LIBGDX Animacions

UNIDADE 3: Animacións

Sumario

Introdución

Nota: Esta explicación está Esta explicación está relacionada coa sección de Movendo os gráficos.

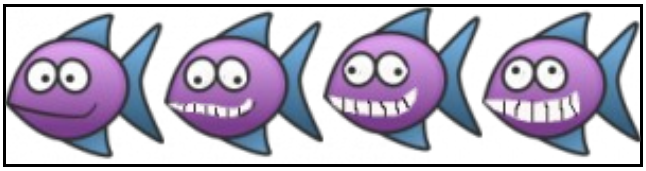
Información na wiki: https://github.com/libgdx/libgdx/wiki/2D-Animation Clase que se utiliza: Clase Animation.

O idea é moi sinxela. Consiste en debuxar de forma continuada un conxunto de TextureRegion que conforman a animación.

Despois mediante unha clase faremos que esas rexións vaian cambiando no método render a medida que pasa o tempo.

Proceso

Imos animar un peixe utilizando este gráfico:



O proceso é o seguinte:

- Cargamos nun obxecto Texture ou TextureRegion as imaxes que conforman a nosa animación.

  Nota: Lembrar que dita carga a podemos facer utilizando un TextureAtlas como explicamos neste punto. Nese caso o que obteremos será unha TextureRegion.

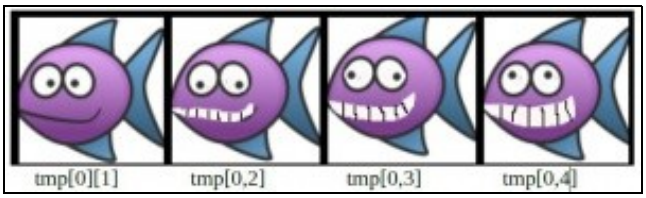
```
private Texture texturePeixe;
.............
texturePeixe = new Texture("LIBGDX_peixeanimado.png");
```

- Agora necesitamos dividir dita imaxe en anacos máis pequenos. Cada peixe ten un tamaño de 96x96 polo tanto imos dividir dita imaxe en 4 anacos. Isto o facemos có método split que devolve un array bidimensional de TextureRegion, no noso caso devolverá un array de 1 fila e 4 columnas.

```
TextureRegion[][] tmp = TextureRegion.split(texturePeixe,96,96);
```

No caso de ter como referencia do noso gráfico a TextureRegion o faríamos así:

```
TextureRegion[][] tmp = textureRegionPeixe.split(96,96);
```



Nota: Poderíamos ter máis imaxes (máis filas) e o método split ó chegar ó final pasaría a seguinte fila deixando unha altura de 96 píxeles (o indicado no método).

- A continuación temos que crear unha textureregion dunha dimensión (agora temos un array bidimensional) que sexa igual ó número de frames a amosar. No noso caso son 1x4 = 4 (unha fila e catro columnas).

```
int num_columnas = tmp[0].length;
int num_filas = tmp.length;
```

Outra forma de facelo:

Para calcular o número de filas e columnas mediante programación podemos utilizar este código: NUM_FILAS = ALTURA_TEXTURAREGION_CON_ANIMACION / 96 = 96 / 96 = 1 NUM_COLUMNA = ANCHURA_TEXTURAREGION_CON_ANIMACION / 96 = 384 / 96 = 4

Isto só é aplicable ás TextureRegion.

No noso caso:

```
int num_filas = textureRegionPeixe.getRegionHeight() / 96;
int num_columnas = textureRegionPeixe.getRegionWidth() / 96;
```

Nota: Para pasar dunha Texture a unha TextureRegion só temos que facer o new pasándolle como parámetro a Texture. Por exemplo: TextureRegion tr = new TextureRegion(texturePeixe);

Agora percorremos o array bidimensional e o pasamos a un array unidimensional:

```
TextureRegion[] framesanimacion = new TextureRegion[num_columnas*num_filas];

for(int fila=0; fila<num_filas;fila++){
for(int col=0; col<num_columnas;col++){
framesanimacion[fila*num_columnas+col]=tmp[fila][col];
}

}
```

- Unha vez pasado a un array unidimensional definimos un obxecto de clase Animation. A dito obxecto hai que pasarlle como parámetros as TextureRegion que van conformar a animación e o tempo que ten que pasar entre cada frame da animación:

```
import com.badlogic.gdx.graphics.g2d.Animation;

        .............
private Animation animPeixe;
        .............
animPeixe = new Animation(0.15f, framesanimacion);
```

- Agora no método render debemos de obter cada un dos frames da animación e debuxalo como o facemos sempre.

