LIBGDX Gardando datos.

UNDIDADE 2: Gardando datos

Sumario

Introdución

Neste punto imos ver como podemos facer para gardar datos de forma persistente, é dicir, que os podamos recuperar no momento que queiramos.

Dependendo do dispositivo onde esteamos a executar o xogo, estes datos poden gardarse no disco duro (versión desktop) ou na tarxeta de memoria (versión móbil).

Imos distinguir dous tipos de datos:

- Referidos a preferencias de configuración do xogo.
- Datos de calquera tipo que necesitemos gardar, como os high-scores do xogo ou partidas gardadas (necesitamos gardar o estado do xogo).

Preferencias

Dirección da wiki: https://github.com/libgdx/libgdx/wiki/Preferences

As preferencias son unha forma de almacenamento no que os datos son gardados como se foran un hash map utilizando pares de cadea-valor.

Só podemos utilizar algúns tipos de datos para gardar información:

- ◊ Mapas de cadeas.
- ◊ Boolean.
- ◊ Float.
- ◊ Long.
- ◊ String.

Importante: As preferencias é a única forma de gardar datos persistentes na versión HTML do xogo.

Usar as preferencias

A forma de traballar con preferencias é a seguinte:

- Primeiro debemos obter un obxecto pertencente á clase Preferences:

```
Preferences prefs = Gdx.app.getPreferences("nomedearquivo");
```

Nota: Podemos ter múltiples arquivos de preferencia. Só temos que darlle un nome diferente.

- Agora podemos acceder ou escribir sobre dito arquivo de preferencias.

    ◊ Para escribir:

```
Preferences prefs = Gdx.app.getPreferences("nomedearquivo");

prefs.putString("nomeparametro1","valor1");
prefs.putInteger("nomeparametro2","valor2");

prefs.flush();
```

Como vemos temos que utilizar un nome para o dato que queremos gardar (nomeparametro).

Unha vez modificados os datos das preferencias, ditas modificación non se pasan a disco / tarxeta ata que chamemos ó **método flush**.

    ◊ Para ler:

```
Preferences prefs = Gdx.app.getPreferences("nomedearquivo");
String dato = prefs.getString("nomeparametro","valorpordefecto");
```

Como comentamos antes, cando gardamos un dato temos que indicar un nome para o parámetro. Dito nome é o que temos que utilizar cando recuperamos a información. Se dito valor non se atopa gardado no arquivo de preferencias podemos asinar un valor por defecto.

O recurso que utiliza Libgdx para gardar as preferencias na versión móbil son as SharedPreferences.

Almacenamento físico

As preferencias se gardan nun arquivo XML de nome o indicado na orde **Gdx.app.getPreferences("nomedearquivo")**.
Fisicamente se garda no cartafol:

- Windows: %UserProfile%/.prefs/NomeArquivo
- Linux and OS X: ~/.prefs/NomeArquivo

Un exemplo do código de dito arquivo que se atopa no cartafol .prefs:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="musica">false</entry>
</properties>
```

TAREFA 2.13 A FACER: Esta parte está asociada á realización dunha tarefa.

Datos

Dirección da wiki; https://github.com/libgdx/libgdx/wiki/File-handling

Imos dar o máis básico para o manexo de arquivos, tendo toda a información no enlace anterior.

Coas clases que nos proporciona o framework LIBGDX imos poder:

- Ler dun arquivo.
- Escribir nun arquivo.
- Copiar, mover ou borrar un arquivo.
- Listar arquivos e cartafois.
- Comprobar se un arquivo existe ou non.

O manexo de arquivos vai diferir da plataforma utilizada:

- DESKTOP (Windows, LINUX, MAC OS): Os arquivos poden ser referencias de forma relativa ó cartafol de traballo (onde está a aplicación que se executa). Os arquivos son de lectura-escritura por todas as aplicacións.

- Móbil (ANDROID): Aquí temos varias opcións:

  ◊ Os datos poden ser gardados internamente no cartafol assets, o cal é 'empaquetado' dentro da aplicación apk e non se poden modificar (só permisos de lectura).
  ◊ Os datos poden ser gardados internamente. Os datos poden ser lidos e modificados pero só ten acceso a aplicación que os crea.
  ◊ Os datos poden ser gardados na tarxeta sd externa. É necesario engadir os permisos correspondentes no AndroidManifiest.xml do proxecto Android:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.plategaxogo2d.angel.android"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19" />
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-feature android:required="false" android:name="android.hardware.sensor.accelerometer"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

- IOS: en ios todos os arquivos son accesibles.

- Javascript/WebGL: só accesibles os que se atopan no cartafol assets aínda que os navegadores modernos soportan os datos gardados internamente.

Un resume obtido do enlace anterior nos pode dar unha idea xeral en función da plataforma:

Tipo	Descrición,path e características	Desktop	Android	HTML5	IOS
Internal	Os arquivos de tipo internal son gardados no cartafol raíz da versión desktop (onde se atopa o executable), relativos ó cartafol assets na versión Android e relativos ó cartafol war/assets na versión GWT. Os arquivos aquí gardados son só de lectura.	Si	Si	Si	Si
Local	Os arquivos de tipo local son gardados no cartafol raíz da versión desktop (onde se atopa o executable), relativos ó cartafol interno na versión Android (as aplicacións Android crear un cartafol na tarxeta sd interna onde só elas poden acceder).	Si	Si	Non	Si
External	Os arquivos de tipo external son gardados relativos ó cartafol raíz da SD Card na versión móbil ou relativos ó cartafol home do usuario da versión Desktop (normalmente C:\Documents And Settings\Ususario). Pódense ler ou escribir.	Si	Si	Non	Si

No cadro orixinal veñen dous tipos máis pero que nos xogos non se usan.

Normalmente imos utilizar:

- Arquivos Internal: son todos os arquivos que se atopan no cartafol assets (gráficos, música,...). Son empaquetados xunto coa aplicación nun arquivo jar.
- Arquivos Local: para gardar pequenas cantidades de información, como o estado dun xogo ou as preferencias do mesmo.
- Arquivos External: para gardar grandes cantidades de información ou descargar información da web.

Comprobar se temos acceso ó almacenamento

Nunca está de mais comprobar se o almacenamento está dispoñible.

