LIBGDX Scene UI

UNIDADE 3: Scene2D UI

Sumario

- 1 Introdución
- 2 Elementos necesarios
- 3 Proceso de uso
    - ♦ 3.1 Cargar o atlas e o estilo asociado
    - ♦ 3.2 Elementos gráficos do scene2d UI
    - ♦ 3.3 Eventos e Accións
        - ◊ 3.3.1 Eventos
            - · 3.3.1.1 Diálogos
            - · 3.3.1.2 Botón BACK de Android
        - ◊ 3.3.2 Accións
    - ♦ 3.4 Exemplo de código
- 4 TAREFA OPTATIVA A FACER

Introdución

Información na wiki: https://github.com/libgdx/libgdx/wiki/Scene2d.ui

Neste punto imos explicar outro recurso do framework que é o Scene2d-ui (scene 2d user interface).

Ven ser como o Scene pero aplicado os recursos gráficos que imos necesitar para construír unha interface de usuario. Este paquete incorpora moitas clases típicas dunha interface (botóns, cadros de texto, checkbox....).

O Scene é outra forma de xestionar os elementos gráficos nun xogo. Basease no uso de Actor´s. Cada Actor ven ser un elemento gráfico do xogo sobre o cal queremos ter algún control. Permite debuxar e xestionar o control do elemento dentro da propia clase Actor (mestura Modelo-Vista-Controlador). Fai que o seu control sexa moi sinxelo e como característica que a min máis me gosta, ten a posibilidade de asociar a cada Actor unha serie de accións. As accións poden afectar ó seu aspecto (deformalo, cambiarlle a cor,....) como a súa posición,... Ditas accións pódense programar para que vaian unha detrás doutra, en paralelo, que se executen despois dun tempo, que se repitan....

Centrándonos no Scene2D UI, teremos esas mesmas opcións pero aplicados a elementos gráficos propios dun deseño da interface dun usuario.

Elementos necesarios

Necesitamos:

- Un Atlas con todos os gráficos necesarios: Os gráficos das fontes, dos botóns,....
- Un arquivo json no que están definidos a cor, fonte, gráficos...para cada elemento gráfico. É o que se coñece como un **estilo** e ven ser un obxecto da clase Skin.

O estilo o podemos xerar por programación, pero o máis normal é telo nun arquivo externo coa extensión json.

Como punto de partido imos a utilizar o arquivo json xunto cos bmp, atlas e fonte necesarios que veñen nos exemplos do motor de xogos: https://github.com/libgdx/libgdx/tree/master/tests/gdx-tests-android/assets/data => Arquivo uiskin.atlas, uiskin.png,uiskin.json e default.fnt.

Podedes descargar estes arquivos dende este enlace: Media:LIBGDX_stage2dui.zip

Nota: Se descargades os arquivos do sitio web, o estilo definido para a lista variou, polo que vos dará un erro. Podedes ver a diferenza no arquivo adxunto.

Descomprimide os arquivos e copialos ó cartafol Assets do proxecto Android.

Nota: Tamén podedes crear o voso propio skin. Un exemplo o tedes neste enlace.

O que imos facer con estes arquivos é crear un estilo (parecido o CSS de HTML ou os arquivos skin de ASP.NET).

Se editamos o arquivo json podemos ver entre outras cousas:

```
com.badlogic.gdx.graphics.g2d.BitmapFont: { default-font: { file: default.fnt } },
com.badlogic.gdx.graphics.Color: {
green: { a: 1, b: 0, g: 1, r: 0 },
white: { a: 1, b: 1, g: 1, r: 1 },
red: { a: 1, b: 0, g: 0, r: 1 },
black: { a: 1, b: 0, g: 0, r: 0 }
},
com.badlogic.gdx.scenes.scene2d.ui.Skin$TintedDrawable: {
dialogDim: { name: white, color: { r: 0, g: 0, b: 0, a: 0.45 } }
},
com.badlogic.gdx.scenes.scene2d.ui.Button$ButtonStyle: {
default: { down: default-round-down, up: default-round },
toggle: { down: default-round-down, checked: default-round-down, up: default-round }
},
.....................
```

Analicemos o código:

Nesta liña:

```
com.badlogic.gdx.graphics.g2d.BitmapFont: { default-font: { file: default.fnt } },
```

Indicamos o tipo de fonte de texto que imos a usar (arquivo default.fnt) nos elementos gráficos (label´s, botóns con texto,...)

