

1 LIBGDX Tareas entregar Unidade 2

1.1 Sumario

- 1 UNIDADE 2
 - ◆ 1.1 Tarefa 2.0
 - ◆ 1.2 Tarefa 2.1
 - ◇ 1.2.1 Apartado A)
 - ◇ 1.2.2 Apartado B)
 - ◆ 1.3 Tarefa 2.2
 - ◆ 1.4 Tarefa 2.3
 - ◆ 1.5 Tarefa 2.4
 - ◆ 1.6 Tarefa 2.5
 - ◆ 1.7 Tarefa 2.6
 - ◆ 1.8 Tarefa 2.7
 - ◆ 1.9 Tarefa 2.8
 - ◇ 1.9.1 Tarefa 2.8.A
 - ◇ 1.9.2 Tarefa 2.8.B
 - ◆ 1.10 Tarefa 2.9
 - ◆ 1.11 Tarefa 2.10
 - ◆ 1.12 Tarefa 2.11
 - ◆ 1.13 Tarefa 2.12
 - ◆ 1.14 Tarefa 2.13
 - ◆ 1.15 Tarefa 2.14
 - ◆ 1.16 Tarefa 2.15

1.2 UNIDADE 2

1.2.1 Tarefa 2.0

[ENLACE A MIRAR.](#)

Captura unha pantalla onde aparezan os diferentes eventos polos que pasa o voso xogo imprimindo o voso nome e indicando o evento, tal como está nos apuntes.

1.2.2 Tarefa 2.1

1.2.2.1 Apartado A)

[ENLACE A MIRAR.](#)

- Imos descargar unha icona calquera [desta web](#) e o imos gardar no cartafol assets do proxecto Android, cónome 'xogo.png'.

O tamaño que temos que escoller dependerá da plataforma: 128x128 (para Mac), 32x32 (para Windows e Linux), e 16x16 (para Windows). Nos imos a escoller o da resolución 32x32. Asociar dita icona ó voso xogo.

- Fai que non se poida cambiar o tamaño da xanela na versión Desktop.

1.2.2.2 Apartado B)

[ENLACE A MIRAR.](#)

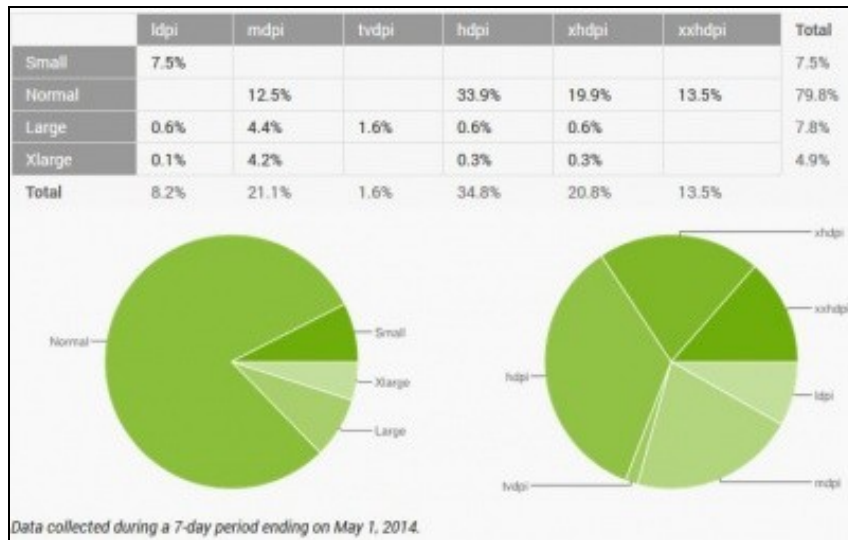
- Fai que o xogo na versión Móbil apareza en orientación portrait (non apaisada).

1.2.3 Tarefa 2.2

[ENLACE A MIRAR.](#)

Mirando as seguintes imaxes:

	Low density (120), ldpi	Medium density (160), mdpi	High density (240), hdpi	Extra high density (320), xhdpi
Small screen	QVGA (240x320)		480x640	
Normal screen	WQVGA400 (240x400) WQVGA432 (240x432)	HVGA (320x480)	WVGA800 (480x800) WVGA854 (480x854) 600x1024	640x960
Large screen	WVGA800** (480x800) WVGA854** (480x854)	WVGA800* (480x800) WVGA854* (480x854) 600x1024		
Extra Large screen	1024x600	WXGA (1280x800) [†] 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600



Observando os gráficos anteriores modifica a versión Desktop para que lance o xogo coas dimensións do dispositivo móbil que teña máis cuota (% por tamaño de pantalla, como amosa o segundo gráfico).

