1 LIBGDX Fontes

UNIDADE 2: Fontes

Información da wiki: https://github.com/libgdx/libgdx/wiki/Bitmap-fonts

Normalmente nos xogos imos ter que escribir texto na pantalla.

Para facelo imos ter que utilizar a clase BitMapFont.

Empecemos indicando que é un BitMap. Para nos un BitMap é unha imaxe que representa unha letra. Dita letra (imaxe) pode ser escalada (aumentar ou reducir de tamaño).

Dependendo do tamaño inicial da letra, este escalado pode dar lugar a letras con dentes de serra. O ideal para que isto non suceda é ter o tamaño de letra igual de grande que na visualización para non ter que facer escalado de 'aumento'.

Inconveniente que teñen os BitMap: o tamaño que ocupan en disco. Se temos un xogo cunha única fonte ó mellor non é problema, pero se temos moitos tipos de fonte e estamos en Android si pode dar lugar a inconvenientes.

Para solucionalo mellor utilizar as fontes TrueType xa que ocupan moito menos espazo. Para aprender a utilizar este tipo de fontes podedes visitar este enlace: https://github.com/libgdx/libgdx/wiki/Gdx-freetype

Neste itinerario imos utilizar o BitMapFont por defecto que ten como inconveniente que non escala moi ben e pode dar lugar a texto cun aspecto malo (dentes de serra).

O proceso para utilizalas é moi sinxelo.

Preparación: Agora ides facer unha copia da clase RendererXogo, xa que imos modificala para amosarvos como se utilizan os BitMapFont. Premede co rato sobre a clase, botón dereito e escollede a opción Copy. Despois repetides a operación pero escolledes a opción Paste. Vos preguntará un nome para a clase. Indicade **UD2_6_RendererXogo**.

Código da clase UD2_6_RendererXogo
Obxectivo: utilizar BitMapFont.

```
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.OrthographicCamera;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.plategaxogo2d.modelo.Mundo;

/*
 * Comprobamos como utilizar as BitMapFont.
 */

public class UD2_6_RendererXogo {

    private OrthographicCamera camara2d;
    private SpriteBatch spritebatch;


    public UD2_6_RendererXogo(Mundo mundo) {

        camara2d = new OrthographicCamera();
        spritebatch = new SpriteBatch();

    }
```

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

    /**
     * Debuxa todos os elementos graficos da pantalla
     *
     * @param delta
     *            : tempo que pasa entre un frame e o seguinte.
     */
    public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);


spritebatch.begin();


spritebatch.end();

    }



    public void resize(int width, int height) {

camara2d.setToOrtho(false, Mundo.TAMANO_MUNDO_ANCHO,
Mundo.TAMANO_MUNDO_ALTO);
camara2d.update();

spritebatch.setProjectionMatrix(camara2d.combined);


    }

    public void dispose() {
        spritebatch.dispose();
    }


}
```

Modificade a clase PantallaXogo para que chame a esta clase. Ó iniciar o xogo aparece a pantalla de presentación. Debedes premer Novo Xogo para que chame a esta pantalla.

Clase BitMapFont. Por defecto utiliza unha fonte que ven integrada no jar do Libgdx. Ten un tamaño de 15 puntos e é Arial.

Métodos máis importantes:

- draw(Batch batch, java.lang.CharSequence str, float x, float y): Debuxa unha cadea na posición indicada. Este método está sobrecargado e ten varias alternativas.
- scale(float amount): Cambia a escala da fonte de forma relativa á escala actual.
- setScale(float scaleXY): Escala a fonte en ancho e alto á vez. Este método está sobrecargado e permite escalar de forma independente en cada eixe. Tamén comentar que dependendo da relación da aspecto do noso xogo, as letras poden saír deformadas por defecto tendo que escalas nos para unha visualización correcta.
- setColor(Color color): Cambia a cor do texto.
- dispose(): Libera a memoria utilizada polo BitMapFont.

Código da clase UD2_6_RendererXogo
Obxectivo: Exemplo de uso da clase BitMapFont.