Aquí definimos constantes de cores:

```
com.badlogic.gdx.graphics.Color: {
green: { a: 1, b: 0, g: 1, r: 0 },
white: { a: 1, b: 1, g: 1, r: 1 },
red: { a: 1, b: 0, g: 0, r: 1 },
black: { a: 1, b: 0, g: 0, r: 0 }
},
```

Aquí definimos o estilo dos botóns:

```
com.badlogic.gdx.scenes.scene2d.ui.Button$ButtonStyle: {
default: { down: default-round-down, up: default-round },
toggle: { down: default-round-down, checked: default-round-down, up: default-round }
},
```

Por defecto cando prememos sobre un, poñerá o gráfico 'default-round-down' e o soltalo 'default-round'. Ditos gráficos os podedes atopar no atlas.

Proceso de uso

Cargar o atlas e o estilo asociado

Para cargar os estilos necesitamos utilizar a clase AssetManager:

```
private AssetManager assetManager;
private Skin skin;
        ............

@Override
public void create () {
                ............
```

```
assetManager = new AssetManager();
assetManager.load("uiskin.atlas",TextureAtlas.class);
assetManager.finishLoading();
TextureAtlas atlas = assetManager.get("uiskin.atlas", TextureAtlas.class);
skin = new Skin(Gdx.files.internal("uiskin.json"), atlas); // Cargamos os estilos
}

@Override
public void dispose() {

assetManager.dispose();
skin.dispose();
}
```

Elementos gráficos do scene2d UI

Máis información en: https://github.com/libgdx/libgdx/wiki/Scene2d.ui#layout-widgets

Unha vez cargados os elementos gráficos entramos no Scene2D.

Este manual é unha pequena introdución do que se pode facer, polo que explicaremos con exemplos, algunhas das posibilidades.

Ó igual que sucede en Android, imos ter diferentes formas de colocar os elementos gráficos na pantalla. O modo en como se colocan é o que se coñece como layout.

A nivel de layout dispoñemos das seguintes opcións:

- Table
- Container
- Stack
- ScrollPane
- SplitPane
- Tree
- VerticalGroup
- HorizontalGroup

Un dos máis habituais a utilizar é o Table.

Tedes neste enlace unha explicación bastante completa sobre este layout e o que se pode facer.

Unha vez decidido o layout (como se van colocar os elementos) pasaríamos a engadir os elementos gráficos (Widgets) que conformarán a nosa pantalla.

Entre eles temos:

- Label
- Image
- Button
- TextButton
- ImageButton
- CheckBox
- TextField
- List

E moitos outros.

Para poder debuxar todos estes elementos (layout e widgets) imos facer uso da clase Stage.

Dentro do Stage existen 'Actors' que son os elementos que se debuxan (widgets). 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: Neste punto actualizaríamos posicións (se ten acc
O método draw vai debuxar todos os actores que se atopen no Stage.
```

Por exemplo, imos facer que o Stage estea composto por un label cun texto.

```
private Stage stage;
        ...........
@Override
public void create () {

assetManager = new AssetManager();
assetManager.load("uiskin.atlas",TextureAtlas.class);
assetManager.finishLoading();
TextureAtlas atlas = assetManager.get("uiskin.atlas", TextureAtlas.class);
skin = new Skin(Gdx.files.internal("uiskin.json"), atlas); // Cargamos os estilos

stage = new Stage();
cargarElementosGraficos();

}
private void cargarElementosGraficos(){
Label titulo = new Label("Texto de exemplo",skin);
titulo.setColor(Color.RED);
titulo.setFontScale(2);
titulo.setBounds(0, 100, Gdx.graphics.getWidth(), 10);
titulo.setAlignment(Align.center);

stage.addActor(titulo);
}

@Override
public void resize(int width, int height) {
// TODO Auto-generated method stub
stage.getViewport().update(width, height, true);
}
@Override
public void dispose() {

assetManager.dispose();
skin.dispose();

stage.dispose();
}
```

- Liñas 16-24: Creamos un elemento gráfico de tipo Label, cun texto e o estilo cargado previamente.

  Indicamos:

  Liña 18: Unha cor.
  Liña 19: Un tamaño.
  Liña 20: Unha posición e tamaño que vai ocupar o contenedor que ten o label (ven ser como un bloque que rodea ó label).
  Liña 21: Aliñación do texto dentro do bloque. Neste caso centrado xa que o bloque ocupa todo o ancho da pantalla.

  Liña 23: Engadimos o Actor (a etiqueta) ó Stage.

O Stage vai ter a súa propia cámara co que vai debuxar todos os actores.

Para facelo:

```
@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();
}
```

Liña 6: Se os actores tiveran algunha acción ou o programador modificar a súa posición, chamando o método act se actualizarían.
Liña 7: Debuxa todos os actores do Stage.

Agora amosaremos un exemplo de uso dunha táboa, engadindo un TextButton e o Label anterior a unha táboa, e engadindo a táboa o Stage:

Nota: Lembrar que tedes neste enlace unha explicación bastante completa sobre este layout e o que se pode facer.

```
private void cargarElementosGraficos(){

Table tabla = new Table(skin);
tabla.setFillParent(true);
tabla.defaults().space(5); // Espazo por defecto arredor dos compoñenetes
tabla.defaults().fillX(); // o elemento (label,botón,..) ocupa todo o ancho => centrar
tabla.align(Align.top|Align.center);

Label titulo = new Label("Texto de exemplo",skin);
titulo.setColor(Color.RED);
titulo.setFontScale(2);
titulo.setBounds(0, 100, Gdx.graphics.getWidth(), 10);
titulo.setAlignment(Align.center);
titulo.setPosition(20, 50);

tabla.add(titulo);
tabla.row();
tabla.row();

TextButton boton = new TextButton("Premer aquí", skin);

tabla.add(boton).height(100).width(100);

stage.addActor(tabla);
}
```

Nota: Notaredes que cando prememos o botón no pasa nada e non cambia de aspecto.

Veremos como cando engadamos o evento todo aparece correctamente.

Eventos e Accións

Eventos

Para xestionar os eventos dos botóns (ou de calquera outro elemento gráfico) temos que utilizar a interface asociada á xestión de dito evento.

- Pero primeiro temos que indicarlle que os eventos os vai xestionar o Stage.

Nota: Se o voso xogo xa ten unha xestión de eventos (usando o InputProcessor) necesitaredes incorporar esta outra forma e polo tanto teredes que ler este enlace:
http://manuais.iessanclemente.net/index.php/LIBGDX_Xestion_Eventos_GestureListener#Xestionando_m.C3.BAltiples_interfaces_de_eventos

```
@Override
public void create () {
            ..............
                 Gdx.input.setInputProcessor(stage);


        }
@Override
public void dispose() {

Gdx.input.setInputProcessor(null);
                ............
```

```
}
```

- Despois temos que engadir a interface ó botón. No exemplo anterior:

```
boton.addListener(new ClickListener(){
public void clicked (InputEvent event, float x, float y) {
Gdx.app.exit();
}
});
```

Neste exemplo sairemos da aplicación ó premer.

Diálogos

Pode ser interesante que apareza unha caixa de diálogo preguntando se estamos seguros de querer saír.

Para facelo temos que utilizar a clase Dialog

Un exemplo de uso, sobre o botón anterior:

```
new Dialog("Sair do xogo", skin, "dialog") { // Os acentos non aparecen xa que non están no png das letras
       protected void result (Object object) {
                    if ((Boolean) object)
                         Gdx.app.exit();
       }
}.text("Estás seguro de querer sair ?")
.button("Yes", true)
.button("No",false)
.key(Keys.ENTER, true)
.key(Keys.ESCAPE, false)
.show(stage);
```

Analicemos o código:

- Liña 1: Creamos un obxecto da clase Dialog, indicando o título, o estilo e o tipo de diálogo.

O tipo de diálogo está definido no arquivo uiskin.json:

```
com.badlogic.gdx.scenes.scene2d.ui.Window$WindowStyle: {
default: { titleFont: default-font, background: default-window, titleFontColor: white },
dialog: { titleFont: default-font, background: default-window, titleFontColor: white, stageBackground: dialogDim }
},
```

- Liña 2-5: A este método se chamará automaticamente cando escollamos unha das opcións do Dialog. O parámetro vai servir para saber que se escolleu (true / false = aceptar/cancelar).
- Liña 5: Texto que aparece no diálogo.
- Liña 6-7: Texto e valores asociados ás opción de Aceptar-Cancelar. Neste caso valores true/false que son os usados como valores recibidos no método result analizado na liña anterior.
- Liñas 8-9: Que teclas teñen os mesmos efectos que premer sobre as opcións do diálogo. No exemplo, premer a tecla ENTER ten o mesmo efecto que premer o botón Yes.
- Liña 10: Amosar o diálogo no Stage.

Este código tería que ser chamado dende o click do botón.

Botón BACK de Android

Como sabedes, cando prememos o botón BACK dun dispositivo Android estamos a pechar a aplicación.

Libgdx permite capturar dito evento e preguntar (como fixemos antes) se realmente queremos saír do xogo.

Para facelo:

- Temos que chamar ó método setCatchBackKey do paquete Gdx.input, indicando true como parámetro.
- Sobreescribimos ó método keyDown do Stage.

```
@Override
public void create () {
            ...........
            Gdx.input.setCatchBackKey(true);
    stage = new Stage(){
@Override
public boolean keyDown(int keyCode) {
   if (keyCode == Keys.BACK) {
// CHAMAMOS O DIALOGO PARA PECHAR OU NON
   }
   return super.keyDown(keyCode);
}
};
```

Accións

Información na wiki: https://github.com/libgdx/libgdx/wiki/Scene2d#actions

É un dos elementos máis espectaculares que ten este tipo de programación.

O uso da clase Stage vainos permitir facer diferentes accións sobre os actors que o compoñen.

Dispoñemos de 3 tipos de accións:

1. Accións animadas.
2. Accións compostas.
3. Outras accións.

As **accións animadas** modifican varias propiedades dos actor´s, coma a posición, rotación, escala e factor alfa. Son as seguintes.

- FadeIn ? modifica o factor alfa do actor do valor que teña ó valor 1.
- FadeOut - modifica o factor alfa do actor do valor que teña ó valor 0.
- FadeTo - modifica o factor alfa do actor do valor que teña ó valor especificado.
- MoveBy ? move o actor unha distancia especificada.
- MoveTo ? move o actor a unha posición específica.
- RotateBy ? rota o actor engadindo ó ángulo especificado.
- RotateTo ? rota o actor ata un ángulo especificado.
- ScaleTo ? escala o actor ó valor indicado.

Accións compostas combinan múltiple accións en unha:

- Parallel ? executa todas as accións en paralelo. Todas á vez.
- Sequence ? executa todas as accións en secuencia. Unha detrás de outra.

Outras accións:

- Repeat ? repite a acción n-veces.
- Forever ? repite a acción indefinidamente.
- Delay ? retrasa a execución dunha acción o tempo especificado.
- Remove ? elimina o Actor especificado do Stage.

As accións poden aplicarse sobre cada Actor ou sobre o Stage completo.

Por exemplo, imos facer que a pantalla apareza toda negra e vaian aparecendo os compoñentes do Stage.

Para iso temos que utilizar as Accións: Actions.fadeOut e Actions.fadeIn.

```
stage.addAction(Actions.fadeOut(0));
  stage.addAction(Actions.fadeIn(4));
```

Pero podemos facelo doutra forma máis elegante, xa que a acción pode estar composta por moitas accións as cales se poden executar en paralelo, secuencial, ... como vimos antes.

```
stage.addAction(Actions.sequence(Actions.fadeOut(0),Actions.fadeIn(4)));
```

Nota: Ós propios actores poden ter accións asociadas, chamando o método addAction de cada un deles.

Nota: Para non ter que estar escribindo continuamente Actions.acción podemos facer este import:

```
import static com.badlogic.gdx.scenes.scene2d.actions.Actions.*;
```

Outro exemplo de accións, no que movemos continuamente e de forma indefinida un botón de esquerda a dereita:

```
boton.addAction(Actions.forever(Actions.sequence(Actions.moveTo(0, 0, 5),Actions.moveTo(Gdx.graphics.getWidth(), Gdx.graphics.getHeig
```

Como vemos temos:

- Actions.forever: Repite de forma indefinida a acción que ven a continuación.
- Actions.sequence: Executa na orde indicada o conxunto de accións que veñen a continuación.
- Actions.moveTo(0, 0, 5),Actions.moveTo(Gdx.graphics.getWidth(), Gdx.graphics.getHeight(), 5): Move cara a coordenada (0,0) o botón cunha duración de 5 segundos e move cara a esquina superior dereita o botón cunha duración de 5 segundos.

Exemplo de código

Preparación:

- Descargade o seguinte arquivo e descomprimídeo no cartafol assets da versión Android.Media:LIBGDX_stage2dui.zip.
- Crear unha nova clase e cambiar os diferentes proxectos para que carguen dita clase.

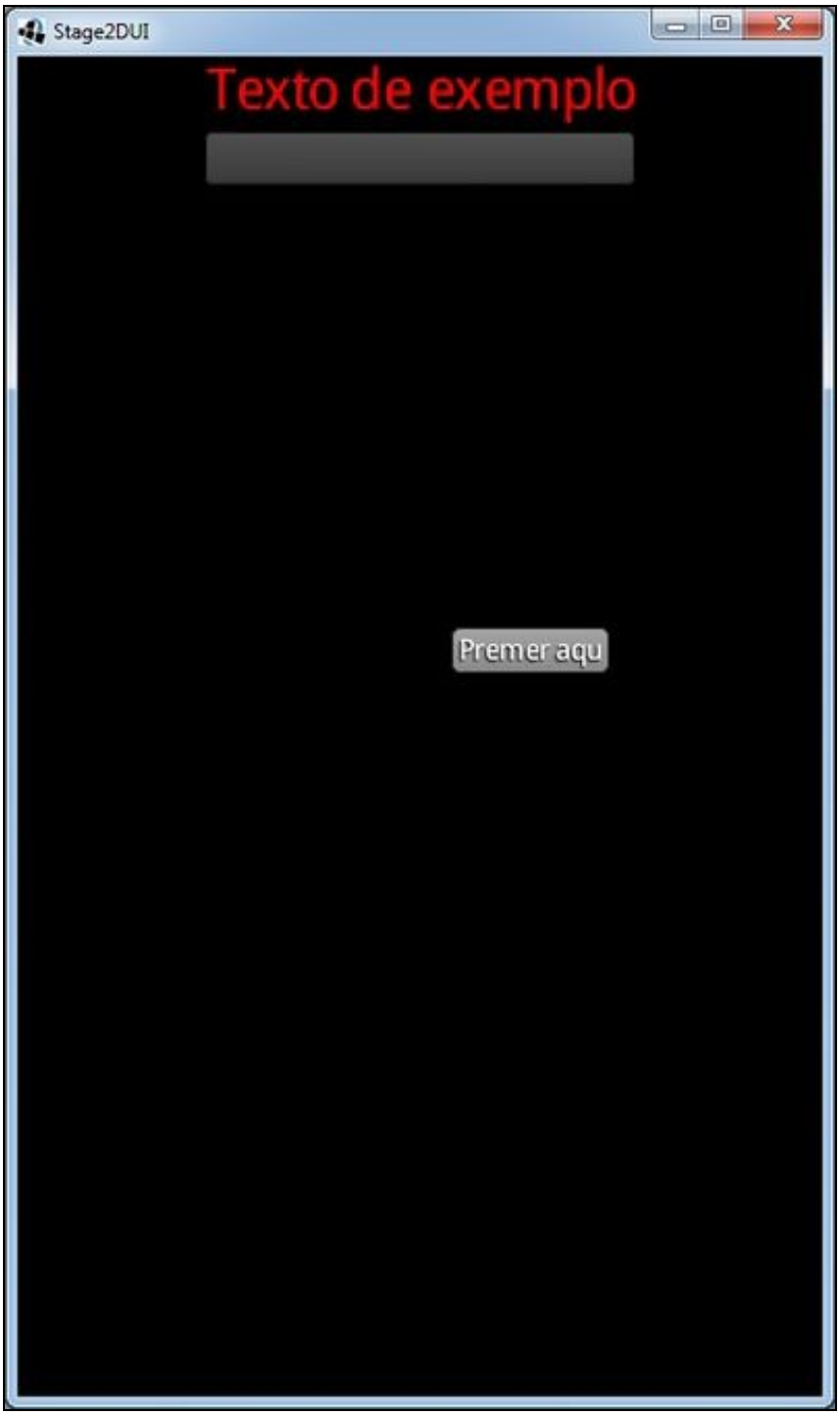
Cȩdigo da clase Stage2DUI
Obxectivo: Exemplo de uso do Stage2D ui, eventos e accións.