Faino como segundo vimos [neste punto](#).

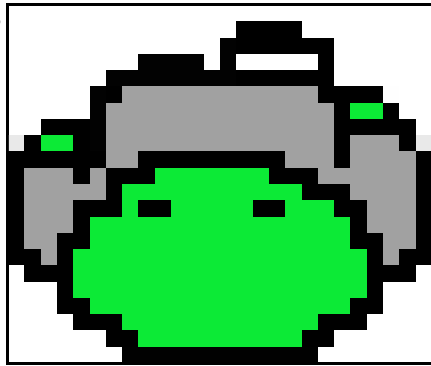
1.2.4 Tarefa 2.3

- [Enlace cámara](#): Necesario ler puntos 1.1,1.2,1.3. Posteriormente será necesario ler o punto 1.6.
- [Enlace ós gráficos](#): Necesario ler os puntos 1,2,3,4,5,6.

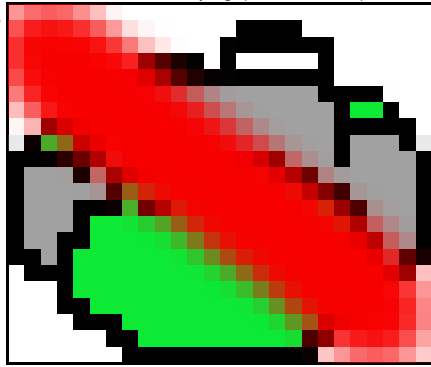
Dados os seguintes gráficos engádeos ó cartafol assets/GRAFICOS da plataforma Android e carga ditos gráficos no método cargarTexturas e libera ditas texturas no método liberarTexturas da clase AssetsXogo. O nome das variables que referencian a cada textura está entre os parénteses.

Podedes descargar todos os gráficos nun único arquivo: [Media:LIBGDx_itinerario1_alien.zip](#)

- Imaxes a descargar para a tarefa 2.3



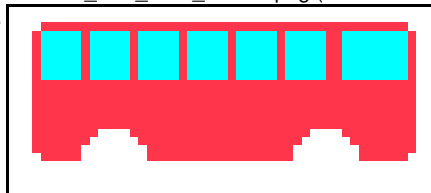
LIBGDX_itin1_alien.png (textureAlien).



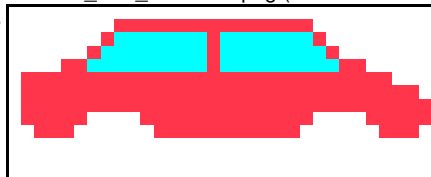
LIBGDX_itin1_alien_dead.png (textureAlienDead).



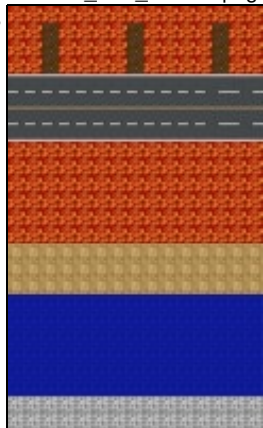
LIBGDX_itin1_alien_rescue.png (textureAlienRescue).



LIBGDX_itin1_autobus1.png (textureAutobus).



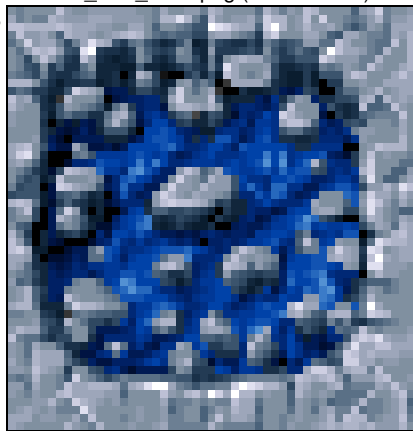
LIBGDX_itin1_coche1.png (textureCoche).



