DESARROLLO DE JUEGOS PARA MÓVILES

1.- Introducción a la programación de Juegos

Existes dos formas generales para la realización de juegos en Android. Por una parte existe la posibilidad de programar utilizando los recursos que nos ofrece el propio Android y por otra parte existe la posibilidad de utilización de motores y/o librerías externas. En este curso vamos a realizar un juego utilizando los propios recursos de java y otro mediante la librería libgdx.

Antes de realizar un juego hay que tener claro que es lo que va a hacer nuestro juego y tendremos que pasar por una serie de fases hasta llegar a nuestro producto final. Podemos dividir un proyecto de un juego en las siguientes fases:

1.1.- Fase Inicial. Definición del juego.

No hay una regla fija. Ni estándares. Tenemos que crear una historia para nuestro juego, definir cómo va a funcionar. Podemos dibujar las pantallas y las transiciones entre ellas con una herramienta o a mano alzada, que nos servirán en un futuro para el diseño gráfico.

1.2.- Fase de Diseño

Utilizaremos diferentes herramientas como Gimp o Photoshop para dibujar los diferentes elementos gráficos que van a aparecer en nuestro juego. Personajes, fondos,...

1.3.- Fase de Desarrollo

Determinaremos que herramientas de desarrollo vamos a utilizar. El entorno los motores o librerías. Una vez realizado esto comenzaremos a convertir en realidad lo planteado en la fase de definición del juego.

1.4.- Fase de Pruebas

Probaremos que nuestro juego funciona y que lo hace como nos lo habíamos planteado inicialmente. Por otra parte lo probaremos en diferentes dispositivos para ver cuál es su resultado.

1.5.- Fase de empaquetado y distribución.

Crearemos nuestro APK y lo subiremos a GooglePlay.

A la hora de desarrollar un juego hay que tener en cuenta su **arquitectura**:

- Intentar dividir el problema en partes simples y reutilizables.
- Separar la lógica del juego de la arquitectura android. Separar la lógica del juego con la lógica de Android, para que se pueda hacer un juego o un programa java sin tener que reescribir todo el programa.

2.- Introducción programación de juegos Android

En los próximos puntos vamos a incorporar nuevos conceptos que tienen que ver con el desarrollo de juegos en Android como:

- **Game loop:** Son varias imágenes vista en frames por segundo para hacer la simulación de un video, en el juego 10 veces por segundo.
- **Motor de juegos**: El motor se encarga de actualizar el estado del juego y la impresión **(update y onDraw)**. Son los dos métodos más importantes del GameLoop.
- **Sprites**: Son los diferentes elementos u objetos que van a aparecer en nuestra pantalla. Pueden tener movimiento, sonidos, pueden desaparecer,...etc.
- Detección de **colisiones** entre Sprites. Cuando llegan al final de la pantalla o entre muñecos, cuando hago click sobre la pantalla y mato un personaje.
- **Recursos** del juego: imágenes y sonidos. Vamos a ver de dónde vamos a conseguir las imágenes. En drawable y en raw.

Además vamos a ver conceptos como:

- **SurfaceView** que es la vista a más bajo nivel y gestor de eventos holder.
- **Canvas:** Te permite dibujar animaciones sobre un lienzo.
- Gestión de recursos en android, **bitmaps y sonido**. Como metemos en memoria y lo cargamos.

3.- Canvas, View y SurfaceView

3.1.- View

En este punto vamos a imprimir una imagen directamente en la pantalla Android. Primero vamos a extender un ***View*** y después vamos a utilizar un objeto ***SurfaceView***, que es una manera más directa (bajo nivel). Vamos a utilizar *SurfaceView* porque queremos tener el control de la pantalla.

```
package com.example.amaia.juegoenandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new VistaJuego(this));
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.View;

public class VistaJuego extends View {
    private Bitmap bmp;

    public VistaJuego(Context context) {
        super(context);
        bmp = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawColor(Color.BLACK);
        canvas.drawBitmap(bmp, 10, 10, null);
    }
}
```

En la implementación de `View` el método ***onDraw*** es llamado directamente por el *holder* a través de un código que no podemos ver cuando se crea el `View`. El *holder* es el objeto que contiene al `View`.

3.2.- Canvas y Paint

Vamos a utilizar **canvas**, que nos permite dibujar sobre un lienzo o pizarra diferentes gráficos. Los métodos más utilizados en Canvas son los siguientes:

- `getWidth` y `getHeight`.
- `drawPoint(int x, int y, Paint)`; Paint es la clase que define las características de cómo pintar el objeto.
- `drawLines`, `drawRect()`, el estilo del Paint determinará si es o no relleno.
- `drawCircle()`
- `drawBitMap(bm, float topx, float topy, null)`
- `drawBitMap(bm, Rect xyorigen, Rect xydestino, null)`. En este método los rectángulos origen y destino no tienen por qué coincidir en tamaño, por lo que se hará un escalado, pero esta operación es muy costosa, así que mejor no utilizar. El rectángulo

`xyorigen` expresado según el sistema de coordenadas del bitmap y el rectángulo `xydestino` en el sistema de coordenadas de Canvas.

- Para cargar un bitmap utilizaremos:
  - `Bitmap bm=BitmapFactory.decodeStream(..);`

o

  - `Bitmap bm=BitmapFactory.decodeResource(..);`
- Podemos usar fuentes propias `TrueType` mediante la clase `Typeface`:
  `font=Typeface.createFormAsset(context.getAssets(), "nombre.ttf");`
  Después se utilizara en el Paint con **`setFont`**`(font)` junto a **`setTextSize`** determinando el aspecto final del texto. Se puede usar la llamada a **`setTextAlign`**`()` para cambiar el punto de referencia del cuadrado contenedor del texto. Y recoger el tamaño de dicho cuadrado con la llamada a: **`getTextBount`**`("txt", ci, cf, rect);`

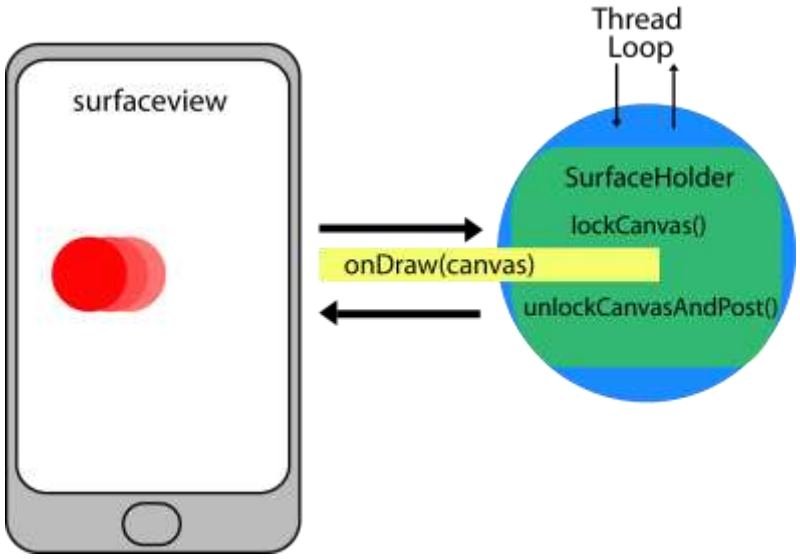
3.3.- SurfaceView

Cuando implementamos la versión que extiende de ***`SurfaceView`***, no se llama al método *`onDraw`*. Tenemos que llamarlo de una manera **explícita**. Para llamar al método *`onDraw`*, necesitamos que el objeto *`Canvas`* sea pasado como parámetro. Se puede pensar en el *Canvas* como una pizarra donde podemos pintar lo que queramos.

Para tener todo listo para pintar tenemos que crear un ***callback listener*** al ***holder***. En el método *`surfaceCreated`* (que es llamado cuando el *`View`* se crea), conseguimos el *`Canvas`* del *holder* y después llamamos al método *`onDraw`*.

Para conseguir el *`Canvas`*, utilizamos el método *`lockCanvas`* que bloquea el *`Canvas`* para evitar que nadie más dibuje cuando otro está dibujando. Una vez que se termina de dibujar, se desbloquea el *`Canvas`* llamando al método *`unlockCanvasAndPost`*.

Hay que tener en cuenta que durante el tiempo que se tiene un recurso bloqueado, estamos perdiendo rendimiento y por ello es importante minimizar el tiempo de bloqueo.




```
package com.example.amaia.juegoenandroid;

import android.content.Context;
```

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;

public class VistaJuego extends SurfaceView {

    private Bitmap bmp;
    private SurfaceHolder holder;


    public VistaJuego(Context context) {
        super(context);
        holder = getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                Canvas c = holder.lockCanvas(null);
                onDraw(c);
                holder.unlockCanvasAndPost(c);
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
            }
        });
        bmp = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        //super.onDraw(canvas);
        canvas.drawColor(Color.BLACK);
        canvas.drawBitmap(bmp, 10, 10, null);
    }
}
```

4.- Game Loop y Animación.

El "game loop" es la repetición de dos actividades principales;

1. Actualización de la física; se actualizan los datos del juego, como por ejemplo la posición "x" e "y" para un carácter (posiciones de los sprites, puntuación ...)
2. Dibujo; el dibujo de la imagen que se ve en la pantalla. Cuando este método es llamado repetidamente produce la percepción de ser una película o una animación.

Vamos a ejecutar el "game loop" en un *thread* diferente. En un hilo, hacemos las actualizaciones y los dibujos y en el hilo principal, manejamos los eventos como hacemos en una aplicación normal. El código que tenemos a continuación nos muestra esta implementación:

