

# LIBGDX TouchPad

## UNIDADE 3: Touchpad

### Sumario

- 1 Introducción
- 2 Proceso de visualización
- 3 Control do Touchpad
- 4 Exemplo de código
- 5 TAREFA OPTATIVA A FACER

### Introdución

**Nota:** Esta explicación está relacionada coa sección de Interfaces.

Clase que se utiliza: [TouchPad](#).

O Touchpad ven ser a representación dun joystick na pantalla.

Desta forma podemos manexar o noso protagonista dunha forma máis cómoda, aínda que dependerá do tipo de xogo e pode ser que non sexa conveniente este tipo de control.

O normal é que sexa o control elixido para manexar unha nave (avión, nave espacial,...) non xogo no que se teña un movemento aberto en todas as direccións.

### Proceso de visualización

Para visualizar o Touchpad imos necesitar dúas texturas que van representar o joystick, como os da seguinte imaxe:



Cando prememos no centro, a textura central moverase (impulsada por nos) por dentro do círculo exterior.

- Deberedes por tanto definir dúas texturas, unha para o fondo e outra para o interior.
- Unha vez teñades as texturas é necesario definir un estilo para o Touchpad. Para definir o estilo necesitamos facer uso dun obxecto da clase `Skin` que ven ser como un almacén onde definimos recursos (texturas, fontes, cores...) que van poder ser usados por elementos gráficos como botóns, caixas de texto,...

Nota: Isto o veremos posteriormente no [sección de Stage UI](#).

```
private Skin touchpadSkin;
    ....
@Override
public void create () {
    .....
    touchpadSkin = new Skin();
    touchpadSkin.add("touchBackground", new Texture("LIBGDX_touchBackground.png"));
    touchpadSkin.add("touchKnob", new Texture("LIBGDX_touchKnob.png"));
    .....
}
```

Nese Skin imos cargar as dúas texturas anteriores.

- Agora necesitamos asinar o estilo ó Touchpad. Creamos por tanto un obxecto da clase `TouchpadStyle`.

```
private TouchpadStyle touchpadStyle;
    .....
@Override
public void create () {
    .....
    touchpadStyle = new TouchpadStyle();
}
```

- A este obxecto temos que asinarlle a dúas propiedades as texturas que se atopan no Skin, pero non podemos usar obxectos da clase `Texture`, se non que necesitamos obxectos da clase `Drawable`. O obxectos da clase `Drawable` os podemos obter a partires do método `getDrawable` da clase `Skin` no que lle pasamos de argumento o nome utilizado na liña: `touchpadSkin.add("touchBackground", new Texture("LIBGDX_touchBackground.png"))`;

```
.....
@Override
public void create () {
    .....
    Drawable D_background = touchpadSkin.getDrawable("touchBackground");
    Drawable D_knob = touchpadSkin.getDrawable("touchKnob");

}
```

- Como as texturas poden ter tamaños diferentes, definimos o tamaño de cada unha chamando ó método `setMinHeight` e `setMinWidth`:

```
.....
@Override
public void create () {
    .....
    D_background.setMinHeight(40);
    D_background.setMinWidth(40);

    D_knob.setMinHeight(15);
    D_knob.setMinWidth(15);
}
```

Como vemos estamos a poñer un tamaño de 40 píxeles ó círculo exterior e de 15 píxeles ó círculo interior.

- Unha vez temos modificados os obxectos da clase `Drawable`, os asinamos o estilo:

```
.....
```

```

@Override
public void create () {
    .....
    touchpadStyle.background = D_background;
    touchpadStyle.knob = D_knob;
}

```

- Agora só queda por definir o obxecto da **clase Touchpad** e asinarlle dito estilo:

```

private Touchpad touchpad;
    .....
@Override
public void create () {
    .....
    touchpad = new Touchpad(5, touchpadStyle);
    touchpad.setBounds(40, 40, 50, 50);
}

```

- Liña 7: Instanciamos o Touchpad e pasamos como parámetros o número de píxeles con respecto ó centro, a partir do cal a bola interior do joystick se move. Como segundo parámetro leva o estilo definido.
- Liña 8: Moi importante. O Touchpad se debe usar coa **clase Stage** que veremos posteriormente na **sección de Scene**. Coa chamada ó método `setBounds(x,y,width,height)` estamos a indicar a posición e tamaño do joystick.

A modo de introdución rápida diremos que a clase que necesitamos para debuxar é Stage. Dentro do Stage existen 'Actors' que son os elementos que se debuxan. Cada actor ten unha textura, posición, tamaño....

O Stage vai ter dous métodos moi importantes: `act` e `draw`.

O método `act` vai 'actualizar' todos os actores que se atopen engadidos ó Stage.  
O método `draw` vai debuxar todos os actores que se atopen no Stage.

Estes métodos deben ir no método `render`.

```

private Touchpad touchpad;
    .....
@Override
public void create () {
    .....
    /* Stage */

    stage = new Stage();
    stage.addActor(touchpad);

    Gdx.input.setInputProcessor(stage);
}

```

No noso caso, o Stage está conformado por un único actor que é o Touchpad. Como queremos que o Stage xestione os eventos dos actores que están engadidos a el (así moverá o roda interior do joystick) indicamos que a xestión de eventos o leve él (liña 12).