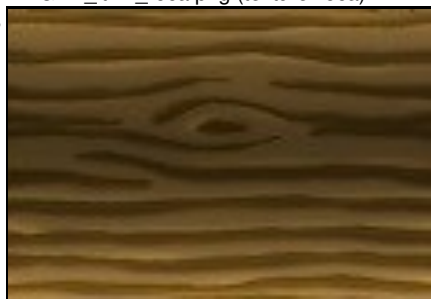
LIBGDX_itin1_fondoxogo.jpg (textureFondo).



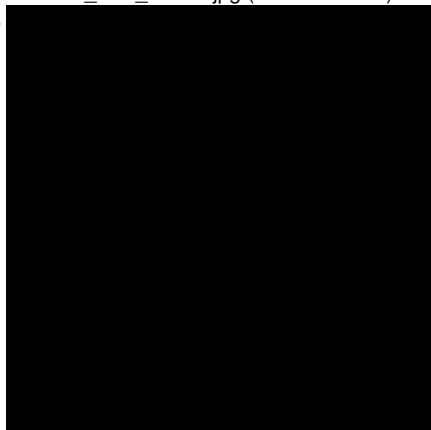
LIBGDX_itin1_nave.png (textureNave).



LIBGDX_itin1_roca.png (textureRoca).



LIBGDX_itin1_tronco.jpg (textureTronco).



LIBGDX_puntonegro.jpg (texturePuntoNegro).

1.2.5 Tarefa 2.4

ENLACE A MIRAR

- Crea unha nova clase de nome Nave que derive da clase Personaxe. Lembra crear o constructor como no exemplo do manual.
- Instancia e crea un obxecto da clase Nave na clase Mundo.

Os valores iniciais para a nave son:

Posición:(0,480)

Tamaño:(40,20)

Velocidade:60

- Crea método para debuxala de nome **debuxarNave()** na clase **RendererXogo**.

1.2.6 Tarefa 2.5

[ENLACE A MIRAR.](#)

Modifica a velocidade do Alien cando creas o obxecto na clase **Mundo** para que teña unha velocidade máxima que lle permita percorrer o ancho do seu mundo en 3 segundos.

1.2.7 Tarefa 2.6

[ENLACE A MIRAR.](#)

Seguindo o mesmo exemplo que os coches fai que as rochas aparezan no xogo. O número de rochas queda á vosa elección. O tamaño de cada rocha é de 60x60 unidades.

Os pasos a seguir serán:

- Definir o array de rochas e engadilas na clase **Mundo**.
- Debuxar as rochas na clase **RendererXogo**.
- Controlar as rochas e actualizar a súa posición na clase **ControladorXogo**.

[Unha posible alternativa á solución proposta.](#)

1.2.8 Tarefa 2.7

[ENLACE A MIRAR.](#)

Informar á clase controladora cando o xogador libera a tecla (**método `keyUp`**), chamando ó método **liberarTecla**.

1.2.9 Tarefa 2.8

1.2.9.1 Tarefa 2.8.A

TAREFA OPTATIVA DE FACER

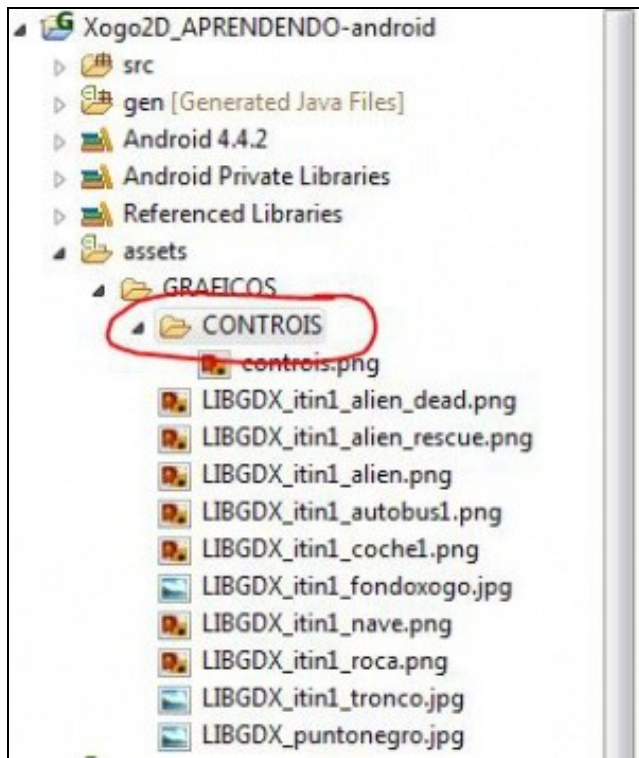
[ENLACE A MIRAR: interfaces para capturar eventos.](#)