```
public class UD2_6_RendererXogo {

    private OrthographicCamera camara2d;
    private SpriteBatch spritebatch;
    private BitmapFont bitMapFont;


    public UD2_6_RendererXogo(Mundo mundo) {

        camara2d = new OrthographicCamera();
        spritebatch = new SpriteBatch();

        bitMapFont = new BitmapFont();
    }



public OrthographicCamera getCamara2d(){
return camara2d;
}

    /**
     * Debuxa todos os elementos graficos da pantalla
     *
     * @param delta
     *            : tempo que pasa entre un frame e o seguinte.
     */
    public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);


spritebatch.begin();

        bitMapFont.setColor(Color.RED);
        bitMapFont.setScale(10);
bitMapFont.draw(spritebatch, "TEXTO ESCALADO A 10 VECES O SEU TAMAÑO", 0, 250);

        bitMapFont.setColor(Color.YELLOW);
        bitMapFont.setScale(0.5f, 2);
bitMapFont.draw(spritebatch, "TEXTO ESCALADO", 0, 350);

spritebatch.end();

    }



    public void resize(int width, int height) {

camara2d.setToOrtho(false, Mundo.TAMANO_MUNDO_ANCHO,
Mundo.TAMANO_MUNDO_ALTO);
camara2d.update();

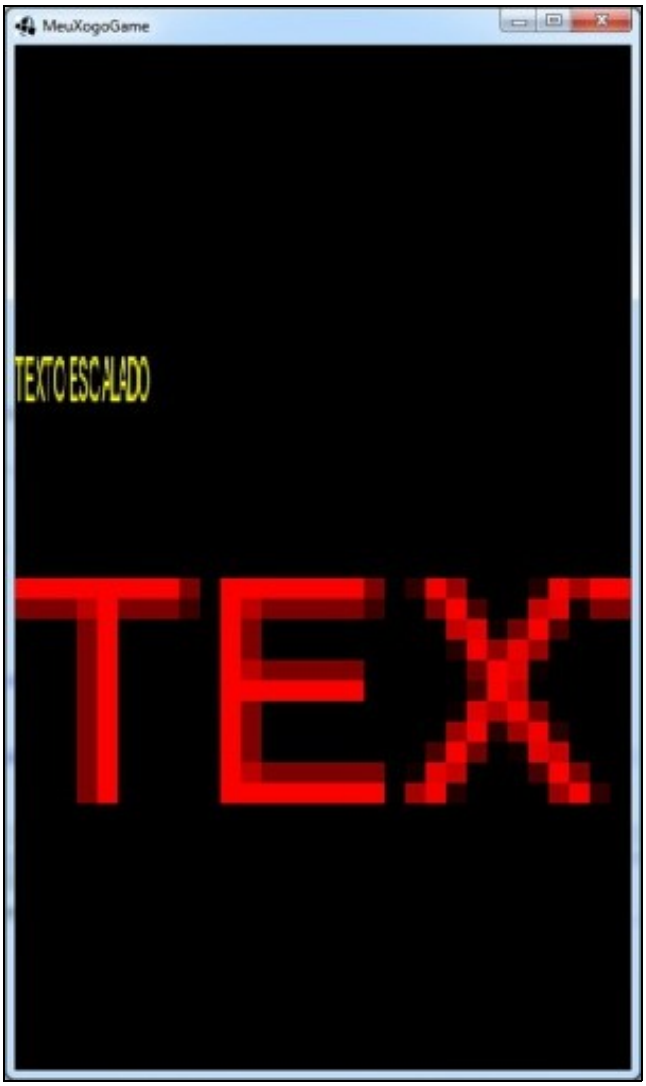
spritebatch.setProjectionMatrix(camara2d.combined);


    }

    public void dispose() {
        bitMapFont.dispose();
        spritebatch.dispose();
    }


}
```

Dará como resultado isto:



Nota: Loxicamente se a escala ou cor no van modificarse nunca é conveniente levar estas ordes ó constructor para só se executen unha vez.

Outro factor a ter en conta cando manexamos cadeas é o de optimizar o uso da memoria. Cando queremos concatenar dúas cadeas normalmente faríamos isto:

```
String cad1 = "CAD1";
        String cad2 = "CAD2";

        String cad3 = cad1 + cad2;
```

Esta forma de concatenar ten como inconveniente que crear unha nova zona de memoria para a nova cadea. Se o facemos dentro do método render, estaremos creando novas cadeas continuamente. Para evitalo debemos facer uso da clase StringBuilder.

Tedes unha explicación do seu uso neste enlace.

Un exemplo de uso:

Código da clase UD2_6_RendererXogo
Obxectivo: Usa a clase StringBuilder para crear novas cadeas e concatenar.

