Primeiro polígono

Este primeiro tutorial tem como objetivo, da o passo inicial, obtendo algo na tela.

Agora que você já tenha configurado o eclipse e lwjl3, é hora de fazer uma aplicação gráfica OpenGL que processa um quadrado na tela.

|  |  |
| --- | --- |
| 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. |  |
|  |  |

if (!GLFW.glfwInit()) {

System.exit(1);

}

--------------------

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.

long win = glfwCreateWindow(640,420, "window", 0,0);

---------------------

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.

glfwShowWindow(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);

-------------------------------

Para que LWJGL realmente saiba sobre o contexto OpenGL e se inicialize usando esse contexto, temos

que chamar GL.createCapabilities ()

GL.createCapabilities();

----------------------

Loop até que o usuário feche a janela

while(!glfwWindowShouldClose(win))

--------------------

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.

glfwPollEvents();

---------

Limpa todos os pixels para a cor preta

glClear(GL\_COLOR\_BUFFER\_BIT);

---------------------

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.

glBegin(GL\_QUADS);

glColor4f(1, 0, 0,0);

glVertex2f( -0.5f, -0.5f );

glVertex2f( 0.5f, -0.5f );

glVertex2f( 0.5f, 0.5f );

glVertex2f( -0.5f, 0.5f );

glEnd();

----------------------------

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.

glfwSwapBuffers(win);

---------------

Depois de terminarmos o GLFW, devemos destruí-lo.

GLFW.glfwTerminate();

-----------Código completo-------------

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

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

import org.lwjgl.glfw.GLFW;

import org.lwjgl.opengl.GL;

class Tutorial01{

public Tutorial01() {

if (!GLFW.glfwInit()) {

System.exit(1);

}

long win = glfwCreateWindow(640,420, "window", 0,0);

glfwShowWindow(win);

glfwMakeContextCurrent(win);

GL.createCapabilities();

while(!glfwWindowShouldClose(win)){

glfwPollEvents();

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_QUADS);

glColor4f(1, 0, 0,0);

glVertex2f( -0.5f, -0.5f );

glVertex2f( 0.5f, -0.5f );

glVertex2f( 0.5f, 0.5f );

glVertex2f( -0.5f, 0.5f );

glEnd();

glfwSwapBuffers(win);

}

GLFW.glfwTerminate();

}

public static void main(String[] args) {

new Tutorial01();

}

}