```
package com.example.amaia.juegoenandroid;

import android.graphics.Canvas;

public class GameLoopThread extends Thread {
    private VistaJuego view;
    private boolean running = false;

    public GameLoopThread (VistaJuego view){
        this.view = view;
    }

    public void setRunning (boolean run) {
        running = run;
    }

    @Override
    public void run() {
        while (running){
            Canvas c = null;
            try {
                c=view.getHolder()().lockCanvas();
                synchronized (view.getHolder()){
                    view.onDraw(c);
                }
            }finally {
                if (c!=null){
                    view.getHolder().unlockCanvasAndPost(c);
                }
            }
        }
    }
}
```

El campo "running" es un "flag" que hace que se pare el "game loop". Dentro de este loop, llamamos al método *onDraw*, como hemos visto en el punto anterior. En este caso, por simplicidad, hacemos el update y las actividades de dibujo en el método *onDraw*. Utilizamos la sincronización para evitar que cualquier otro hilo nos produzca conflicto cuando estemos dibujando.

En el SurfaceView añadimos un campo `int x` para mantener la coordenada "`x`" para dibujar la imagen en el método `onDraw`. En el método `onDraw` también incrementamos la posición "`x`" si no ha llegado al borde derecho.

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class VistaJuego extends SurfaceView {

    private Bitmap bmp;
    private SurfaceHolder holder;
```

```
    private GameLoopThread gameLoopThread;
    private int x = 0;

    public VistaJuego(Context context) {
        super(context);
        gameLoopThread = new GameLoopThread(this);
        holder = getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                boolean retry = true;
                gameLoopThread.setRunning(false);
                while (retry){
                    try{
                        gameLoopThread.join();
                        retry = false;
                    }catch (InterruptedException e){

                    }
                }
            }
        });
        bmp = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.BLACK);
        if (x < getWidth() - bmp.getWidth()){
            x++;
        }
        canvas.drawBitmap(bmp,x,10,null);
    }
}
```

Al ejecutar, obtenemos un resultado con una imagen animada que se mueve del lado izquierdo al derecho de la pantalla. Si lo miramos con atención podemos ver que la velocidad de la animación no es constante. Esto es porque cuando el "game loop" obtiene más tiempo de la CPU, la animación es más rápida. Podemos arreglar esto definiendo cuantos **FPS** (frames per second; imágenes por segundo) queremos en la aplicación.

Limitamos el dibujo a 10 FPS, que es 100 ms (milisegundos). Utilizamos el método sleep para que el tiempo restante sea 100ms. Si el bucle tarda más de 100ms, lo dormimos 10ms igualmente para evitar que la aplicación utilice demasiada CPU.

```
package com.example.amaia.juegoenandroid;

import android.graphics.Canvas;

public class GameLoopThread extends Thread {
    static final long FPS = 10;
    private VistaJuego view;
    private boolean running = false;

    public GameLoopThread (VistaJuego view){
        this.view = view;
    }

    public void setRunning (boolean run) {
        running = run;
    }

    @Override
    public void run() {
        long ticksPS = 1000 / FPS;
        long startTime;
        long sleepTime;
        while (running){
            Canvas c = null;
            startTime = System.currentTimeMillis();
            try {
                c=view.getHolder().lockCanvas();
                synchronized (view.getHandler()){
                    view.onDraw(c);
                }
            }finally {
                if (c!=null){
                    view.getHolder().unlockCanvasAndPost(c);
                }
            }
            sleepTime = ticksPS - (System.currentTimeMillis() -startTime);
            try {
                if (sleepTime > 0)
                    sleep(sleepTime);
                else
                    sleep(10);
            }catch (Exception e){}
        }
    }
}
```

En SurfaceView añadimos un campo ***xSpeed*** para mantener la dirección de la animación y en `onDraw` cambiamos la dirección cuando se llega a los bordes. Ahora la imagen va y viene del borde derecho al izquierdo indefinidamente.

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
```

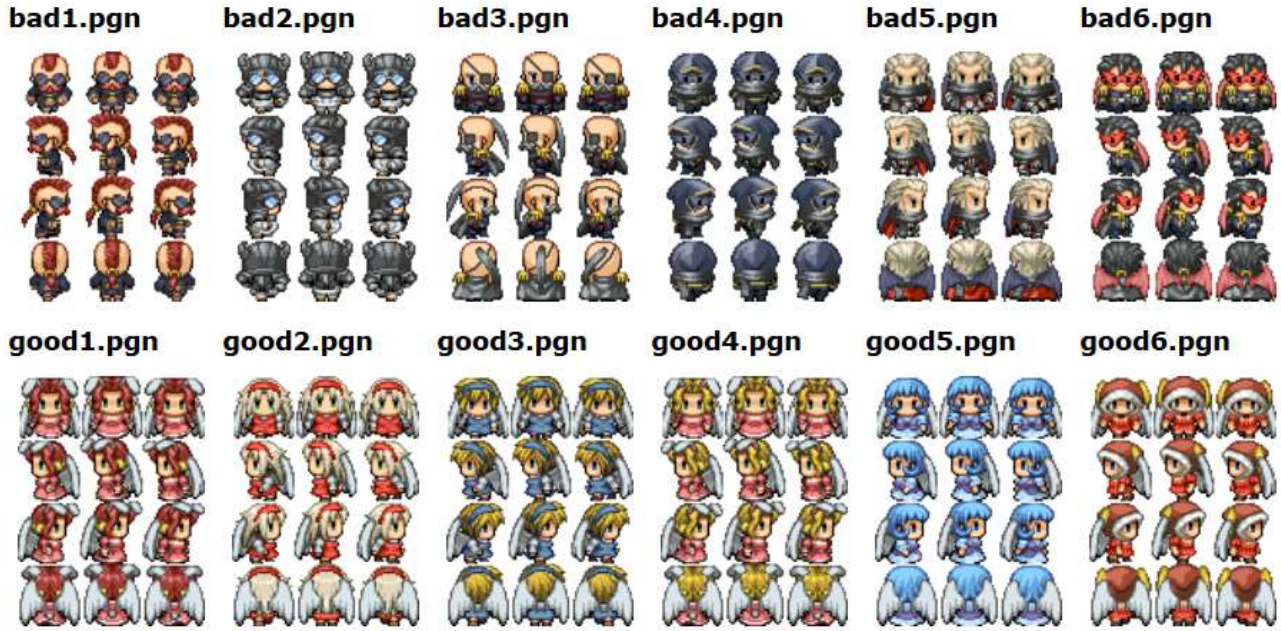
```
import android.view.SurfaceView;
import android.view.View;

public class VistaJuego extends SurfaceView {

    private Bitmap bmp;
    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private int x = 0;
    private int xSpeed=1;


    public VistaJuego(Context context) {
        super(context);
        gameLoopThread = new GameLoopThread(this);
        holder = getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                boolean retry = true;
                gameLoopThread.setRunning(false);
                while (retry){
                    try{
                        gameLoopThread.join();
                        retry = false;
                    }catch (InterruptedException e){

                    }
                }
            }
        });
        bmp = BitmapFactory.decodeResource(getResources(), R.mipmap.ic_launcher);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.BLACK);

        if (x == getWidth() - bmp.getWidth()) {
            xSpeed = -1;
        }
        if (x == 0) {
            xSpeed = 1;
        }
        x = x + xSpeed;
        canvas.drawBitmap(bmp, x, 10, null);
    }
}
```

5.- Sprites

Un sprite es una **imagen rasterizada 2D,** parcialmente transparente, que está normalmente situada encima de una imagen de fondo. Los sprites se utilizan en los videojuegos. Suele haber **más de un sprite** en la pantalla al mismo tiempo. Pueden **representar** o bien a **entidades** de inteligencia artificial o el **personaje** principal controlado por el usuario.

Un sprite tiene una **posición** (x,y) y una **velocidad** (x,y). Puede contener su propia **animación y sonido**.

A continuación se pueden ver los bitmaps de nuestros personajes (sprites):



Para hacer tus propios personajes se puede utilizar la siguiente página web:

http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator/#

Ahora que tenemos los bitmaps, vamos a crear nuestra primera clase Sprite:

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;


public class Sprite {
    private int x = 0;
    private int xSpeed = 5;
    private VistaJuego;
    private Bitmap bmp;


    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
    }
```

```
    private void update(){
        if (x > vistaJuego.getWidth() - bmp.getWidth() - xSpeed) {
            xSpeed = -5;
        }
        if (x + xSpeed < 0) {
            xSpeed = 5;
        }
        x = x + xSpeed;
    }

    public void onDraw (Canvas canvas){
        update();
        canvas.drawBitmap(bmp, x, 10, null);
    }
}
```

Como se ve, todo el **código referente a la posición y a la velocidad se ha movido** (incluyendo los bordes de la vista) a nuestra clase **`Sprite`**. También se ha movido el gráfico del sprite a esta clase y se han hecho unas modificaciones en la velocidad.

Ahora tenemos todo lo referente a nuestro sprite, dentro de esta clase.

En el **constructor** pedimos que nos pasen como parámetro la **vista** (vistaJuego) y el **bitmap** (bpm). La vista y el bitmap son necesarios para obtener los anchos para la detección de la colisión con el borde. Bmp también se utiliza para dibujar en el canvas.

Si en la clase **`VistaJuego`** eliminamos todo esto y añadimos una nueva instancia de Sprite obtendremos lo siguiente:

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class VistaJuego extends SurfaceView {

    private Bitmap bmp;
    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private Sprite sprite;

    public VistaJuego(Context context) {
        super(context);
        gameLoopThread = new GameLoopThread(this);
        holder = getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }
```

