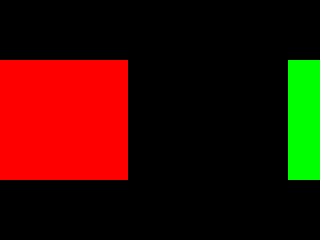
**Tutorial 04 - Percorrendo e pilha de matrizes**



Em muitos, se não a maioria dos jogos, os ambientes de jogo são maiores do que uma tela. Aqui vamos aprender a percorrer grandes ambientes.

Ao lidar com grandes ambientes, você precisa de algum tipo de câmera para definir a área que você está fazendo. Aqui vamos salvar as transformações para a matriz modelview para fazer isso.

Neste tutorial, vamos aplicar transformação de translação para a matriz modelview para percorrer o ambiente. Desta vez, no entanto, ao invés de chamar glLoadIdentity () e glOrtho () todas as vezes que fizemos no tutorial Matrizes e Colorir Polígonos, vamos empurrar uma cópia da matriz inicial do modelo percorrendo para a posição da câmera na pilha para salvá-la para quando precisarmos aplicar transformações nela.

**float gCameraX = 0.f, gCameraY = 0.f;**

No começo do código definimos coordenadas x / y para a nossa câmera. A função initGL() é praticamente a mesma, só agora há uma chamada para glPushMatrix().

Uma vez que as transformações geométricas no espaço são representadas por matrizes, o uso de uma pilha de matrizes de transformação ajuda a lembrar a seqüência de transformações realizadas. No OpenGL, esta facilidade é provida pelas funções glPushMatrix(), que insere a matriz de transformação corrente na pilha, e glPopMatrix(), que retira a matriz do topo da pilha e torna esta última a matriz de transformação corrente.

**//Tire a matriz guardada da pilha e reseta ela**

**glMatrixMode( GL\_MODELVIEW );**

**glPopMatrix();**

**glLoadIdentity();**

**//Move a camera para a posição indicada**

**glTranslatef( -gCameraX, -gCameraY, 0.f );**

**//Salva novamente a matriz padrão com a mudança de translação na câmera**

**glPushMatrix();**

Desde que mudamos a posição da câmera quando o usuário pressionou uma tecla, precisamos mudar nossa matriz de câmera padrão.

Primeiro, removemos a antiga matriz padrão fora da pilha para a matriz atual com glPopMatrix(). Em seguida, carregamos a matriz de identidade na matriz atual de modelview. Depois disso, traduzimos a matriz de modelview pelos deslocamentos da câmera, de modo que tudo se tornará relativo à câmera.

Como removemos a matriz padrão da pilha, precisamos colocar a nossa nova na parte superior da pilha para que possamos guardá-la para mais tarde

**//Limpa as cores do buffer**

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

**//Inicializa a Matriz Modelview e carrega a matriz que salvamos com a translação da câmera**

**glMatrixMode( GL\_MODELVIEW );**

**glPopMatrix();**

**//Salva a matriz padrão novamente**

**glPushMatrix();**

Em vez de usar glLoadIdentity() para redefinir a matriz de modelview, usaremos glPopMatrix() para carregar a matriz que guardamos com a translação da câmera. Como precisamos dessa matriz de modelview padrão no próximo quadro, imediatamente o empurramos para a pilha para salvá-lo para mais tarde.

Agora que a matriz modelview torna tudo relativo à câmera, podemos começar a renderizar nossa geometria

**if( glfwGetKey(win, GLFW\_KEY\_W) == GL\_TRUE ){**

**gCameraY -= 0.5f;**

**}else if( glfwGetKey(win, GLFW\_KEY\_S) == GL\_TRUE ){**

**gCameraY += 0.5f;**

**}else if( glfwGetKey(win, GLFW\_KEY\_A) == GL\_TRUE ){**

**gCameraX -= 0.5f;**

**}else if( glfwGetKey(win, GLFW\_KEY\_D) == GL\_TRUE ){**

**gCameraX += 0.5f;**

**}**

Move de posição de renderização da câmera quando o usuário pressiona w/a/s/d

**//Move para o centro da tela**

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

**//Quadrado vermelho**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glEnd();**

**//Move para a direita da tela**

**glTranslatef( SCREEN\_WIDTH, 0.f, 0.f );**

**//Quadrado verde**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glEnd();**

**//Move para a parte inferior direita da tela**

**glTranslatef( 0.f, SCREEN\_HEIGHT, 0.f );**

**//Quadrado azul**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glEnd();**

**//Move a tela para baixo**

**glTranslatef( -SCREEN\_WIDTH, 0.f, 0.f );**

**//Quadrado amarelo**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

**glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );**

**glEnd();**

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

**glfwSwapBuffers(win);**

Renderizamos uma cena duas vezes a largura / altura da tela. Nossa geometria nunca muda de posição, apenas a câmera muda por toda parte do exemplo.

--------------- 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 Tutorial04 {

final int SCREEN\_WIDTH = 640;

final int SCREEN\_HEIGHT = 480;

float gCameraX = 0.f, gCameraY = 0.f;

private GLFWKeyCallback keyCallback;

public Tutorial04() {

           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 );

            glPopMatrix();

            //Salva a matrix padrão novamente

            glPushMatrix();

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

        //Quadrado vermelho

            glBegin( GL\_QUADS );

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

                glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

                glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

            glEnd();

            //Move para a direita da tela

            glTranslatef( SCREEN\_WIDTH, 0.f, 0.f );

            //Quadrado verde

            glBegin( GL\_QUADS );

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

                glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

                glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

            glEnd();

            //Move para a parte inferior direita da tela

            glTranslatef( 0.f, SCREEN\_HEIGHT, 0.f );

            //Quadrado azul

            glBegin( GL\_QUADS );

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

                glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

                glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

            glEnd();

            //Move a tela para baixo

            glTranslatef( -SCREEN\_WIDTH, 0.f, 0.f );

            //Quadrado amarelo

            glBegin( GL\_QUADS );

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

                glVertex2f( -SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );

                glVertex2f(  SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

                glVertex2f( -SCREEN\_WIDTH / 4.f,  SCREEN\_HEIGHT / 4.f );

            glEnd();

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

           glfwSwapBuffers(win);

           //Move de posição de renderização da câmera quando o usuário pressiona w/a/s/d

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

            gCameraY -= 0.5f;

            }else if( glfwGetKey(win, GLFW\_KEY\_S) == GL\_TRUE ){

            gCameraY += 0.5f;

           }else if( glfwGetKey(win, GLFW\_KEY\_A) == GL\_TRUE ){

            gCameraX -= 0.5f;

            }else if( glfwGetKey(win, GLFW\_KEY\_D) == GL\_TRUE ){

            gCameraX += 0.5f;

            }

            //Tire a matriz guardada da pilha e reseta ela

            glMatrixMode( GL\_MODELVIEW );

            glPopMatrix();

            glLoadIdentity();

            //Move a camera para a posição indicada

            glTranslatef( -gCameraX, -gCameraY, 0.f );

            //Salva novamente a matriz padrão com a mudança de translação na câmera

            glPushMatrix();

        }

            }

boolean initGL() {

//Define o viewport

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

//Inicializa a Matriz de projeção

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();

//Salva o modelo padrão da matriz

   glPushMatrix();

//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 Tutorial04(); }}