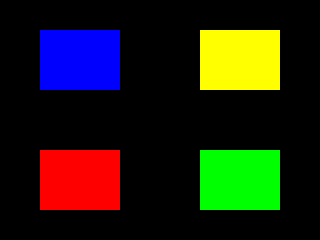
**O Viewport**

****

Então, nós sabemos como definir o nosso sistema de coordenadas usando a matriz de projeção, mas e se apenas quisermos renderizar parte da tela? Aqui é onde a janela de visualização entra.

A viewport define a área retangular da tela que queremos renderizar. Aqui vamos jogar com ele para fazer coisas como a renderização em tela dividida.

**A partir de LWJGL:**

**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;**

Só para que você não fique confuso quando chegamos ao código de renderização, temos algumas constantes enumeradas que definem as diferentes maneiras de usar a viewport.

Aqui está o topo do nosso arquivo LUTIL.cpp. Próximo ao topo, temos a variável global "gViewportMode", que define como vamos usar a viewport.

Nossa função initGL () é praticamente a mesma de antes, mas agora ele tem uma chamada para glViewport () para inicializar a viewport. GlViewport () define que parte da tela queremos renderizar definindo a coordenada x, a coordenada y, a largura ea altura da área de renderização. Como você pode ver aqui, estamos apenas dizendo para renderizar a tela inteira.

**O tutorial03.java:**

Aqui está o topo do nosso arquivo tutorial03.java. Próximo ao topo, temos a variável global "gViewportMode", que define como vamos usar a viewport.

**int gViewportMode = VIEWPORT\_MODE\_FULL;**

**boolean initGL() {**

**//Set the viewport**

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

**//Initialize Projection Matrix**

**glMatrixMode( GL\_PROJECTION );**

**glLoadIdentity();**

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

**//Initialize Modelview Matrix**

**glMatrixMode( GL\_MODELVIEW );**

**glLoadIdentity();**

**//Initialize clear color**

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

**//Check for error**

**int error = glGetError();**

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

**{**

**// System.out.printf( "Error initializing OpenGL! %s\n", GLUtil.Er( error ) );**

**return false;**

**}**

**return true;**

}

Nossa função initGL () é praticamente a mesma de antes, mas agora ele tem uma chamada para glViewport () para inicializar a viewport. A função glViewport () define que parte da tela queremos renderizar definindo a coordenada x, a coordenada y, a largura e a altura da área de renderização. Como você pode ver aqui, estamos apenas dizendo para renderizar a tela inteira.

**O tutorial03.java**

**if( gViewportMode == VIEWPORT\_MODE\_FULL){**

**//Fill the screen**

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

**//Red quad**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

**glVertex2f( SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

**glVertex2f( SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glVertex2f( -SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glEnd();**

Em "VIEWPORT\_MODE\_FULL", definimos a janela de visualização como a tela inteira e renderizamos um quadrado em tela cheia.

Pode parecer redundante definir a viewport novamente, mas nesta demonstração vamos estar mudando a viewport dependendo de "gViewportMode".

**else if( gViewportMode == VIEWPORT\_MODE\_HALF\_CENTER )**

**{**

**//Center viewport**

**glViewport( SCREEN\_WIDTH / 4, SCREEN\_HEIGHT / 4, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**//Green quad**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

**glVertex2f( SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

**glVertex2f( SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glVertex2f( -SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glEnd();**

**}**

**O tutorial03.java**

Aqui nós renderizamos o mesmo quadrado 640x480 em uma viewport que é metade da largura / altura da tela no meio de nossa área de renderização.

**Isso resulta em:**



**Nota:** as informações de coordenadas

Assim, as coordenadas de renderização ainda são 640x480 mesmo se a viewport for 320x240.

**O tutorial03.java**

**else if( gViewportMode == VIEWPORT\_MODE\_HALF\_TOP )**

**{**

**//Viewport at top**

**glViewport( SCREEN\_WIDTH / 4, SCREEN\_HEIGHT / 2, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**//Blue quad**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

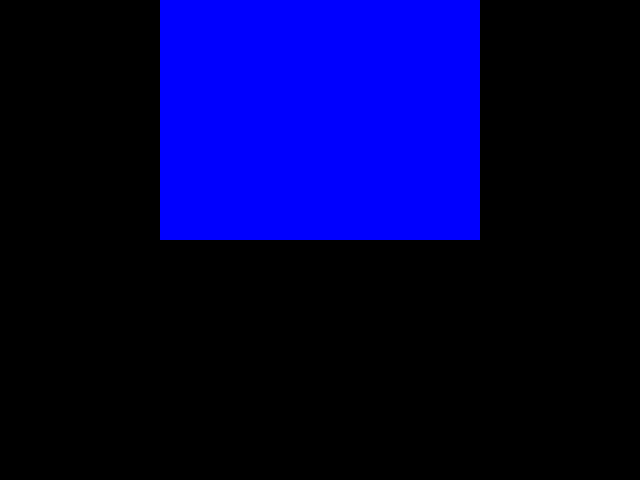
**glVertex2f( SCREEN\_WIDTH / 2.f, -SCREEN\_HEIGHT / 2.f );**

**glVertex2f( SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glVertex2f( -SCREEN\_WIDTH / 2.f, SCREEN\_HEIGHT / 2.f );**

**glEnd(); }**

Aqui está um quadrado de 640x480, renderizado em uma viewport de 320x240, renderizado na posição da janela de visualização x = 80, y = 240. No GLUT, ele sai assim:



Uma coisa importante a observar é que enquanto nossas coordenadas de projeção têm Y + como baixo e Y- como acima, as coordenadas de viewport não são garantidas. No GLUT, Y + está para cima e Y- está para baixo para as coordenadas da viewport. Em alguns sistemas de janelas, Y + está para baixo e Y- é para cima para coordenadas de viewport. Portanto, ao trabalhar com diferentes sistemas de janelas, não assume que as coordenadas da viewport são as mesmas.