```
            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                boolean retry = true;
                gameLoopThread.setRunning(false);
                while (retry){
                    try{
                        gameLoopThread.join();
                        retry = false;
                    }catch (InterruptedException e){
                    }
                }
            }
        });
        bmp = BitmapFactory.decodeResource(getResources(), R.drawable.princesa);
        sprite = new Sprite(this, bmp);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.BLACK);
        sprite.onDraw(canvas);
    }
}
```

Si todo está bien, podremos ver el sprite de princesas, moviéndose de lado a lado de la pantalla.

Ahora vamos a modificarlo para dibujar una sola de las 12 imágenes. Queremos dibujar un personaje diferente cada vez para obtener una animación. Cada fila es un dibujo diferente. Para hacerlo más simple vamos a empezar con la segunda fila en la que el personaje parece que anda desde la derecha a la izquierda.

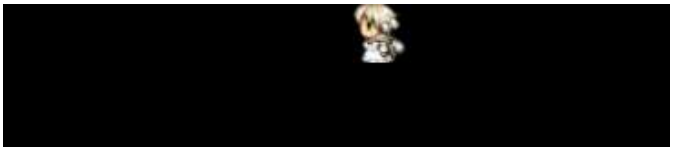
```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

public class Sprite {
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
```

```
    private int x = 0;
    private int y = 0;

    private int xSpeed = 5;

    private VistaJuego vistaJuego;
    private Bitmap bmp;

    private int currentFrame = 0;
    private int width;
    private int height;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;
    }

    private void update(){
        if (x > vistaJuego.getWidth() - width - xSpeed) {  //Para que llegue al final
            xSpeed = -5;
        }
        if (x + xSpeed < 0) {
            xSpeed = 5;
        }
        x = x + xSpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        int srcY = 1 * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
        //canvas.drawBitmap(bmp, x, 10, null);
    }
}
```

En el constructor calculamos el ancho (`width`) y el alto (`height`) del sprite.

En el método *`update`* fijamos donde se dibuja el sprite (`x = x + xSpeed`) y vamos cambiando la animación utilizada (`currentFrame = ++currentFrame % BMP_COLUMNS`). La variable `currentFrame` sólo puede tener los valores 0, 1 o 2.

En el método *`onDraw`* utilizamos el método *`canvas.drawBitmap(bmp, src, dst, null);`* *`src`* es el rectángulo fuente dentro del bitmap que va a ser dibujado dentro del rectángulo destino en el canvas (*`dst`*).



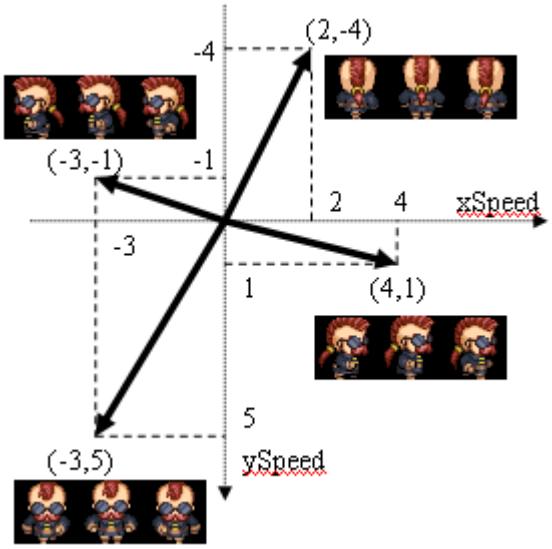
6.- Velocidad y animación de los sprites.

Ahora vamos a darle a nuestro sprite una velocidad "x" e "y" aleatoria (con el método `random`). Con esto, el sprite arrancará con una **dirección aleatoria**, que cambiará cada vez que llegue a un borde.

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;

public class Sprite {
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private int x = 0;
    private int y = 0;
    private int xSpeed = 5;
    private VistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
    private int width;
    private int height;
    private int ySpeed;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;
        Random rnd = new Random();
        xSpeed = rnd.nextInt(10) - 5;
        ySpeed = rnd.nextInt(10) - 5;
    }

    private void update(){
        if (x > vistaJuego.getWidth() - width - xSpeed || x + xSpeed < 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y > vistaJuego.getHeight() - height - ySpeed || y + ySpeed < 0) {
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        int srcY = 1 * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
        //canvas.drawBitmap(bmp, x, 10, null);
    }
}
```

La imagen de nuestro sprite contiene 4 animaciones con 3 imágenes cada una:

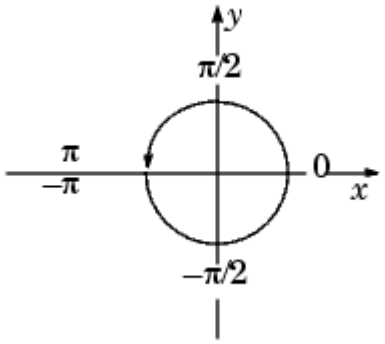
Dependiendo en qué dirección vaya el sprite tenemos que enseñar una animación diferente, por ejemplo;



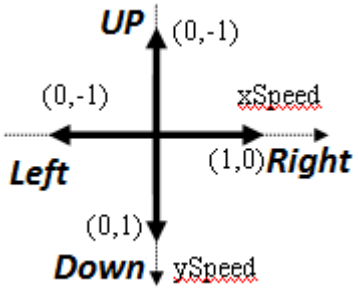
Necesitamos una función que tenga como parámetros "`xSpeed`" y "`ySpeed`" y que devuelva la fila que tenemos que utilizar para la animación [0,1,2,3]

Utilizaremos la función `Math.atan2(xSpeed, ySpeed)` para calcular la dirección del sprite en tiempo de ejecución.

`atan2(x,y)` nos da el ángulo en radianes en *double* desde (-PI a PI), pero necesitamos el resultado en *int* de 0 a 3 para saber que animación utilizar.



Observemos el gráfico que tenemos a continuación. En él escribo las coordenadas (x,y) para cada dirección en la que tenemos una animación: *Arriba (0,-1)*, *derecha (1,0)*, *abajo (0,1)* e *izquierda (0,-1)*.



Cuando el ángulo resultante está cerca de una de estas direcciones, queremos utilizar la animación correspondiente (up, down, left, right). Para esto utilizaremos la función `Math.round`.

Bueno, ahora viene la parte complicada que se resuelve con esta función a continuación y una constante array;

```
//direction = 0 up, 1 left, 2 down, 3 right,
//animation = 3 back, 1 left, 0 font, 2 right
int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};

private int getAnimationRow(){
    double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
    int direction = (int)Math.round(dirDouble) % BMP_ROWS;
    return DIRECTION_TO_ANIMATION_MAP[direction];
}
```

En esta tabla se muestra que se ha hecho en la función;

1. con `atan2(xSpeed,ySpeed)` se obtiene el ángulo en radianes desde (-PI a PI)
2. Se divide el ángulo por PI/2 y se obtiene un *double* de (-2 a 2)
3. Se suma 2 para cambiar el rango de (0 a 4)
4. Se utiliza % (modulo) para reducir el rango de (0 a 3) (ver que 0 y el 4 son la misma dirección)
5. Se Mapea cada dirección a la animación correcta utilizando el array { 3, 1, 0, 2 }

	x	y	atan2(x,y)	atan2(x,y)/(PI/2)	(atan2(x,y)/(PI/2)+2)%4	bmp row (from 0)
up	0	-1	PI or -PI	2 or -2	4 or 0	3
right	1	0	PI/2	1	3	2
down	0	1	0	0	2	0
left	-1	0	-PI/2	-1	1	1

Nota: aquí "x" y "y" son xSpeed y ySpeed.

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;
```

```
public class Sprite {
    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private int x = 0;
    private int y = 0;
    private int xSpeed = 5;
    private VistaJuego vistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
    private int width;
    private int height;
    private int ySpeed;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;

        //Random rnd = new Random();
        Random rnd = new Random(System.currentTimeMillis());
        xSpeed = rnd.nextInt(10) - 5;
        ySpeed = rnd.nextInt(10) - 5;
    }

    private void update(){
        if (x >= vistaJuego.getWidth() - width - xSpeed || x + xSpeed <= 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y >= vistaJuego.getHeight() - height - ySpeed || y + ySpeed <= 0) {
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        int srcY = getAnimationRow() * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
        //canvas.drawBitmap(bmp, x, 10, null);
    }

    //direction = 0 up, 1 left, 2 down, 3 right
    //animation = 3 back, 1 left, 0 front, 2 right
    private int getAnimationRow(){
        double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
        int direction = (int)Math.round(dirDouble) % BMP_ROWS;
        return DIRECTION_TO_ANIMATION_MAP[direction];
    }
}
```

7.- Trabajar con varios Sprites

Ahora vamos a ver la creación de varios sprites. Cada sprite con su propia posición, velocidad y orientación.

En `VistaJuego` vamos a cambiar la variable *`sprite`* por una Lista de *`sprite`*`s`.

```
private List<Sprite> sprites = new ArrayList<Sprite>();
```

Copiamos las imágenes de los caracteres en el directorio de recursos como hemos hecho con la imagen *princiesa.png*. Ahora creamos un nuevo `sprite` en el constructor, con cada imagen.

```
sprites.add(createSprite(R.drawable.princesa));
sprites.add(createSprite(R.drawable.malo));
sprites.add(createSprite(R.drawable.malo));
sprites.add(createSprite(R.drawable.malo2));
sprites.add(createSprite(R.drawable.malo2));
sprites.add(createSprite(R.drawable.malo3));
sprites.add(createSprite(R.drawable.malo3));
sprites.add(createSprite(R.drawable.malo4));
sprites.add(createSprite(R.drawable.malo4));
```

Utilizando el siguiente método:

```
private Sprite createSprite(int resouce) {
    bmp = BitmapFactory.decodeResource(getResources(), resouce);
    return new Sprite(this, bmp);
}
```

y actualizamos el método onDraw con

```
for (Sprite sprite:sprites) {
    sprite.onDraw(canvas);
}
```

Si lo ejecutamos, vamos a ver un efecto no muy bonito, debido al hecho de que todos los sprites empiezan en el mismo punto.

Corregimos esto fácilmente, añadiendo las dos siguientes líneas de código en el constructor del Sprite.