Para obter ese frame debemos utilizar un cronómetro (calquera contador de tempo que utilicemos no xogo) e debemos chamar ó método getKeyFrame, pasándolle como parámetro ese contador e unha variable booleana indicando se queremos que a animación se repita ou non.

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

crono+=Gdx.graphics.getDeltaTime();

batch.begin();
batch.draw(animPeixe.getKeyFrame(crono, true),0f,0f,96f,96f);
batch.end();

}
```

Exemplo de código

Deberedes de cambiar a clase co que inician as diferentes plataformas pola seguinte:

- Deberedes copiar o gráfico dos peixes animados ó cartafol assets do proxecto Android.
- Crear unha nova clase.

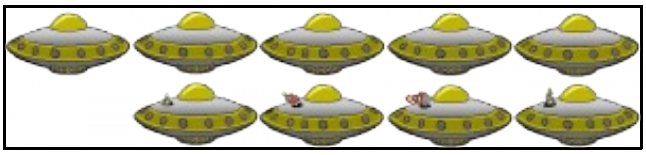
Código da clase Animacions
Obxectivo: Exemplo de animación

```
package com.plategaxogo2davanz.angel;

import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.Animation;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.graphics.g2d.TextureRegion;

public class Animacions extends ApplicationAdapter {
private SpriteBatch batch;
private Animation animPeixe;
private float crono;

@Override
public void create () {
batch = new SpriteBatch();
Texture texturePeixe = new Texture("LIBGDX_peixeanimado.png");

TextureRegion[][] tmp = TextureRegion.split(texturePeixe,96,96);

int num_columnas = tmp[0].length;
int num_filas = tmp.length;
TextureRegion[] framesanimacion = new TextureRegion[num_columnas*num_filas];

for(int fila=0; fila<num_filas;fila++){
for(int col=0; col<num_columnas;col++){
framesanimacion[fila*num_columnas+col]=tmp[fila][col];
}

}

animPeixe = new Animation(0.15f, framesanimacion);

}

@Override
public void render () {
```

```
Gdx.gl.glClearColor(1, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

crono+=Gdx.graphics.getDeltaTime();

batch.begin();
batch.draw(animPeixe.getKeyFrame(crono, true),0f,0f,96f,96f);
batch.end();

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

}
```

TAREFA OPTATIVA A FACER

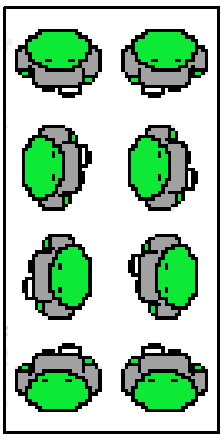
TAREFA 1 OPTATIVA A FACER: Modifica a clase AssetsXogo e referencia a nave mediante a clase Animated utilizando os seguintes gráficos de tamaño 200x110 pixeles:



Nota:

- Se mirastes o punto TextureAtlas deberedes crear outra vez o atlas engadindo a animación da nave.

TAREFA 2 OPTATIVA A FACER: Modifica a clase AssetsXogo e referencia o alien mediante a clase Animated utilizando os seguintes gráficos de tamaño 32x32 pixeles:



Nota:

- Se mirastes o punto TextureAtlas deberedes crear outra vez o atlas engadindo a animación do alien.
- Este gráfico consta de dous frames por cada dirección do alien. Polo tanto para debuxalo ides ter que, no método render, en función da dirección do alien, obter o frame da animación correspondente (esquerda, dereita, arriba ou abaixo). Para saber cara onde vai o alien podedes facer uso da velocidade (x,y).
- A forma máis sinxela é crear catro animacións, unha por cada movemento.

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