```
boolean isExtAvailable = Gdx.files.isExternalStorageAvailable();
```

```
boolean isLocAvailable = Gdx.files.isLocalStorageAvailable();
```

Acceder ós arquivos

A forma de acceder ós arquivos é cun obxecto da clase FileHandle.

Almacenamento interno:

```
FileHandle arquivo = Gdx.files.internal("data/arquivo.txt");
```

Facemos referencia ó cartafol assets da versión móbil.

Almacenamento externo:

```
FileHandle arquivo = Gdx.files.external("data/arquivo.txt");
```

Lectura e escritura de arquivos

Dependendo do tipo de información (texto ou binaria) podemos gardar e ler a información de dúas maneiras.

LECTURA:

Lectura de texto:

```
FileHandle arquivo = Gdx.files.external("data/arquivo.txt");
String texto = file.readString();
```

Nota:: le todo o arquivo.

Lectura de bytes (como unha imaxe):

```
FileHandle arquivo = Gdx.files.external("data/arquivo.txt");
byte[] bytes = arquivo.readBytes();
```

Nota:: le todo o arquivo.

ESCRITURA:

Escritura de texto:

```
FileHandle arquivo = Gdx.files.external("data/arquivo.txt");
arquivo.writeString("Texto", false);
```

O parámetro false indica se queremos engadir ó final do arquivo a nova cadea ou crear un arquivo novo.

Escritura de bytes:

```
FileHandle arquivo = Gdx.files.external("data/arquivo.txt");
arquivo.writeBytes(new byte[] { 20, 3, -2, 10 }, false);
```

TAREFA 2.14 A FACER: Esta parte está asociada á realización dunha tarefa.

Lectura e escritura de arquivos. Outra opción

Outra forma de acceder ó almacenamento externo é:

LECTURA:

Clases InputStreamReader e BufferedReader:

Exemplo de código para ler liña a liña un arquivo

```
BufferedReader in = null;
try {
in = new BufferedReader(new InputStreamReader(Gdx.files.external(
"NOME_ARQUIVO").read()));
String valor = "";
valor = String.valueOf(in.readLine());
} catch (Throwable e) {
// :( It's ok we have defaults
} finally {
try {
if (in != null)
in.close();
} catch (IOException e) {
}
}
```

ESCRITURA:

Clases OutputStreamWriter e BufferedWriter:

Exemplo de código para escribir a un arquivo

```
BufferedWriter out = null;
try {
out = new BufferedWriter(new OutputStreamWriter(Gdx.files.external(
"NOME_ARQUIVO").write(false)));
out.write("DATOS A ESCRIBIR");
out.newLine();
} catch (Throwable e) {
} finally {
try {
if (out != null)
out.close();
} catch (IOException e) {

}
}
```

Gardando o estado do meu xogo

O visto ata o de agora a forma máis fácil de gardar o estado un xogo é utilizando as preferencias.

Pero se necesitamos gardar grandes cantidades de datos pode ser interesante utilizar a serialización JSON que consiste en pasar os datos en forma de obxectos dunha clase a arquivos de texto que poidan ser gardados na SD ou internalmente.

A idea é que se eu quero gardar o estado do meu xogo podo gardar toda a información (ou polo menos a que necesito) do Alien (posición, num. vidas,...), posición dos inimigos, tempo que queda,...Case toda esta información está gardada en instancias de clases. Podemos pasar cada obxecto de cada clase a un formato JSON e gardalo a disco para posteriormente recuperalo.

Máis información en: https://github.com/libgdx/libgdx/wiki/Reading-&-writing-JSON

Exemplo de código:

```
Json json = new Json();
```

```
json.toJson(playerInfo, Gdx.files.external("myfile.json"));


//Lemos PlayerInfo dende o arquivo gardado
PlayerInfo savedInfo = json.fromJson(PlayerInfo.class, Gdx.files.external("myfile.json"));
```

Sendo playerInfo un obxecto da clase PayerInfo onde temos os datos a gardar.

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