```
x = rnd.nextInt(vistaJuego.getWidth() - width);
y = rnd.nextInt(vistaJuego.getHeight()- height);
```

Si se vuelve a ejecutar, aparece un error.

Aquí inspeccionamos el valor de `vistaJuego.getWidth()` y encontramos que el *width* = 0. ¿Qué ocurrió? El problema es que hemos pedido el *view width* antes de que el *view* esté listo.

Si recuerdas, utilizamos el método *surfaceCreated* en el *Callback listener* para saber cuándo está listo *view*.

Podemos mover la creación de los sprites a este método y solucionar nuestro problema.

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class VistaJuego extends SurfaceView {

    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private List<Sprite> sprites = new ArrayList<Sprite>();

    public VistaJuego(Context context) {
        super(context);
        gameLoopThread = new GameLoopThread(this);
        holder = getHolder();
        holder.addCallback(new SurfaceHolder.Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                createSprites(); //Creamos los sprites
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                boolean retry = true;
                gameLoopThread.setRunning(false);
                while (retry){
                    try{
                        gameLoopThread.join();
                        retry = false;
                    }catch (InterruptedException e){

                    }
                }
            }
        });
    }

    private void createSprites(){
        sprites.add(createSprite(R.drawable.princesa));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo4));
```

```
        sprites.add(createSprite(R.drawable.malo4));
    }

    private Sprite createSprite(int resouce) {
        Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
        return new Sprite(this, bmp);
    }


    @Override
    protected void onDraw(Canvas canvas) {
        //super.onDraw(canvas);
        canvas.drawColor(Color.BLACK);
        for (Sprite sprite:sprites) {
            sprite.onDraw(canvas);
        }
    }
}
```

La clase Sprite poniendo una constante para la velocidad máxima MAX_SPEED, quedaría así:

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;


public class Sprite {
    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private static final int MAX_SPEED = 5;
    private int x = 0;
    private int y = 0;
    private int xSpeed;
    private VistaJuego vistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
    private int width;
    private int height;
    private int ySpeed;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;

        //Random rnd = new Random();
        Random rnd = new Random(System.currentTimeMillis());
        x = rnd.nextInt(vistaJuego.getWidth() - width);
        y = rnd.nextInt(vistaJuego.getHeight()- height);
        xSpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
        ySpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    }
```

```
    private void update(){
        if (x >= vistaJuego.getWidth() - width - xSpeed || x + xSpeed <= 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y >= vistaJuego.getHeight() - height - ySpeed || y + ySpeed <= 0) {
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        int srcY = getAnimationRow() * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
    }

    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    private int getAnimationRow(){
        double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
        int direction = (int)Math.round(dirDouble) % BMP_ROWS;
        return DIRECTION_TO_ANIMATION_MAP[direction];
    }
}
```

8.- Eventos táctiles y detección de colisiones de sprites.

En este apartado vamos a gestionar los eventos táctiles en la pantalla y ver si el eje de coordenadas (x,y) colisiona con alguno de nuestros sprites.

Lo primero que hacemos es añadir el siguiente método `onTouchEvent` en el *view* para gestionar cada toque en la pantalla.

Para cada sprite de la lista de sprites vamos a preguntar si existe una colisión con las coordenadas (x,y). Si existe la colisión, eliminamos el sprite de la lista de sprites. Iteramos la lista de sprites hacia atrás para evitar errores en la siguiente iteración, cuando hayamos eliminado el sprite.

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        for(int i = sprites.size() - 1; i >= 0; i--){
            Sprite = sprites.get(i);
            if (sprite.isCollition(event.getX(), event.getY())){
                sprites.remove(sprite);
            }
        }
        return super.onTouchEvent(event);
    }
```

La siguiente vez que el método *onDraw* dibuja la lista de sprites, los sprites eliminados de la lista, no serán dibujados. El efecto final es que el carácter desaparece debajo de nuestro dedo.

Para hacerlo **funcionar**, tenemos que implementar el método *isCollition* en la clase *Sprite*. Este método tiene que devolver `true` si las coordenadas (x,y) están dentro del área cubierta por el sprite.

```
public boolean isCollition(float x2, float y2) {
    return x2 > x && x2 < x + width && y2 > y && y2 < y + height;
}
```

Si `x2` no es mayor que `x`, esto significa que el toque en pantalla ha sido fuera del sprite. Si `x2` es mayor que `x+ancho`, esto significa que el toque está fuera a la derecha del sprite. Si `If x2 > x && x2 < x + width` es verdadero, significa que el toque ha sido en la misma columna pero hay que chequear si ha sido en la misma fila. `y2 > y && y2 < y + height` chequea si hay colisión en la fila de la misma manera.

Al ejecutarlo, se ve que cuando se hace click (o en el teléfono se da un toque) todos los sprites de la pantalla que coinciden con la posición, desaparecen. Un comportamiento más correcto para nuestra aplicación sería que sólo el sprite más arriba desaparezca. Podemos hacer esto añadiendo un `break` en

```
for(int i = sprites.size() - 1; i >= 0; i--){
    Sprite sprite = sprites.get(i);
    if (sprite.isCollition(event.getX(), event.getY())){
        sprites.remove(sprite);
        break;
    }
}
```

Por último hacemos la *sincronización*. Los eventos son gestionados en un hilo diferente al del dibujo y la actualización. Esto puede producir un conflicto si actualizamos los datos que están siendo dibujados al mismo tiempo.

La solución es sencilla; envolvemos todo el código dentro del método `onTouchEvent` con el mismo objeto *holder* que usamos en el game loop. Para recordarlo hemos utilizado;

```
c=view.getHolder().lockCanvas();
synchronized (view.getHolder()){
    view.onDraw(c);
}
```

y vamos a añadir

```
synchronized (getHolder()) {
    for (int i = sprites.size() - 1; i >= 0; i--) {
        Sprite sprite = sprites.get(i);
        if (sprite.isCollition(event.getX(), event.getY())) {
            sprites.remove(sprite);
            break;
        }
    }
}
```

Con esto evitamos un error que aparece aleatoriamente pero que es bastante molesto. Si lo ejecutamos otra vez, veremos que el problema persiste. En este caso el problema no es el loop.

Lo que ocurre es que los eventos son tan seguidos que parece que uno mata a varios. La solución es hacer que vaya más lento para no permitir más de un click cada 300 milisegundos.

```
if (System.currentTimeMillis() - lastClick > 300) {
    lastClick = System.currentTimeMillis();
```

Conseguimos esto envolviendo al evento handler con este if. Este utiliza una propiedad *lastClick* que evita que haya más de un click en 0.3 segundos.

Si lo ejecutamos ahora, vemos la diferencia.

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class VistaJuego extends SurfaceView {

    //private Bitmap bmp;
    private SurfaceHolder holder;
    private GameLoopThread;
    private List<Sprite> sprites = new ArrayList<Sprite>();
    private long lastClick;

    public VistaJuego(Context context) {
        super(context);
        gameLoopThread = new GameLoopThread(this);
        holder = getHolder();
        holder.addCallback(new Callback() {

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                createSprites(); //Creamos los sprites
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
            }
        });
        //bmp = BitmapFactory.decodeResource(getResources(), R.drawable.princesa);
        //sprite = new Sprite(this, bmp);
    }
```

```
    private void createSprites(){
        sprites.add(createSprite(R.drawable.princesa));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo4));
        sprites.add(createSprite(R.drawable.malo4));
    }

    private Sprite createSprite(int resouce) {
        Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
        return new Sprite(this, bmp);
    }


    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.BLACK);
        //Para dibujas más de un sprite
        for (Sprite sprite:sprites) {
            sprite.onDraw(canvas);
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (System.currentTimeMillis() - lastClick > 300) {
            lastClick = System.currentTimeMillis();
            synchronized (getHolder()) {
                for (int i = sprites.size() - 1; i >= 0; i--) {
                    Sprite sprite = sprites.get(i);
                    if (sprite.isCollition(event.getX(), event.getY())) {
                        sprites.remove(sprite);
                        break;
                    }
                }
            }
        }
        return true;
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;

public class Sprite {
    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private static final int MAX_SPEED = 5;
    private int x = 0;
```

```
    private int y = 0;
    private int xSpeed;
    private VistaJuego vistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
    private int width;
    private int height;
    private int ySpeed;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;

        Random rnd = new Random();
        x = rnd.nextInt(vistaJuego.getWidth() - width);
        y = rnd.nextInt(vistaJuego.getHeight()- height);
        xSpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
        ySpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    }

    private void update(){
        if (x >= vistaJuego.getWidth() - width - xSpeed || x + xSpeed <= 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y >= vistaJuego.getHeight() - height - ySpeed || y + ySpeed <= 0) {
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        int srcY = getAnimationRow() * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
    }

    //direction = 0 up, 1 left, 2 down, 3 right
    //animation = 3 back, 1 left, 0 front, 2 right
    private int getAnimationRow(){
        double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
        int direction = (int)Math.round(dirDouble) % BMP_ROWS;
        return DIRECTION_TO_ANIMATION_MAP[direction];
    }

    public boolean isCollition(float x2, float y2) {
        return x2 > x && x2 < x + width && y2 > y && y2 < y + height;
    }
}
```

GameLoopThread igual,....

```
package com.example.amaia.juegoenandroid;

import android.graphics.Canvas;
```

```
public class GameLoopThread extends Thread {
    static final long FPS = 10;
    private VistaJuego view;
    private boolean running = false;

    public GameLoopThread (VistaJuego view){
        this.view = view;
    }

    public void setRunning (boolean run) {
        running = run;
    }

    @Override
    public void run() {
        long ticksPS = 1000 / FPS;
        long startTime;
        long sleepTime;
        while (running){
            Canvas c = null;
            startTime = System.currentTimeMillis();
            try {
                c=view.getHolder().lockCanvas();
                synchronized (view.getHolder()){
                    view.onDraw(c);
                }
            }finally {
                if (c!=null){
                    view.getHolder().unlockCanvasAndPost(c);
                }
            }
            sleepTime = ticksPS - (System.currentTimeMillis() -startTime);
            try {
                if (sleepTime > 0)
                    sleep(sleepTime);
                else
                    sleep(10);
            }catch (Exception e){}
        }
    }
}
```

9.- Sprites temporales.

La mancha de sangre puede ser un *Sprite temporal*. Cuando matamos a un Sprite aparece y después de unos milisegundos esta sangre también desaparece. En este punto vamos a desarrollar esta funcionalidad utilizando un nuevo tipo de sprites.

Estos nuevos sprites no van a tener ni movimiento ni velocidad. Su posición va a ser definida en tiempo de construcción y vamos a tener uno cuando hagamos click. Estos sprites son, en muchos sentidos, mucho más simples que los primeros que creamos. La única nueva complejidad viene con el hecho de que tienen que desaparecer después de que pasa un tiempo.

Vamos a empezar con la creación de una nueva clase llamada `TempSprite`.

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;

import java.util.List;

public class TempSprite {
    private float x;
    private float y;
    private Bitmap bmp;
    private int life = 15;
    private List<TempSprite> temps;

    public TempSprite(List<TempSprite> temps, VistaJuego vistaJuego,
                      float x, float y, Bitmap bmp) {
        this.x = Math.min(Math.max(x - bmp.getWidth() /2, 0),
                vistaJuego.getWidth() - bmp.getWidth());
        this.y = Math.min(Math.max(y - bmp.getHeight() /2,0),
                vistaJuego.getHeight() - bmp.getHeight());
        this.temps = temps;
        this.bmp = bmp;
    }

    public void onDraw (Canvas canvas){
        update();
        canvas.drawBitmap(bmp, x, y, null);
    }

    private void update (){
        if (--life < 1){
            temps.remove(this);
        }
    }
}
```

Para hacer que el sprite desaparezca, introducimos un nuevo campo llamado ***life***. Va a contener el número de *ticks* en los que el sprite va a estar vivo. Un *tick* es una marca para cada vez que se llama al método `update` dentro del *game loop*. Las imágenes por segundo (FPS) es una marca para cada llamada al método `onDraw` en un segundo. En nuestro juego FPS y los ticks de update están sincronizados pero no tiene por qué ser así.