[ENLACE A MIRAR: facer un unproyect.](#)

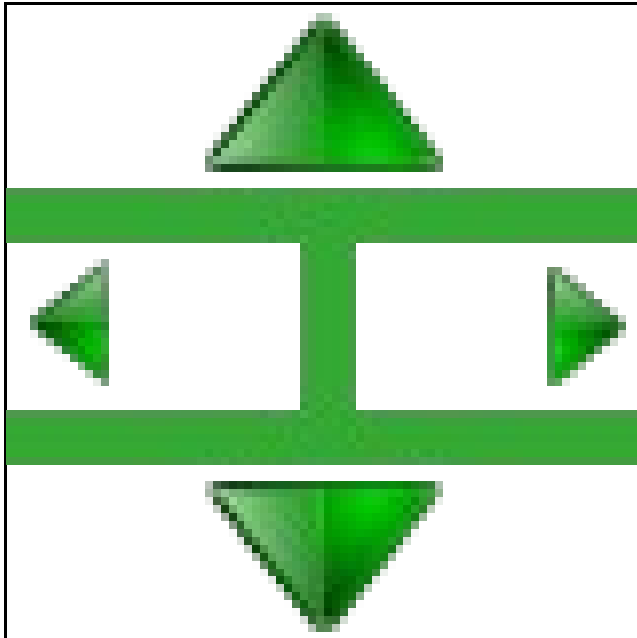
Tarefas a realizar:

O que temos que facer será poñer un gráfico formado por catro frechas que vai ser utilizado para mover o noso Alien. Debemos de facer o **unProyect** e ver sobre que frecha estamos a premer para mover o Alien.

- Crear un cartafol de nome **CONTROIS** dentro do cartafol **/assets/GRAFICOS/** da versión android.



- Subir o seguinte gráfico a dito cartafol CONTROIS e cargar nunha textura de nome **control** dito gráfico.



Nota: Calquera alumno é libre de crear un control diferente ou dividir o control en dous e poñer unha parte a dereita (arriba-abaxo) e outra a esquerda (esquerda-dereita). Tedes total liberdade.

- Definir o tamaño que ten dito control na **clase Controis** (copiade o código seguinte).

Para isto definimos unha variable de tipo **Rectangle** que permite gardar catro valores (x,y,width,height).

```
public final static Rectangle CONTROL = new Rectangle(10,40,50,70);
```

- Debuxar o control na clase **RendererXogo** no método **debuxarControis** tomando como valores os definidos no rectángulo.

Podedes facer que dito gráfico só se debuxe na versión móbil poñendo esta condicion:

```
if (Gdx.app.getType()==ApplicationType.Android) .....
```

- Para facer o seguinte punto (**facer un unproyect**) temos que poder acceder á cámara orthográfica dende a clase PantallaXogo. Podemos definila como static ou crear un método que nos devolva a cámara:

Código da clase **RendererXogo**

Obxectivo: definimos un método que devolva a cámara 2D. Usado pola clase PantallaXogo para facer o unProyect.

```
public OrthographicCamera getCamara2d() {
    return camara2d;
}
```

- Na clase PantallaXogo, no método touchdown, facer o unProyect cando pulsemos sobre a pantalla.

Facer un log indicando a posición premida en pixeles e en unidades do voso mundo.