**O tutorial03.java**

Aqui temos várias viewports, que é útil para jogos de tela dividida e simulações. O mesmo quad é renderizado 4 vezes, apenas com cores diferentes e locais de viewport.

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

**//Bottom left red quad**

**else if( gViewportMode == VIEWPORT\_MODE\_QUAD )**

**{**

**glViewport( 0, 0, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**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();**

**//Bottom right green quad**

**glViewport( SCREEN\_WIDTH / 2, 0, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**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();**

**//Top left blue quad**

**glViewport( 0, SCREEN\_HEIGHT / 2, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**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();**

**//Top right yellow quad**

**glViewport( SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**glBegin( GL\_QUADS );**

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

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

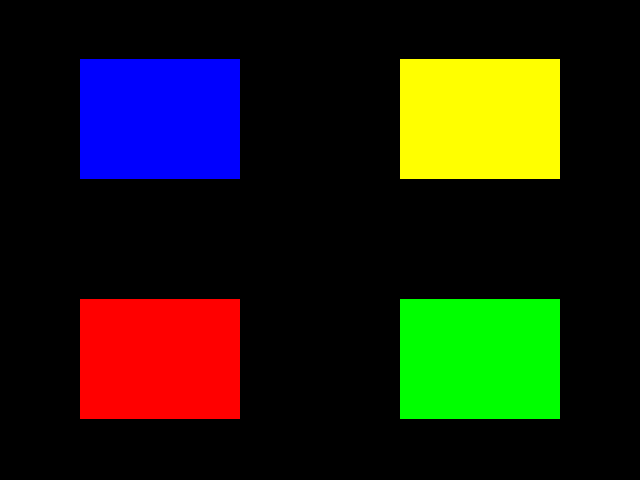
**glVertex2f( SCREEN\_WIDTH / 4.f, -SCREEN\_HEIGHT / 4.f );**

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

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

**glEnd();**

**}**



**O tutorial03.java**

**else if( gViewportMode == VIEWPORT\_MODE\_RADAR )**

**{**

**//Full size quad**

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

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 8.f, -SCREEN\_HEIGHT / 8.f );**

**glVertex2f( SCREEN\_WIDTH / 8.f, -SCREEN\_HEIGHT / 8.f );**

**glVertex2f( SCREEN\_WIDTH / 8.f, SCREEN\_HEIGHT / 8.f );**

**glVertex2f( -SCREEN\_WIDTH / 8.f, SCREEN\_HEIGHT / 8.f );**

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

**glVertex2f( -SCREEN\_WIDTH / 16.f, -SCREEN\_HEIGHT / 16.f );**

**glVertex2f( SCREEN\_WIDTH / 16.f, -SCREEN\_HEIGHT / 16.f );**

**glVertex2f( SCREEN\_WIDTH / 16.f, SCREEN\_HEIGHT / 16.f );**

**glVertex2f( -SCREEN\_WIDTH / 16.f, SCREEN\_HEIGHT / 16.f );**

**glEnd();**

**//Radar quad**

**glViewport( SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2 );**

**glBegin( GL\_QUADS );**

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

**glVertex2f( -SCREEN\_WIDTH / 8.f, -SCREEN\_HEIGHT / 8.f );**

**glVertex2f( SCREEN\_WIDTH / 8.f, -SCREEN\_HEIGHT / 8.f );**

**glVertex2f( SCREEN\_WIDTH / 8.f, SCREEN\_HEIGHT / 8.f );**

**glVertex2f( -SCREEN\_WIDTH / 8.f, SCREEN\_HEIGHT / 8.f );**

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

**glVertex2f( -SCREEN\_WIDTH / 16.f, -SCREEN\_HEIGHT / 16.f );**

**glVertex2f( SCREEN\_WIDTH / 16.f, -SCREEN\_HEIGHT / 16.f );**

**glVertex2f( SCREEN\_WIDTH / 16.f, SCREEN\_HEIGHT / 16.f );**

**glVertex2f( -SCREEN\_WIDTH / 16.f, SCREEN\_HEIGHT / 16.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); }**

E para a última das demonstrações de viewport, renderizamos uma cena de tamanho completo e, em seguida, renderizamos uma versão menor no canto superior esquerdo.

Ter uma viewport dentro de uma viewport pode ser útil para coisas como renderizar um radar na tela.

Claro, no final da nossa função de renderização, atualizamos a tela.

**O tutorial03.java**

**int gViewportMode = VIEWPORT\_MODE\_FULL;**

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

**glfwPollEvents();**

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

**// Input.keys[GLFW\_KEY\_SPACE]**

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

**gViewportMode++;**

**if( gViewportMode > VIEWPORT\_MODE\_RADAR )**

**{**

**gViewportMode = VIEWPORT\_MODE\_FULL;**

**}**

**}**

Por fim, aqui está o nosso manipulador de chaves. Tudo o que faz é percorrer os modos de renderização do viewport quando o usuário pressiona SPACE.