Cada vez que se llama al método `update`, disminuimos el valor de la vida en uno y cuando es menor que 1, eliminamos nuestro sprite de la lista de sprites temporales *temps*. La lista *temps* guarda todos los sprites temporales y es pasado como parámetro por el *view* en el constructor de *TempSprite*.

El constructor recibe además como parámetros una referencia al *view*, la posición (x,y) de donde hacemos click y el bitmap. La posición (x,y) se ajusta en el constructor teniendo en cuenta varios factores;

- El centro del bitmap de la sangre tiene que estar en el eje de coordenadas (`x,y`). Si no lo está, vamos a tener la sensación de que la sangre está abajo a la izquierda de donde hacemos click.
  Conseguimos esto con: `x - bmp.getWidth() /2`
- (`x,y`) no pueden salir de la pantalla. Si ocurre tendremos comportamientos inesperados y errores. Por lo tanto:

1. *x* e *y* tienen que ser mayor que 0. `Math.max(x - bmp.getWidth() /2, 0)`
2. *x* tiene que ser menor que `vistaJuego.getWidth()`, descontando el bitmap *y* tiene que ser menor que `gameView.getHeight()` descontando la altura del bitmap. `Math.min(... , vistaJuego.getWidth() - bmp.getWidth());`

```
this.x = Math.min(Math.max(x - bmp.getWidth() /2, 0),
        vistaJuego.getWidth() - bmp.getWidth());
this.y = Math.min(Math.max(y - bmp.getHeight() /2,0),
        vistaJuego.getHeight() - bmp.getHeight());
```

El bitmap se imprime completo en la posición calculada de (`x,y`) en el método *onDraw*.

Todavía hacen falta unos cambios más;

1. Hay que copiar la siguiente imagen a la carpeta recursos: `blood1.png`

2. Añadir la siguiente línea de código como última línea en el constructor de *view*.

```
bmpBlood = BitmapFactory.decodeResource(getResources(), R.drawable.blood1);
```

3. Incluir el campo `temps`

```
private List<TempSprite> temps = new ArrayList<TempSprite>();
```

4. Y añadir los dibujos de los sprites temps al método *onDraw*.

```
for (int i = temps.size() - 1; i >= 0; i--){
    temps.get(i).onDraw(canvas);
}
```

Iteramos hacia atrás para evitar errores cuando los sprites sean eliminados. Escribimos este código antes de dibujar los otros sprites para dar la sensación de que la sangre está en un segundo plano.

5. Y por último añadimos

```
temps.add(new TempSprite(temps, this, x, y, bmpBlood));
```

justo después de que cada sprite sea eliminado en el método `onTouchEvent`.

```
package com.example.amaia.juegoenandroid;

import android.graphics.Canvas;

public class GameLoopThread extends Thread {
    static final long FPS = 10;
    private VistaJuego view;
    private boolean running = false;

    public GameLoopThread (VistaJuego view){
        this.view = view;
    }

    public void setRunning (boolean run) {
        running = run;
    }
```

```
    @Override
    public void run() {
        //super.run();
        long ticksPS = 1000 / FPS;
        long startTime;
        long sleepTime;
        while (running){
            Canvas c = null;
            startTime = System.currentTimeMillis();
            try {
                c=view.getHolder().lockCanvas();
                synchronized (view.getHolder()){
                    view.onDraw(c);
                }
            }finally {
                if (c!=null){
                    view.getHolder().unlockCanvasAndPost(c);
                }
            }
            sleepTime = ticksPS - (System.currentTimeMillis() -startTime);
            try {
                if (sleepTime > 0)
                    sleep(sleepTime);
                else
                    sleep(10);
            }catch (Exception e){}
        }
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new VistaJuego(this));
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;

public class Sprite {
    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private static final int MAX_SPEED = 5;
    private int x = 0;
```

```
    private int y = 0;
    private int xSpeed;
    private VistaJuego vistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
    private int width;
    private int height;
    private int ySpeed;

    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;

        Random rnd = new Random();
        x = rnd.nextInt(vistaJuego.getWidth() - width);
        y = rnd.nextInt(vistaJuego.getHeight()- height);
        xSpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
        ySpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    }

    private void update(){
        if (x >= vistaJuego.getWidth() - width - xSpeed || x + xSpeed <= 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y >= vistaJuego.getHeight() - height - ySpeed || y + ySpeed <= 0) {
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        //int srcY = 1 * height;
        int srcY = getAnimationRow() * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
    }

    private int getAnimationRow(){
        double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
        int direction = (int)Math.round(dirDouble) % BMP_ROWS;
        return DIRECTION_TO_ANIMATION_MAP[direction];
    }

    public boolean isCollition(float x2, float y2) {
        return x2 > x && x2 < x + width && y2 > y && y2 < y + height;
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;

import java.util.List;

public class TempSprite {
    private float x;
    private float y;
    private Bitmap bmp;
    private int life = 15;
    private List<TempSprite> temps;

    public TempSprite(List<TempSprite> temps, VistaJuego vistaJuego,
                      float x, float y,Bitmap bmp) {

        this.x = Math.min(Math.max(x - bmp.getWidth() /2, 0),
                vistaJuego.getWidth() - bmp.getWidth());
        this.y = Math.min(Math.max(y - bmp.getHeight() /2,0),
                vistaJuego.getHeight() - bmp.getHeight());
        this.temps = temps;
        this.bmp = bmp;
    }

    public void onDraw (Canvas canvas){
        update();
        canvas.drawBitmap(bmp, x, y, null);
    }

    private void update (){
        if (--life < 1){
            temps.remove(this);
        }
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class VistaJuego extends SurfaceView {

    //private Bitmap bmp;
    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private List<Sprite> sprites = new ArrayList<Sprite>();
    private List<TempSprite> temps = new ArrayList<TempSprite>();
    private long lastClick;
    private Bitmap bmpBlood;
```

```
public VistaJuego(Context context) {
    super(context);
    gameLoopThread = new GameLoopThread(this);
    //holder = getHolder();
    //holder.addCallback(new Callback() {
    getHolder().addCallback(new Callback(){

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            createSprites(); //Creamos los sprites
            gameLoopThread.setRunning(true);
            gameLoopThread.start();
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
                                   int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
        }
    });
    bmpBlood = BitmapFactory.decodeResource(getResources(),R.drawable.blood1);
}

private void createSprites(){
    sprites.add(createSprite(R.drawable.princesa));
    sprites.add(createSprite(R.drawable.malo));
    sprites.add(createSprite(R.drawable.malo));
    sprites.add(createSprite(R.drawable.malo2));
    sprites.add(createSprite(R.drawable.malo2));
    sprites.add(createSprite(R.drawable.malo3));
    sprites.add(createSprite(R.drawable.malo3));
    sprites.add(createSprite(R.drawable.malo4));
    sprites.add(createSprite(R.drawable.malo4));
}

private Sprite createSprite(int resouce) {
    Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
    return new Sprite(this, bmp);
}


@Override
protected void onDraw(Canvas canvas) {
    //super.onDraw(canvas);
    canvas.drawColor(Color.BLACK);
    //Para dibujas más de un sprite
    for (Sprite sprite:sprites) {
        sprite.onDraw(canvas);
    }
    for (int i = temps.size() - 1; i >= 0; i--){
        temps.get(i).onDraw(canvas);
    }
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (System.currentTimeMillis() - lastClick > 300) {
        lastClick = System.currentTimeMillis();
        float x = event.getX();
```

```
            float y = event.getY();
            synchronized (getHolder()) {
                for (int i = sprites.size() - 1; i >= 0; i--) {
                    Sprite sprite = sprites.get(i);
                    //if (sprite.isCollition(event.getX(), event.getY())) {
                    if (sprite.isCollition(x ,y )) {
                            sprites.remove(sprite);
                            temps.add(new TempSprite(temps, this, x, y, bmpBlood));
                            break;
                    }
                }
            }
        }
        return true;
    }
}
```

10.- Sonido en Sprites

Cuando *matemos* a uno de los malos vamos a utilizar un `soundPool`.