1.2.9.2 Tarefa 2.8.B

TAREFA OPTATIVA DE FACER

ENLACE A MIRAR: [As colisións.](#)

- Verificar se prememos algunha parte do control do Alien, chamando o método pulsarTecla correspondente da clase ControladorXogo tal cal está feito no método keydown. **Lembrar liberar a pulsación da tecla no método touchup.**

Estas son as coordenadas para crear un rectángulo con cada un das partes do control de movemento (a imaxe anterior):

- **Frecha esquerda:**
Controis.CONTROL.x,Controis.CONTROL.y+Controis.CONTROL.height/3,Controis.CONTROL.width/2,Controis.CONTROL.height/3
- **Frecha dereita:**
Controis.CONTROL.x+Controis.CONTROL.width/2,Controis.CONTROL.y+Controis.CONTROL.height/3,Controis.CONTROL.width/2,Controis.CONTROL.height/3
- **Frecha arriba:**
Controis.CONTROL.x,Controis.CONTROL.y+Controis.CONTROL.height*2/3,Controis.CONTROL.width,Controis.CONTROL.height/3
- **Frecha abaixo:** Controis.CONTROL.x,Controis.CONTROL.y,Controis.CONTROL.width,Controis.CONTROL.height/3

A idea é crear na clase **class Controis** un rectángulo para cada frecha e asinarlle as posicións anteriores. Despois comprobaredes en cada un dos if's se se intersecciona o rectángulo có círculo que representa o dedo premendo a pantalla.

• Método touchDown:

- ◊ Pasar as coordenadas reais (en pixeles) a coordenadas do noso mundo (está feito da tarefa anterior 2.8.A).
- ◊ **Crear un círculo** coas coordenadas do dedo na pantalla. A clase círculo espera recibir tres parámetros (x,y,radio). O radio do círculo pode ser de 2 unidades. A coordenada x,y é a coordenada calculada na tarefa 2.8.A que indica a posición premida no sistema de coordenadas da cámara.
- ◊ Comprobar utilizando a clase Intersector se o Circle intersecciona con cada un dos rectángulos que representan as frechas e que están definidos na clase Controis.

if (Intersecciona frecha esquerda con dedo) entón

controladorXogo.pulsarTecla(ControladorXogo.Keys.ESQUERDA);

e así por cada frecha.

- **Método touchUp:** liberar a tecla correspondente chamando ó método liberarTecla da clase Controladora dependendo da frecha liberada.

1.2.10 Tarefa 2.9

ENLACE A MIRAR: [As colisións.](#)

Tarefa a realizar:

Fai que cando o alien toque á nave se inicialice e se engada unha vida de tipo SALVADO ó array de vidas.

1.2.11 Tarefa 2.10

[ENLACE A MIRAR: Facer un unproject.](#)

[ENLACE A MIRAR: As colisións.](#)

[ENLACE A MIRAR: Cambiando as pantallas.](#)

Tarefa a realizar:

Fai que, na pantalla de presentación (a que ten os botóns de Novo Xogo, Marcadores e Saír), en función do botón premido, cambiemos de pantalla ou saiamos. Se estamos na pantalla de Marcadores e prememos na pantalla deberemos volver á pantalla de presentación.

Pistas:

- Cada unha das opcións se pode calcular en que punto están facendo unha regra de tres. Así se a opción Saír se atopa no punto (160,320) (esquina inferior esquerda) nun tamaño de (480,800) píxeles e o noso mundo ten (300,500) unidades podemos concluír que o punto (160,320) = (100,200)

```
160-----480
x -----300 =====> x = (300*160)/480 = 100
```

Para non teres que facer estas operacións xa vos dou as coordenadas de cada botón nun array de Rectangles:

```
private Rectangle botons[]={new Rectangle(100, 268, 98, 32),new Rectangle(100, 235, 98, 32),new Rectangle(100, 200, 98, 32)};
```

- Para saír da aplicación temos que chamar ó método **Gdx.app.exit()**
- O proceso para controlar cando tocamos cada opción sería:

Ir ó método touchdown.

Obter as coordenadas do noso mundo cando prememos sobre a pantalla:

```
Vector3 temp = new Vector3(screenX,screenY,0); // Para optimizar poderiase instanciar unha vez no constructor.
camara2d.unproject(temp);
Circle dedo = new Circle(temp.x,temp.y,2); // Para optimizar poderiase instanciar unha vez no constructor.
```

Comprobar se tocamos algún dos botóns:

```
if (Intersector.overlaps(dedo,botons[0])) { // Prememos novo xogo
}
```

No caso de tocar, liberamos os gráficos e/ou recursos (no caso da PantallaPresentacion non é necesario) e cambiamos de pantalla:

```
dispose();
meuxogogame.setScreen(new PantallaXogo(meuxogogame));
```

- Lembrar chamar ó método dispose para liberar a textura e o spritebatch na clase PantallaMarcadores.