**Nota:** O cambiar a xestión de eventos ó Stage perdemos a xestión de interface `InputProcessor`. Se queredes manter as dúas teriades que utilizar **múltiples interfaces de eventos** como vimos anteriormente.

- O Stage ó igual que a cámara, ten un **viewport** asociado. Se queremos que se modifique en función do tamaño da pantalla teremos que facer isto:

```

public void resize(int width, int height) {
    stage.getViewport().update(width, height, true);
}

```

Co parámetro `true` indicamos que a cámara debe situarse no centro.

- Por último temos que facer o `render` dos actores:

```

@Override
public void render() {
    Gdx.gl.glClearColor(0, 0, 0, 1);
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

    stage.act(Gdx.graphics.getDeltaTime());
    stage.draw();
}

```

## Control do Touchpad

Para saber en que posición está o Touchpad temos os seguintes métodos:

- **getKnobPercentX()**: Devolve a posición X do círculo central coma unha porcentaxe con respecto á distancia có círculo exterior. Vai dende -1 a +1 (esquerda-dereita).
- **getKnobPercentY()**: Devolve a posición Y do círculo central coma unha porcentaxe con respecto á distancia có círculo exterior. Vai dende -1 a +1 (arriba-abaxo).
- **getKnobX()**: Devolve a posición X do círculo central con respecto á distancia có círculo exterior. O valor vai depender do tamaño asinado ó Touchpad.
- **getKnobY()**: Devolve a posición Y do círculo central con respecto á distancia có círculo exterior. O valor vai depender do tamaño asinado ó Touchpad.

## Exemplo de código

Neste exemplo vaise mover un gráfico pola pantalla de acordo ó movemento do Touchpad.

Para mover o gráfico se fixo uso da clase **Sprite**. Dita clase permite almacenar a textura, tamaño e posición. Poderíamos utilizar unha textura e unha clase separada para gardar os datos da figura (posición, tamaño, velocidade) como fixemos no xogo seguindo o Modelo-Vista-Contralador. Non se utilizou esta clase por mezclar o modelo-vista, pero a estas alturas tedes total liberdade para utilizar esta clase.

### Preparación:

- Descargade o seguinte arquivo e descomprimídeo no cartafol assets da versión Android. [Media:LIBGDX\\_touchpad.zip](#)
- Crear unha nova clase e cambiar os diferentes proxectos para que carguen dita clase.

### Código da clase **UsoTouchPad**

**Obxectivo:** Amosar como funciona o Touchpad.

```

import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.Sprite;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.scenes.scene2d.ui.Skin;
import com.badlogic.gdx.scenes.scene2d.ui.Touchpad;
import com.badlogic.gdx.scenes.scene2d.ui.Touchpad.TouchpadStyle;
import com.badlogic.gdx.scenes.scene2d.utils.Drawable;

public class UsoTouchPad extends ApplicationAdapter {
    private SpriteBatch batch;

    private Skin touchpadSkin;
    private TouchpadStyle touchpadStyle;
    private Touchpad touchpad;

    private OrthographicCamera camara2d;

```

```

private Stage stage;
private Sprite grafico;
private float velocidade = 5;

@Override
public void create () {
batch = new SpriteBatch();

grafico = new Sprite(new Texture("badlogic.jpg"));
grafico.setBounds(100, 150, 25, 25);

camara2d = new OrthographicCamera();
camara2d.setToOrtho(false,300,300);
camara2d.update();

batch.setProjectionMatrix(camara2d.combined);

/* TOUCHPAD */
touchpadSkin = new Skin();
touchpadSkin.add("touchBackground", new Texture("LIBGDX_touchBackground.png"));
touchpadSkin.add("touchKnob", new Texture("LIBGDX_touchKnob.png"));

touchpadStyle = new TouchpadStyle();
Drawable D_background = touchpadSkin.getDrawable("touchBackground");
Drawable D_knob = touchpadSkin.getDrawable("touchKnob");

D_background.setMinHeight(40);
D_background.setMinWidth(40);

D_knob.setMinHeight(15);
D_knob.setMinWidth(15);

touchpadStyle.background = D_background;
touchpadStyle.knob = D_knob;

touchpad = new Touchpad(5, touchpadStyle);
touchpad.setBounds(40, 40, 50, 50);

/* Stage */

stage = new Stage();
stage.addActor(touchpad);

Gdx.input.setInputProcessor(stage);
}

@Override
public void render() {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

batch.begin();
batch.draw(grafico.getTexture(), grafico.getX(), grafico.getY(),grafico.getWidth(),grafico.getHeight());
batch.end();

stage.act(Gdx.graphics.getDeltaTime());
stage.draw();

//Movemos o grafico
grafico.setX(grafico.getX() + touchpad.getKnobPercentX()*velocidade);
grafico.setY(grafico.getY() + touchpad.getKnobPercentY()*velocidade);

}

public void resize(int width, int height) {
stage.getViewport().update(width, height, true);
}

@Override
public void dispose() {
stage.dispose();
}

```

```
batch.dispose();  
Gdx.input.setInputProcessor(null);  
}  
}
```

- Liñas 83-84: Son as encargadas de mover o Sprite en función da velocidade e da distancia do punto central ó extremo. Canto máis próximo máis velocidade.

Se executades dará como resultado isto:



## TAREFA OPTATIVA A FACER

Modifícade o control do xogo para mover o Alien utilizando un TouchPad.