```
import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Input.Keys;
import com.badlogic.gdx.assets.AssetManager;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.g2d.TextureAtlas;
import com.badlogic.gdx.scenes.scene2d.InputEvent;
import com.badlogic.gdx.scenes.scene2d.Stage;
import com.badlogic.gdx.scenes.scene2d.actions.Actions;
import com.badlogic.gdx.scenes.scene2d.ui.Dialog;
import com.badlogic.gdx.scenes.scene2d.ui.Label;
import com.badlogic.gdx.scenes.scene2d.ui.Skin;
import com.badlogic.gdx.scenes.scene2d.ui.Table;
import com.badlogic.gdx.scenes.scene2d.ui.TextButton;
import com.badlogic.gdx.scenes.scene2d.ui.TextField;
import com.badlogic.gdx.scenes.scene2d.utils.Align;
import com.badlogic.gdx.scenes.scene2d.utils.ClickListener;

public class Stage2DUI extends ApplicationAdapter {

private AssetManager assetManager;
private Skin skin;
private Stage stage;
```

```
@Override
public void create () {

assetManager = new AssetManager();
assetManager.load("uiskin.atlas",TextureAtlas.class);
assetManager.finishLoading();
TextureAtlas atlas = assetManager.get("uiskin.atlas", TextureAtlas.class);
skin = new Skin(Gdx.files.internal("uiskin.json"), atlas); // Cargamos os estilos

    Gdx.input.setCatchBackKey(true);
stage = new Stage(){
@Override
public boolean keyDown(int keyCode) {
if (keyCode == Keys.BACK) {
avisarSair();
}
return super.keyDown(keyCode);
}
};
Gdx.input.setInputProcessor(stage);

cargarElementosGraficos();

}

private void cargarElementosGraficos(){

Table tabla = new Table(skin);
tabla.setFillParent(true);
tabla.defaults().space(5); // Espazo por defecto arredor dos compoñenetes
tabla.defaults().fillX(); // o elemento (label,botón,..) ocupa todo o ancho => centrar
tabla.align(Align.top|Align.center);

Label titulo = new Label("Texto de exemplo",skin);
titulo.setColor(Color.RED);
titulo.setFontScale(2);
titulo.setBounds(0, 100, Gdx.graphics.getWidth(), 10);
titulo.setAlignment(Align.center);
titulo.setPosition(20, 50);

tabla.add(titulo);
tabla.row();
tabla.row();

TextField texto = new TextField("", skin);
texto.setBounds(0, 50, Gdx.graphics.getWidth(), 10);

tabla.add(texto);


TextButton boton = new TextButton("Premer aquí", skin);
boton.addListener(new ClickListener(){
public void clicked (InputEvent event, float x, float y) {
avisarSair();
}
});
boton.setPosition(0, 0);
boton.addAction(Actions.forever(Actions.sequence(Actions.moveTo(0, 0, 5),Actions.moveTo(Gdx.graphics.getWidth(), Gdx.graphics.getHeig
stage.addActor(tabla);
stage.addActor(boton);