1.2.12 Tarefa 2.11

TAREFA OPTATIVA DE FACER

[ENLACE A MIRAR: Cambiando as pantallas.](#)

[ENLACE A MIRAR: Fontes.](#)

Imos engadir un tempo ó noso xogo, de tal forma que ó rematar iremos á pantalla de marcadores.

Tarefas a facer:

- Modificar a clase `RendererXogo` para que visualice o cronómetro. Para obter a referencia ó crono lembra utilizar `meuMundo.getCronometro()`.
NON CRES UN CRONOMETRO NOVO.

Utiliza a clase `StringBuilder` para visualizar o texto.
A posición do crono debe ser abaixo á esquerda.

- Fai que cando o cronómetro chegue a 0 remate o xogo e cambie á pantalla `PantallaMarcadores` (igual que cando leva máis de 15 vidas).
- Escribe o texto **ESCAPE DOS ALIEN** na parte superior da `PantallaMarcadores`.

1.2.13 Tarefa 2.12

[ENLACE A MIRAR: Música e efectos de son.](#)

Imos facer que se escoite unha música de fondo no xogo e a engadir diferentes efectos de son.

Para facelo imos crear un cartafol novo dentro de assets, de nome `MUSICA`, onde imos copiar todos os arquivos de audio.

- Música:

- ◊ De fondo: `Media:LIBGDX_grapes_I_dunno_m.mp3`
Facer que teña un volume a metade do máximo e que se repita (non deixe de tocar).
- ◊ Cando esteamos no xogo: `Media:LIBGDX_ocean_sounds.mp3`
Facer que teña un volume a 0.1f e que se repita (non deixe de tocar).

Efectos de son:

- Cando o alien morra se escoite: `Media:LIBGDX_morte.mp3`
- Cando o alien chegue á nave: `Media:LIBGDX_transporter_sfx.mp3`

O que temos que facer é:

- Crear unha clase de nome `Audio`, no paquete principal, onde se van cargar os arquivos multimedia anteriores.
- As variables a crear (que fagan referencia ós arquivos multimedia) serán `public - static`.
- Crea, na clase `Audio`, un método `inicializarMusica` de tipo `public-static` que cargue os arquivos multimedia. Chamao dende a clase `MeuXogoGame`, método `create`.
- Crea, na clase `Audio`, un método `dispose` que libere os recursos de audio. Chámao cando saíamos do xogo (método `dispose` da clase `MeuXogoGame`).
- Facer que na pantalla principal soe a música (deixade tocando para que continúe por todas as pantallas). Faino no constructor da clase `PantallaPresentacion`.
- Facer que cando esteamos na pantalla de pause a música pare e cando volvamos ó xogo a música continúe.

Lembrede que podemos facer pausa cambiando de aplicación (Android) ou minimizando (Desktop). Nese intre imos ó método `pause-resume` da clase. Pero tamén podemos facer pausa premendo o botón de pausa.

- Facer que a música do mar (`LIBGDX_ocean_sounds.mp3`) soe cando esteamos no xogo e pare ó saír (métodos `show / hide` da clase `PantallaXogo`).
- Facer que cando o alien morra soe o efecto de son: `LIBGDX_morte.mp3`.
- Facer que cando o alien suba á nave soe o efecto de son: `LIBGDX_transporter_sfx.mp3`.

Subide tamén ó cartafol `MUSICA` estes tres efectos de son que son utilizados [neste punto](#).

- Claxon1: `Media:LIBGDX_claxon.mp3`
- Claxon2: `Media:LIBGDX_claxon2.mp3`

- Claxon3: [Media:LIBGDX_claxon3.mp3](#)

1.2.14 Tarefa 2.13

TAREFA OPTATIVA DE FACER

[ENLACE A MIRAR: Facer un unproyect.](#)

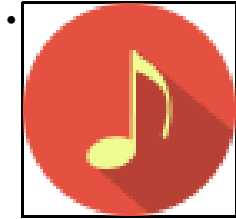
[ENLACE A MIRAR: As colisións.](#)

[ENLACE A MIRAR: Preferencias.](#)

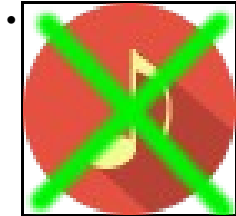
Imos gardar en forma de preferencias a opción de escoitar música de fondo no noso xogo. A idea é ter un gráfico na pantalla de presentación que indique cando queremos que se escoite a música no xogo.