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.media.AudioManager;
import android.media.SoundPool;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class VistaJuego extends SurfaceView {

    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private List<Sprite> sprites = new ArrayList<Sprite>();
    private List<TempSprite> temps = new ArrayList<TempSprite>();
    private long lastClick;
    private Bitmap bmpBlood;

    //Para el sonido:
    //Declaramos las variables
    private SoundPool soundPool;
    private int idExplosion;

    //Para el control de final de juego
    private boolean gameOver = false;

    public VistaJuego(Context context) {
        super(context);
```

```
        //Cargamos sonido en memoria
        soundPool = new SoundPool(5,AudioManager.STREAM_MUSIC,0);
        idExplosion = soundPool.load(this.getContext(),R.raw.explosion, 0);

        gameLoopThread = new GameLoopThread(this);
        getHolder().addCallback(new Callback(){

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                createSprites(); //Creamos los sprites
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                                       int width, int height) {
            }


            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                boolean retry = true;
                gameLoopThread.setRunning(false);
                while (retry){
                    try{
                        gameLoopThread.join();
                        retry = false;
                    }catch (InterruptedException e){
                    }
                }
            }
        });
        bmpBlood = BitmapFactory.decodeResource(getResources(),
                                                    R.drawable.blood1);
    }

    private void createSprites(){
        sprites.add(createSprite(R.drawable.princesa));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo2));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo3));
        sprites.add(createSprite(R.drawable.malo4));
        sprites.add(createSprite(R.drawable.malo4));
    }

    private Sprite createSprite(int resouce) {
        Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
        return new Sprite(this, bmp);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        //super.onDraw(canvas);
        canvas.drawColor(Color.BLACK);

        //Para dibujas más de un sprite
        for (Sprite sprite:sprites) {
            sprite.onDraw(canvas);
        }
```

```
        for (int i = temps.size() - 1; i >= 0; i--){
            temps.get(i).onDraw(canvas);
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (System.currentTimeMillis() - lastClick > 300) {
            lastClick = System.currentTimeMillis();
            float x = event.getX();
            float y = event.getY();
            synchronized (getHolder()) {
                for (int i = sprites.size() - 1; i >= 0; i--) {
                    Sprite sprite = sprites.get(i);
                    if (sprite.isCollition(x ,y )) {
                            sprites.remove(sprite);
                            temps.add(new TempSprite(temps, this,
                                                    x, y, bmpBlood));

                            //Para el sonido
                            if (i==0){
                                gameOver=true;
                            }
                            else{
                                soundPool.play(idExplosion, 1,1,0,0,1);
                            }
                            break;
                    }
                }
            }
        }
        return true;
    }
}
```

En nuestro ejemplo si el identificativo del Sprite no es un 0, que es la princesa, reproducimos el sonido de una explosión. Vamos a ir definiendo una variable `gameOver` que nos va a determinar posteriormente cuando termina el juego.

11.- Marcador

Vamos a añadir un marcador para que se vea los puntos que llevamos y los malos que tenemos que matar, y para recordarnos que el juego termina cuando tenemos 40 malos o matamos a la princesa.

Antes de nada vamos a quitar el **ActionBar** para tener más espacio en nuestra pantalla. Esto lo haremos en el fichero *styles.xml* (res/values/styles.xml), cambiando

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

Por:

```
 <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
```

Además vamos a dejar espacio en la parte superior para que NO se nos metan los Sprites.

En la clase `Sprite`, en el *constructor*, tras la línea:

```
y = rnd.nextInt(vistaJuego.getHeight()- height);
```

Añadiremos:

```
if (y < (vistaJuego.getHeight() / 10))
    y =y + (vistaJuego.getHeight() / 10);
```

Y sustituiremos en el método `update`:

```
if (y >= vistaJuego.getHeight() - height - ySpeed || y + ySpeed <= 0) {
    ySpeed = -ySpeed;
}
```

Por:

```
if (y >= vistaJuego.getHeight() - height - ySpeed ||
        y + ySpeed <= (vistaJuego.getHeight() / 10)){
    ySpeed = -ySpeed;
}
```

Visualizamos los puntos (cada vez que matamos un malo) y el número de malos, para ello en el método `onDraw` de la clase `VistaJuego` pondremos:

```
Paint p = new Paint();
p.setColor(Color.RED);
p.setTextSize(canvas.getWidth() / 10);
p.setTextAlign(Paint.Align.RIGHT);
String text = "Score: "+ score;
Log.i("probando score", score+"");
canvas.drawText(text, canvas.getWidth(), (float)(canvas.getHeight() * 0.1
),p);
text = "Malos: "+ malos;
Log.i("PROBANDO MALOS:", malos+"");
p.setTextAlign(Paint.Align.LEFT);
canvas.drawText(text, 0, (float)(canvas.getHeight()*0.1), p);
```

Quedando las clases `Sprite` y `VistaJuego` de la siguiente manera:

```
package com.example.amaia.juegoenandroid;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Rect;

import java.util.Random;

public class Sprite {
    //direction = 0 up, 1 left, 2 down, 3 right,
    //animation = 3 back, 1 left, 0 font, 2 right
    int [] DIRECTION_TO_ANIMATION_MAP = {3, 1, 0, 2};
    private static final int BMP_ROWS = 4;
    private static final int BMP_COLUMNS = 3;
    private static final int MAX_SPEED = 5;
    private int x = 0;
    private int y = 0;
    private int xSpeed;
    private VistaJuego vistaJuego;
    private Bitmap bmp;
    private int currentFrame = 0;
```

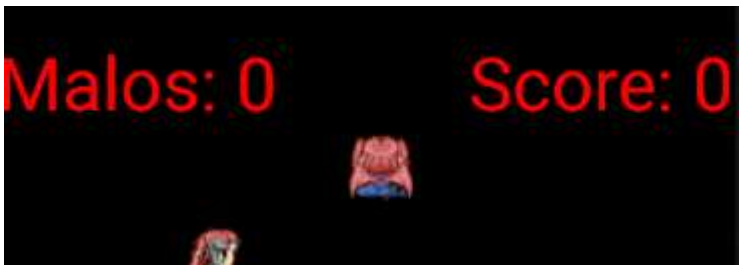
```
    private int width;
    private int height;
    private int ySpeed;


    public Sprite(VistaJuego vistaJuego,Bitmap bmp) {
        this.vistaJuego = vistaJuego;
        this.bmp = bmp;
        this.width = bmp.getWidth() / BMP_COLUMNS;
        this.height = bmp.getHeight() / BMP_ROWS;

        Random rnd = new Random();
        x = rnd.nextInt(vistaJuego.getWidth() - width);
        y = rnd.nextInt(vistaJuego.getHeight()- height);

        if (y < (vistaJuego.getHeight() / 10))
            y =y + (vistaJuego.getHeight() / 10);

        xSpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
        ySpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
    }

    private void update(){
        if (x >= vistaJuego.getWidth() - width - xSpeed || x + xSpeed <= 0) {
            xSpeed = -xSpeed;
        }
        x = x + xSpeed;
        if (y >= vistaJuego.getHeight() - height - ySpeed ||
                y + ySpeed <= (vistaJuego.getHeight() / 10)){
            ySpeed = -ySpeed;
        }
        y = y + ySpeed;
        currentFrame = ++currentFrame % BMP_COLUMNS;
    }

    public void onDraw (Canvas canvas){
        update();
        int srcX = currentFrame * width;
        //int srcY = 1 * height;
        int srcY = getAnimationRow() * height;
        Rect src = new Rect (srcX, srcY, srcX + width, srcY + height);
        Rect dst = new Rect (x, y, x + width, y + height);
        canvas.drawBitmap(bmp, src, dst, null);
        //canvas.drawBitmap(bmp, x, 10, null);
    }

    //direction = 0 up, 1 left, 2 down, 3 right
    //animation = 3 back, 1 left, 0 front, 2 right
    private int getAnimationRow(){
        double dirDouble = (Math.atan2(xSpeed, ySpeed)/ (Math.PI / 2)+ 2 );
        int direction = (int)Math.round(dirDouble) % BMP_ROWS;
        return DIRECTION_TO_ANIMATION_MAP[direction];
    }

    public boolean isCollition(float x2, float y2) {
        return x2 > x && x2 < x + width && y2 > y && y2 < y + height;
    }
}
```

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.media.AudioManager;
import android.media.SoundPool;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class VistaJuego extends SurfaceView {

    //private Bitmap bmp;
    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;
    private List<Sprite> sprites = new ArrayList<Sprite>();
    private List<TempSprite> temps = new ArrayList<TempSprite>();
    private long lastClick;
    private Bitmap bmpBlood;

    //Para el sonido:Declaramso las variables
    private SoundPool soundPool;
    private int idExplosion;

    //Para controlar el final de la partida
    private boolean gameOver;
    //Para el marcador
    private int score = 0;
    private int malos;


    public VistaJuego(Context context) {
        super(context);

        //Cargamos sonido en memoria
        soundPool = new SoundPool(5,AudioManager.STREAM_MUSIC,0);
        idExplosion = soundPool.load(this.getContext(),R.raw.explosion, 0);

        gameLoopThread = new GameLoopThread(this);
        //holder = getHolder();
        //holder.addCallback(new Callback() {
        getHolder().addCallback(new Callback(){

            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                //Canvas c = holder.lockCanvas(null);
                //onDraw(c);
                //holder.unlockCanvasAndPost(c);
                createSprites(); //Creamos los sprites
                gameLoopThread.setRunning(true);
                gameLoopThread.start();
            }
```

```
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
                                   int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            boolean retry = true;
            gameLoopThread.setRunning(false);
            while (retry){
                try{
                    gameLoopThread.join();
                    retry = false;
                }catch (InterruptedException e){
                }
            }

        }
    });
    bmpBlood = BitmapFactory.decodeResource(getResources(),
                               R.drawable.blood1);
}

private void createSprites(){
    sprites.add(createSprite(R.drawable.princesa));
    sprites.add(createSprite(R.drawable.malo));
    sprites.add(createSprite(R.drawable.malo));
    sprites.add(createSprite(R.drawable.malo2));
    sprites.add(createSprite(R.drawable.malo2));
    sprites.add(createSprite(R.drawable.malo3));
    sprites.add(createSprite(R.drawable.malo3));
    sprites.add(createSprite(R.drawable.malo4));
    sprites.add(createSprite(R.drawable.malo4));
}

private Sprite createSprite(int resouce) {
    Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
    return new Sprite(this, bmp);
}


@Override
protected void onDraw(Canvas canvas) {
    //super.onDraw(canvas);
    canvas.drawColor(Color.BLACK);

    //Para dibujas más de un sprite
    for (Sprite sprite:sprites) {
        sprite.onDraw(canvas);
    }
    for (int i = temps.size() - 1; i >= 0; i--){
        temps.get(i).onDraw(canvas);
    }

    //Para visualizar los puntos y el número de malos
    Paint p = new Paint();
    p.setColor(Color.RED);
    p.setTextSize(canvas.getWidth() / 10);
    p.setTextAlign(Paint.Align.RIGHT);
    String text = "Score: "+ score;
    canvas.drawText(text, canvas.getWidth(),
                        (float)(canvas.getHeight() * 0.1 ),p);
    text = "Malos: "+ malos;
```

```
            p.setTextAlign(Paint.Align.LEFT);
            canvas.drawText(text, 0, (float)(canvas.getHeight()*0.1), p);
        }

        @Override
        public boolean onTouchEvent(MotionEvent event) {
            if (System.currentTimeMillis() - lastClick > 300) {
                lastClick = System.currentTimeMillis();
                float x = event.getX();
                float y = event.getY();
                synchronized (getHolder()) {
                    for (int i = sprites.size() - 1; i >= 0; i--) {
                        Sprite sprite = sprites.get(i);
                        //if (sprite.isCollition(event.getX(), event.getY())) {
                        if (sprite.isCollition(x ,y )) {
                               sprites.remove(sprite);
                               temps.add(new TempSprite(temps, this, x, y, bmpBlood));
                                //Para el sonido
                                if (i==0){
                                    gameOver=true;
                                }
                                else{
                                    soundPool.play(idExplosion, 1,1,0,0,1);
                                }
                                break;
                        }
                    }
                }
            }
            return true;
        }
    }
```

De momento, en la parte superior de la pantalla únicamente veremos las puntuaciones y los malos, pero NO se actualiza.