stage.setKeyboardFocus(texto);// Poñemos o foco na caixa de texto
}

private void avisarSair(){
new Dialog("Sair do xogo", skin, "dialog") { // Os acentos non aparecen xa que non están no png das letras
       protected void result (Object object) {
                    if ((Boolean) object)
                         Gdx.app.exit();
       }
}.text("Estás seguro de querer sair ?")
.button("Yes", true)
```

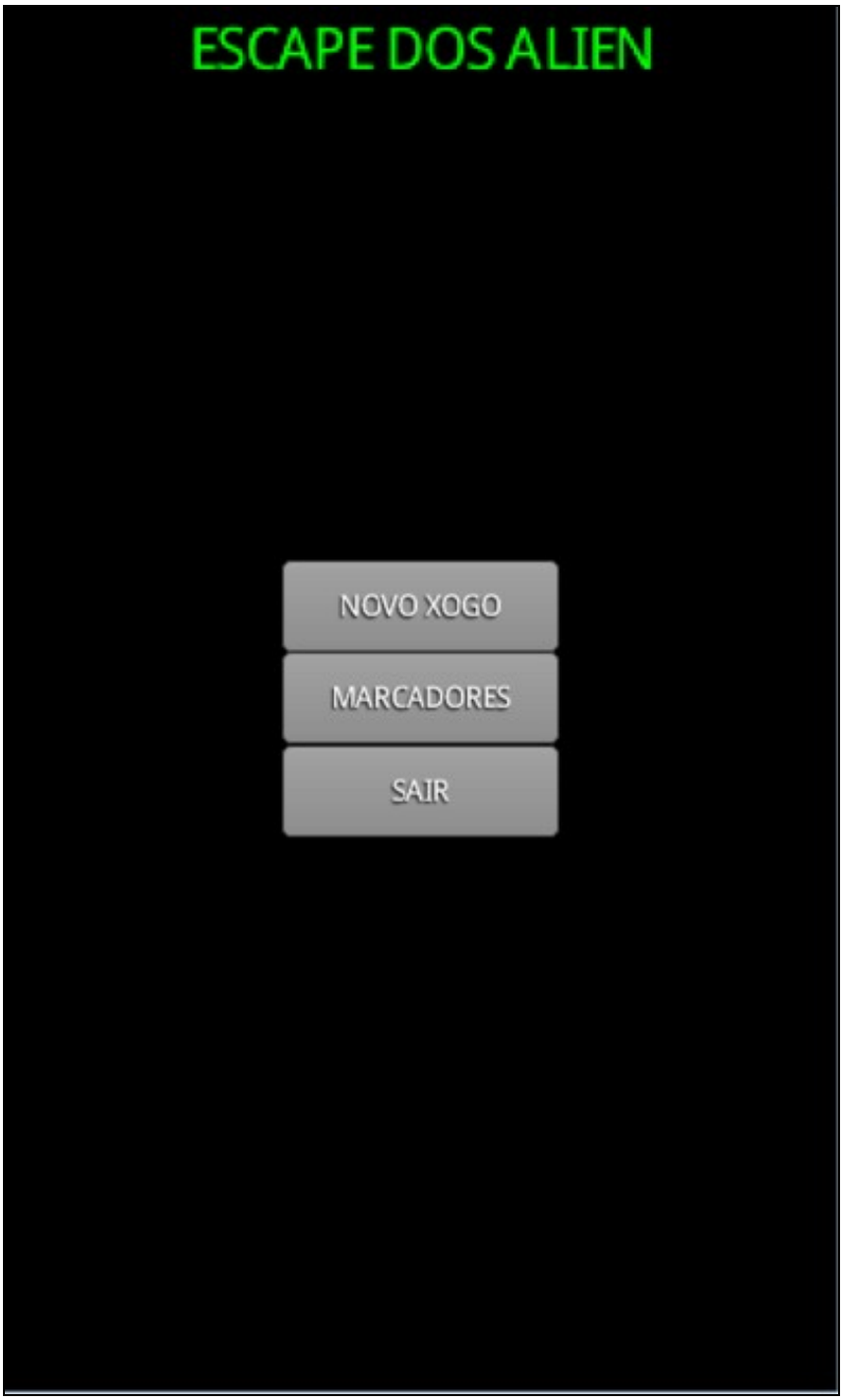
```
.button("No",false)
.key(Keys.ENTER, true)
.key(Keys.ESCAPE, false)
.show(stage);

}

@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();
}
@Override
public void resize(int width, int height) {
// TODO Auto-generated method stub
stage.getViewport().update(width, height, true);
}

@Override
public void dispose() {

Gdx.input.setInputProcessor(null);
assetManager.dispose();
skin.dispose();

stage.dispose();
}


}
```

Ó executar dará como resultado un botón movéndose de forma indefinida e unha caixa de texto para introducir un texto co foco nel.


Stage2DUI
Texto de exemplo
Premer aqu

TAREFA OPTATIVA A FACER

Modifica a pantalla principal do xogo:



E utiliza o Stage2d UI para desenvolvela.

Podes engadir efectos de fadein / fadeout ou calquera outro efecto que queiras.

-- Ángel D. Fernández González -- (2014).