```
public class UD2_6_RendererXogo {

    private OrthographicCamera camara2d;
    private SpriteBatch spritebatch;
    private BitmapFont bitMapFont;
    private StringBuilder sbuffer;
```

```
    public UD2_6_RendererXogo(Mundo mundo) {

        camara2d = new OrthographicCamera();
        spritebatch = new SpriteBatch();

        bitMapFont = new BitmapFont();
        sbuffer = new StringBuilder();
        sbuffer.append("TEXTO ALEAOTORIO:");
    }



public OrthographicCamera getCamara2d(){
return camara2d;
}

    /**
     * Debuxa todos os elementos graficos da pantalla
     *
     * @param delta
     *            : tempo que pasa entre un frame e o seguinte.
     */
    public void render(float delta) {
Gdx.gl.glClearColor(0, 0, 0, 1);
Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);


spritebatch.begin();

                bitMapFont.setColor(Color.RED);
                bitMapFont.setScale(10);
bitMapFont.draw(spritebatch, "TEXTO ESCALADO A 10 VECES O SEU TAMAÑO", 0, 250);

                bitMapFont.setColor(Color.YELLOW);
                bitMapFont.setScale(0.5f, 2);
bitMapFont.draw(spritebatch, "TEXTO ESCALADO", 0, 350);

                bitMapFont.setColor(Color.WHITE);

                if (sbuffer.length()>50) {
                   sbuffer.setLength(0);        // Borramos
                   sbuffer.append("TEXTO ALEATORIO:");
                }
                sbuffer.append(MathUtils.random(9));
                bitMapFont.draw(spritebatch, sbuffer, 0, 50);

spritebatch.end();

    }



    public void resize(int width, int height) {

camara2d.setToOrtho(false, Mundo.TAMANO_MUNDO_ANCHO,
Mundo.TAMANO_MUNDO_ALTO);
camara2d.update();

spritebatch.setProjectionMatrix(camara2d.combined);


    }

    public void dispose() {
        bitMapFont.dispose();
        spritebatch.dispose();
    }


}
```

Nota: Volvede a modificar á clase PantallaXogo para que chame á clase RendererXogo.

1.1 Engadindo un cronómetro ó noso xogo

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

Para facelo imos definir unha constante de clase que gardará o tempo inicial (serán 120 segundos) e outra propiedade de clase de nome cronometro que, como di o seu nome, será un cronómetro :) .

Código da clase Mundo
Obxectivo: definimos un cronómetro.

```
public class Mundo {

        ..................
 private final static int TEMPO_INICIAL=120;
 private float cronometro;

        ..................
        public Mundo(){
           alien = new Alien(new Vector2(100,20), new Vector2(15,15),100);
           nave = new Nave(new Vector2(0,480),new Vector2(40,20),60);

           vehiculos = new Elementos(1.5f,3f,20);
           troncos = new Elementos(2f,5f,6);
           rocas = new Elementos(2f,5f,3);

           cronometro=TEMPO_INICIAL;

        }

public int getCronometro() {
return (int)cronometro;
}

public void setCronometro(float cronometro) {
this.cronometro = cronometro;
}

public void updateCronometro(float delta){
cronometro-=delta;
}

        ..................
}
```

Agora só temos que restar delta a cronometro no controlador do xogo.

Código da clase ControladorXogo
Obxectivo: actualizamos o cronómetro.

```
public void update(float delta) {

meuMundo.updateCronometro(delta);

controlarCoches(delta);
controlarRochas(delta);
controlarTroncos(delta);

controlarNave(delta);
controlarAlien(delta);

procesarEntradas();
```

}

TAREFA 2.11 A FACER: Esta parte está asociada á realización dunha tarefa.

1.2 Tarefas avanzadas

TAREFA AVANZADA: Esta parte está asociada á realización dunha tarefa avanzada. Fontes Avanzadas..

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