12.- Programamos la lógica del juego

Cada segundo vamos a crear **XX** malos. Cuando el marcador sea menor que 10 (<10) 1 malo, a partir de 10 (>=10) 2 malos, a partir de 20 (>20) 3 malos,... El juego terminará cuando superemos 40 malos o matemos a la princesa.

El código necesario a incluir en la clase `VistaJuego`, seria:

En el método `onDraw`:

```
protected void onDraw(Canvas canvas) {

    ...
```

```
        //Creamos un nuevo malo cada tiempoNuevoMalo FPS y cuantos más
        //puntos tengamos más rápido se generaran
        tiempo++;
        if (tiempo > tiempoNuevoMalo){
            int nuevosMalos = Math.round(score / 10)+ 1;
            for (int i = 0; i < nuevosMalos; i++)
                crearMalo();
        }

        if (malos > maxMalos)
            gameOver = true;

        //Información de final de partida que aparece en pantalla
        if (gameOver){
            p.setTextSize(canvas.getWidth()/7);
            p.setTextAlign(Paint.Align.CENTER);
            text = "Game Over";
            canvas.drawText(text, canvas.getWidth()/2,
                    canvas.getHeight()/4, p);
            p.setTextSize(canvas.getWidth() /10);
            text = "Toca la pantalla";
            canvas.drawText (text, canvas.getWidth()/2,
                        canvas.getHeight()/4 +(float)(canvas.getHeight()*0.3)/2, p);
            text = "para salir";
            canvas.drawText (text, canvas.getWidth()/2,
                        canvas.getHeight()/4 +(float)(canvas.getHeight()*0.3),p);
            gameLoopThread.setRunning(false);
        }
    }
```

Método privado `crearMalo()`: Creará un malo diferente de forma aleatoria cada vez.

```
    private void crearMalo() {
        Random rnd = new Random();
        switch (rnd.nextInt(4)) {
            case 0:
                sprites.add(createSprite(R.drawable.malo));
                break;
            case 1:
                sprites.add(createSprite(R.drawable.malo2));
                break;
            case 2:
                sprites.add(createSprite(R.drawable.malo3));
                break;
            case 3:
                sprites.add(createSprite(R.drawable.malo4));
                break;
        }
        tiempo = 0;
        malos ++;
    }
```

Modificaremos también el método createSprites, para que inicialmente cree a la PRINCESA y unos 10 sprites MALOS, todos ellos de forma aleatoria.

```
    private void createSprites(){
        //Creamos una princesa y el valor de la cte MALOS_INICIALES sprites malos
        //de manera aleatoria
        sprites.add(createSprite(R.drawable.princesa));
```

```
    for (int i = 0; i < MALOS_INICIALES; i++){
        crearMalo();
    }
}
```

Cuando termina la partida, pasamos el parámetro puntuación a la actividad principal.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (gameOver){
        gameLoopThread.setRunning(false);
        Intent intent =  new Intent(this.getContext(), MainActivity.class);
        intent.putExtra("puntuación",score+"");
        this.getContext().startActivity(intent);
    }

    if (System.currentTimeMillis() - lastClick > 300) {
        lastClick = System.currentTimeMillis();
        float x = event.getX();
        float y = event.getY();
        synchronized (getHolder()) {
            for (int i = sprites.size() - 1; i >= 0; i--) {
                Sprite sprite = sprites.get(i);
                if (sprite.isCollition(event.getX(), event.getY())) {
                    sprites.remove(sprite);
                    temps.add(new TempSprite(temps, this, x, y, bmpBlood));
                    //Para el sonido
                    if (i==0){
                        gameOver=true;
                    }
                    else{
                        soundPool.play(idExplosion, 1,1,0,0,1);
                        malos--;//decrementamos malos cuando lo eliminamos
                    }
                    score++; //incrementamos marcador
                    break;
                }
            }
        }
    }
    return true;
}
```

Resumiendo, la clase **`VistaJuego`** queda de la siguiente manera:

```
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.media.AudioManager;
import android.media.SoundPool;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class VistaJuego extends SurfaceView {

    private SurfaceHolder holder;
    private GameLoopThread gameLoopThread;

    private List<Sprite> sprites = new ArrayList<Sprite>();
    private List<TempSprite> temps = new ArrayList<TempSprite>();
    private long lastClick;
    private Bitmap bmpBlood;

    //Para el sonido: Declaramos las variables
    private SoundPool soundPool;
    private int idExplosion;

    //Para controlar el final de la partida
    private boolean gameOver;

    //Para el marcador
    private int score = 0;
    private int malos;
    private int maxMalos = 40; //Número máximo de malos
    private int tiempo = 0;
    private int tiempoNuevoMalo = 40; //FPS para crear un nuevo malo
    private final static int MALOSINICIALES = 10;


    public VistaJuego(Context context) {
        super(context);

        //Cargamos sonido en memoria
        soundPool = new SoundPool(5,AudioManager.STREAM_MUSIC,0);
        idExplosion = soundPool.load(this.getContext(),R.raw.explosion, 0);

        gameLoopThread = new GameLoopThread(this);
        //holder = getHolder();
        //holder.addCallback(new Callback() {
        getHolder().addCallback(new Callback(){

            @Override
```

```
        public void surfaceCreated(SurfaceHolder holder) {
            createSprites(); //Creamos los sprites
            gameLoopThread.setRunning(true);
            gameLoopThread.start();
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format,
                                   int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            boolean retry = true;
            gameLoopThread.setRunning(false);
            while (retry){
                try{
                    gameLoopThread.join();
                    retry = false;
                }catch (InterruptedException e){
                }
            }
            //gameLoopThread.interrupt();

        }
    });
    bmpBlood = BitmapFactory.decodeResource(getResources(),
                                                R.drawable.blood1);
}

private void createSprites(){
    //Creamos una princesa y MALOSINICIALES sprites malos de manera aleatoria
    sprites.add(createSprite(R.drawable.princesa));
    for (int i = 0; i < MALOSINICIALES; i++){
        crearMalo();
    }
}

private Sprite createSprite(int resouce) {
    Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
    return new Sprite(this, bmp);
}

@Override
protected void onDraw(Canvas canvas) {
    canvas.drawColor(Color.BLACK);

    //Para dibujar más de un sprite
    for (Sprite sprite:sprites) {
        sprite.onDraw(canvas);
    }
    for (int i = temps.size() - 1; i >= 0; i--){
        temps.get(i).onDraw(canvas);
    }

    //Para visualizar los puntos y el número de malos
    Paint p = new Paint();
    p.setColor(Color.RED);
    p.setTextSize(canvas.getWidth() / 10);
    p.setTextAlign(Paint.Align.RIGHT);
    String text = "Score: "+ score;
    canvas.drawText(text, canvas.getWidth(),
                    (float)(canvas.getHeight() * 0.1 ),p);
```

```
        text = "Malos: "+ malos;
        p.setTextAlign(Paint.Align.LEFT);
        canvas.drawText(text, 0, (float)(canvas.getHeight()*0.1), p);

        //Creamos un nuevo malo cada tiempoNuevoMalo FPS y contra mas
        //puntos tengamos más rápido se generaran
        tiempo++;
        if (tiempo > tiempoNuevoMalo){
            int nuevosMalos = Math.round(score / 10)+ 1;
            for (int i = 0; i < nuevosMalos; i++)
                crearMalo();
        }

        if (malos > maxMalos)
            gameOver = true;

        //Información de final de partida que aparece en pantalla
        if (gameOver){
            p.setTextSize(canvas.getWidth()/7);
            p.setTextAlign(Paint.Align.CENTER);
            text = "Game Over";
            canvas.drawText(text, canvas.getWidth()/2,
                    canvas.getHeight()/4, p);
            p.setTextSize(canvas.getWidth() /10);
            text = "Toca la pantalla";
            canvas.drawText (text, canvas.getWidth()/2,
                      canvas.getHeight()/4 +(float)(canvas.getHeight()*0.3)/2, p);
            text = "para salir";
            canvas.drawText (text, canvas.getWidth()/2,
                         canvas.getHeight()/4 +(float)(canvas.getHeight()*0.3),p);
            gameLoopThread.setRunning(false);
        }
    }

    private void crearMalo() {
        Random rnd = new Random();
        switch (rnd.nextInt(4)) {
            case 0:
                sprites.add(createSprite(R.drawable.malo));
                break;
            case 1:
                sprites.add(createSprite(R.drawable.malo2));
                break;
            case 2:
                sprites.add(createSprite(R.drawable.malo3));
                break;
            case 3:
                sprites.add(createSprite(R.drawable.malo4));
                break;
        }
        tiempo = 0;
        malos ++;
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (gameOver){
            gameLoopThread.setRunning(false);
            Intent intent =  new Intent(this.getContext(), MainActivity.class);
            intent.putExtra("puntuación",score+"");
            this.getContext().startActivity(intent);
        }
```