- Descargade estes dous arquivos e engadídeos ó cartafol assets/CONTROIS do proxecto Android.

- Imaxes a descargar para a tarefa 2.13



LIBGDX_iconmusicaon.png (textureMusicaOn).



LIBGDX_iconmusicaoff.png (textureMusicaOff).

Pistas:

- Cargade ditos gráficos na clase AssetsXogo (e lembrade liberar a memoria).
- Debuxade os gráficos na clase PantallaPresentacion. Podedes definir o seu tamaño e posición con estes datos.

```
private Rectangle controlMusica = new Rectangle(10,10,50,50);
```

- Para controlar o estado (musica on /off) utilizade unha variable booleana que garde o valor da preferencia gardada. Podedes facelo no constructor.
- No método render da clase PantallaPresentacion debuxade a texture correspondente en función do valor da variable anterior.
- No método touchDown facer un unProyect e se se pulsa a icona de música cambiade o estado da variable booleana. Podedes gardar o valor da nova preferencia.

1.2.15 Tarefa 2.14

TAREFA OPTATIVA DE FACER

[ENLACE A MIRAR: Datos.](#)

[ENLACE A MIRAR: Fontes.](#)

Implementando a pantalla de Marcadores utilizando arquivos.

Partindo do **código xa visto da pantalla Presentación** (código que aparece ó principio do punto sinalado) imos darlle unha funcionalidade á pantalla de Marcadores.

A diferenza con respecto á pantalla de presentación é que non imos ter un gráfico que ocupe toda a pantalla (como no caso da pantalla de presentación) se non que imos escribir un texto. Polo tanto, do código da pantalla de presentación teremos que quitar:

```
spritebatch.disableBlending();
```

e

```
spritebatch.draw(fondo, 0, 0, Mundo.TAMANO_MUNDO_ANCHO, Mundo.TAMANO_MUNDO_ALTO);
```

Esta tarefa vai consistir en gardar os tres mellores xogadores e visualizalos na pantalla de Marcadores.

O criterio vai ser aquel que salve a maior cantidade de alien's.

Imos gardar as tres mellores puntuacións.

Isto o faremos no momento que se remate o xogo (acabe o tempo) e pasemos á pantalla PantallaMarcadores.

Aproveitaremos o código para que cando entremos en dita pantalla (tanto dende a PantallaPrincipal, como cando rematamos o xogo) para amosar as mellores puntuacións.

Unha posible forma de facelo (tedes liberdade para implementalo como queirades):

- Crear unha clase de nome HighScores. En dita clase definiremos:

- ◊ O array de Strings (3 valores, por defecto a "0").

```
public static String[] highscores = { "0", "0", "0" };
```

- ◊ O nome do arquivo onde gardar os datos dos highscores.

- ◊ Método public static void load(): carga as puntuacións no array anterior. Utilizado pola pantalla PantallaPuntuacion cando se amose.

Como no arquivo temos gardado o array de cadeas desta forma (un exemplo): "1","2","1" teremos que ler e converter os datos lidos a un array. Para facelo podemos usar o método split. Tedes un exemplo de uso [neste enlace](#).

- ◊ Método public static engadirPuntuacion(int puntuacion): pasamos a nova puntuación dende a pantalla PantallaXogo no momento que se acabe o xogo.

Este método comproba que os datos gardados teñan un valor menor que o novo valor enviado. En caso contrario substitúe un polo novo valor e o garda a disco.

- ◊ Método private static void save(): chamado dende o método engadirPuntuacion cando rematemos. Engade cada un dos datos do array **separados por comas**. Lembrar que para engadir temos que chamar ó método writeString co dato e o valor true como segundo parámetro.

1.2.16 Tarefa 2.15

ENLACE A MIRAR: Empaquetado e distribución.

Esta tarefa vai consistir en xerar o jar e o apk (non ten por que estar firmado) do voso xogo.