```
        if (System.currentTimeMillis() - lastClick > 300) {
            lastClick = System.currentTimeMillis();
            float x = event.getX();
            float y = event.getY();
            synchronized (getHolder()) {
                for (int i = sprites.size() - 1; i >= 0; i--) {
                    Sprite sprite = sprites.get(i);
                    if (sprite.isCollition(event.getX(), event.getY())) {
                        //if (sprite.isCollition(x ,y )) {
                        sprites.remove(sprite);
                        temps.add(new TempSprite(temps, this, x, y, bmpBlood));
                        //Para el sonido
                        if (i==0){
                            gameOver=true;
                        }
                        else{
                            soundPool.play(idExplosion, 1,1,0,0,1);
                            malos--;//decrementamos malos cuando lo eliminamos
                        }
                        score++; //incrementamos marcador
                        break;
                    }
                }
            }
        }
        return true;
    }
}
```

13.- Creamos la Actividad principal

Vamos a crear una pantalla principal, algo básica, para el juego. En ella tendremos 5 botones. Habría que darle una vuelta al diseño.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingTop="15dp" >
    <TextView
        android:id="@+id/textpuntos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="35dp"
        android:text="@string/nombre"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="35sp"
        android:textColor="#ffffff" />
    <Button
        android:id="@+id/btJugar"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
```

```
        android:layout_marginBottom="10dp"
        android:onClick="jugar"
        android:text="Jugar"
        android:textSize="30sp"
        android:textColor="#1642b2"
        />
    <Button
        android:id="@+id/btAyuda"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:onClick="ayuda"
        android:text="Ayuda"
        android:textSize="20sp" />
    <Button
        android:id="@+id/btPuntuaciones"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:onClick="mostrar"
        android:text="Puntuaciones"
        android:textSize="20sp" />
    <Button
        android:id="@+id/btResetear"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:onClick="resetear"
        android:text="Resetear"
        android:textSize="20sp" />
    <Button
        android:id="@+id/btSalir"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="200dp"
        android:onClick="salir"
        android:text="Salir"
        android:textSize="20sp" />
</LinearLayout>
```

El código de la actividad principal (`MainActivity`), en un principio podría ser;

```
package com.example.amaia.juegoenandroid;

import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private Puntuaciones[] puntos = new Puntuaciones[10];
    private int numero;
    private String nombre;
    private int puntosPartida;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        //setContentView(new VistaJuego(this));
        setContentView(R.layout.activity_main);

    public void jugar(View view){
        setContentView(new VistaJuego(this));
    }

    public void ayuda(View view){
        Toast.makeText(getApplicationContext(),"En este botón veremos la ayuda",
Toast.LENGTH_SHORT).show();
    }

    public void mostrar(View view){
        Toast.makeText(getApplicationContext(),"En este botón se mostrarán las
puntuaciones", Toast.LENGTH_SHORT).show();

    }

    public void salir(View view){
        ActivityCompat.finishAffinity(this);
        System.exit(0);
        finish();
    }

    public void resetear(View view){
        Toast.makeText(getApplicationContext(),"En este botón resetearemos todas
la puntuaciones", Toast.LENGTH_SHORT).show();

    }
}
```

14.- Top Puntuaciones

Vamos a guardar el top de las puntuaciones en un fichero de memoria interna. Para ello vamos a recorrer el fichero y cargarlo en una `ListView`.

`fila_puntuacion.xml`

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/rowTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="left"
    android:padding="10dp"
    android:textColor="#FF0000"
    android:textSize="25sp"
    android:textIsSelectable="true">
</TextView>
```

`mostrar_puntuaciones_layout.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:background="#000000">
```

```
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="PUNTUACIONES"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textColor="#FF0000"
            android:textSize="35sp"
            android:gravity="center_horizontal"/>

        <ListView
            android:id="@+id/mainListView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>
```

Puntuaciones.java

```
    package com.example.amaia.juegoenandroid;

    import java.io.Serializable;

    public class Puntuaciones implements Serializable {

        private static final long serialVersionUID = 345L;
        private String nombre="";
        private int puntuacion;

        public Puntuaciones(){}
        public Puntuaciones(String n,int p){
            this.nombre = n;
            this.puntuacion = p;
        }

        public String getNombre(){return nombre;}
        public int getPuntuacion(){return puntuacion;}
    }
```

Mostrar_puntuaciones_Activity.java

```
package com.example.amaia.juegoenandroid;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.widget.ArrayAdapter;
import android.widget.ListAdapter;
import android.widget.ListView;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;


public class Mostrar_puntuaciones_Activity  extends Activity {
```

```
    private Puntuaciones[] puntos = new Puntuaciones[10];
    private int numero;
    private ListView mainListView;
    private ListAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.mostrar_puntuaciones_layout);

        // recurso ListView.
        leerFichero();
        mainListView = (ListView) findViewById( R.id.mainListView );

        String[] puntuaciones = new String[numero];
        for (int i=0;i<numero;i++){
            puntuaciones[i]=i+1+". "+puntos[i].getNombre()+" "
                                    +puntos[i].getPuntuacion();
        }
        ArrayList<String> puntuacionesLista = new ArrayList<String>();
        puntuacionesLista.addAll( Arrays.asList(puntuaciones) );
        listAdapter = new ArrayAdapter<String>(this, R.layout.fila_puntuacion,
                                                puntuacionesLista);
        mainListView.setAdapter( listAdapter );
    }

    public void leerFichero(){
        String nombre;
        int puntuaje;
        Puntuaciones aux;
        try
        {
            BufferedReader fin =
                    new BufferedReader(
                            new InputStreamReader(
                                    openFileInput("puntuaciones.txt")));

            numero = Integer.parseInt(fin.readLine());
            for (int i=0;i<numero;i++){
                nombre = fin.readLine();
                puntuaje = Integer.parseInt(fin.readLine());
                aux = new Puntuaciones(nombre,puntuaje);
                puntos[i]=aux;
            }
            fin.close();
        }
        catch (Exception ex)
        {
            Log.e("Ficheros", "Error al leer fichero desde memoria interna");
        }
    }

}
```

Cuando arranquemos la aplicación vamos a comprobar si tenemos el fichero donde se guarda la puntuación y si es así cargamos las puntuaciones en un array. La vista juego nos va a devolver una puntuación, comprobamos si esa puntuación es menor que la que hay en el fichero de puntuaciones y si no es así recogemos el nombre y lo añadimos al fichero.

La `vistaJuego` nos va a devolver la puntuación.

```java
package com.example.amaia.juegoenandroid;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    private Puntuaciones[] puntos = new Puntuaciones[10];
    private int numero;
    private String nombre;
    private int puntospartida;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(new VistaJuego(this));
        setContentView(R.layout.activity_main);
        comprobarFichero();
        leerFichero();

        if (this.getIntent().getExtras() != null) {
            Intent intent = getIntent();
            Bundle b = intent.getExtras();

            String score = (String) b.get("puntuacion");
            if (score == null)
                score="0";
            puntospartida=Integer.parseInt(score);
            comprobarScore(puntospartida);
        }

        //setContentView(new Logo(this));
    }

    public void comprobarScore(int s){
        if (s > 0 &&(numero<10 || puntos[9].getPuntuacion()<s)){
            setContentView(R.layout.intro_nombre_puntuacion);
        }
    }

    public void jugar(View view){
        setContentView(new VistaJuego(this));
    }

    public void ayuda(View view){
        Toast.makeText(getApplicationContext(),
                       "En este botón veremos la ayuda",
                        Toast.LENGTH_SHORT).show();
    }
```

```
public void mostrar(View view){
    Toast.makeText(getApplicationContext(),
                   "En este botón se mostrarán las puntuaciones",
                    Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(this, Mostrar_puntuaciones_Activity.class);
    startActivity(intent);
}

public void salir(View view){
    escribirFichero();
    // Eliminamos las actividades si existen
    ActivityCompat.finishAffinity(this);
    System.exit(0);
    finish();
}

public void aceptarNombre(View view){
    EditText name = (EditText)findViewById(R.id.editTextNombre);
    nombre = name.getText().toString();
    if (numero<10)numero++;
    insertaPuntuacion(nombre,puntospartida);
    setContentView(R.layout.activity_main);
}

public void resetear(View view){
    Toast.makeText(getApplicationContext(),
                  "En este botón reseteararemos todas la puntuaciones",
                  Toast.LENGTH_SHORT).show();
}

public void insertaPuntuacion(String n,int s){
    int aux=10;
    Puntuaciones aux2 = new Puntuaciones(n,s);
    for (int i=0;i<numero-1;i++){
        if (puntos[i].getPuntuacion()<s){
            aux = i;
            break;
        }
    }
    if (aux == 10)aux=numero-1;
    for (int j=numero-1;j>aux;j--){
        puntos[j]=puntos[j-1];
    }
    puntos[aux]=aux2;
    escribirFichero();
}

public void leerFichero(){
    String nombre;
    int puntuaje;
    Puntuaciones aux;
    try
    {
        BufferedReader fin =
                new BufferedReader(
                        new InputStreamReader(
                                openFileInput("puntuaciones.txt")));

        numero = Integer.parseInt(fin.readLine());
        for (int i=0;i<numero;i++){
            nombre = fin.readLine();
            puntuaje = Integer.parseInt(fin.readLine());
            aux = new Puntuaciones(nombre,puntuaje);
```

```
                puntos[i]=aux;
            }
            fin.close();
        }
        catch (Exception ex)
        {
            Log.e("Ficheros", "Error al leer fichero desde memoria interna");
        }
    }

    public void escribirFichero() {
        try {
            OutputStreamWriter fout =
                    new OutputStreamWriter(
                        openFileOutput("puntuaciones.txt", Context.MODE_PRIVATE));

            fout.write(numero + "\n");
            for (int i = 0; i < numero; i++) {
                fout.write(puntos[i].getNombre() + "\n");
                fout.write(puntos[i].getPuntuacion() + "\n");
            }
            fout.close();
        } catch (Exception ex) {
            Log.e("Ficheros", "Error al escribir fichero a memoria interna");
        }
    }

    public void comprobarFichero(){
        try
        {
            BufferedReader fin =
                    new BufferedReader(
                            new InputStreamReader(
                                    openFileInput("puntuaciones.txt")));
            fin.close();
        }
        catch (Exception ex)
        {
            Log.e("Ficheros", "Error al leer fichero desde memoria interna");
            crearFichero();
        }
    }

   public void crearFichero(){
        try
        {
            OutputStreamWriter fout=
                    new OutputStreamWriter(
                        openFileOutput("puntuaciones.txt", Context.MODE_PRIVATE));

            fout.write("0");
            fout.close();
        }
        catch (Exception ex)
        {
            Log.e("Ficheros", "Error al escribir fichero a memoria interna");
        }
    }
}
```