Juego Plataformas

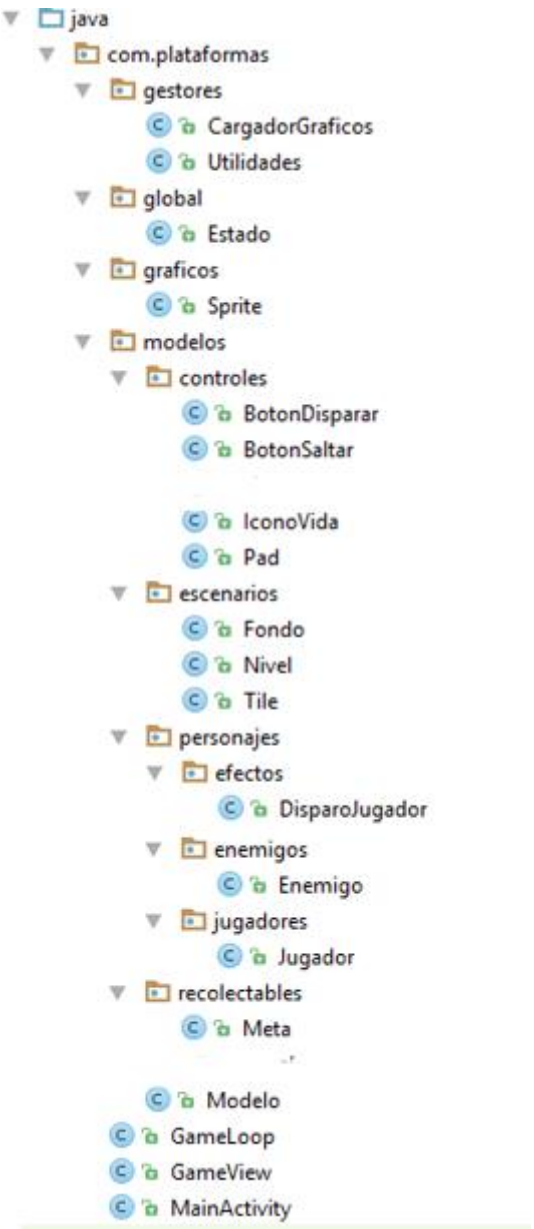
AMPLIACIONES

Adrián Jiménez Villarreal
71904816P | UO227602@UNIOVI.ES

Contenido

Ampliaciones

Antes de realizar cualquiera de las ampliaciones se ha reorganizado la estructura de paquetes, añadido nuevos y creado una clase con los estados que puede estar los objetos.



Nuevos tipos de enemigos

Descripción

Lo más adecuado sería crear una clase base para enemigo o un interface Java y que los diferentes tipos de enemigos heredasen de la clase base o implementaran el interface.

Al menos se debe crear un tipo de enemigo nuevo con diferente imagen, velocidad y capaz de disparar

Creación de la clase base Enemigo y sus derivados

Para empezar, crearemos una clase base para los enemigos, en este caso usaremos la clase Enemigo como base que sea ella quien extienda de modelo

```
public class Enemigo extends Modelo {
    public static final String CAMINANDO_DERECHA =
"Caminando_derecha";
    public static final String CAMINANDO_IZQUIERDA =
"caminando_izquierda";

    public static final String MUERTE_DERECHA = "muerte_derecha";
    public static final String MUERTE_IZQUIERDA =
"muerte_izquierda";


    protected Sprite sprite;
    protected HashMap<String,Sprite> sprites = new
HashMap<String,Sprite>();

    public double velocidadX = 1.2;
    protected int orientacion;
    public int estado;

    public static final int DERECHA = 1;
    public static final int IZQUIERDA = -1;


    public Enemigo(Context context, double xInicial, double
yInicial) {
        super(context, 0, 0, 40, 40);

        this.x = xInicial;
        this.y = yInicial - altura/2;

    }

    public void inicializar (){

        Sprite caminandoDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyrunright),
                ancho, altura,
                4, 4, true);
        sprites.put(CAMINANDO_DERECHA, caminandoDerecha);

        Sprite caminandoIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyrun),
                ancho, altura,
                4, 4, true);
```

```
        sprites.put(CAMINANDO_IZQUIERDA, caminandoIzquierda);


        Sprite muerteDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydieright),
                ancho, altura,
                4, 8, false);
        sprites.put(MUERTE_DERECHA, muerteDerecha);

        Sprite muerteIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydie),
                ancho, altura,
                4, 8, false);
        sprites.put(MUERTE_IZQUIERDA, muerteIzquierda);



        sprite = caminandoDerecha;
    }


    public void actualizar (long tiempo) {
        boolean finSprite = sprite.actualizar(tiempo);

        if ( estado == Estado.INACTIVO && finSprite){
            estado = Estado.ELIMINAR;
        }
        if (estado == Estado.INACTIVO){
            if (velocidadX > 0) {
                sprite = sprites.get(MUERTE_DERECHA);
                orientacion = DERECHA;
            }
            else{
                sprite = sprites.get(MUERTE_IZQUIERDA);
                orientacion = IZQUIERDA;
            }
        } else {

            if (velocidadX > 0) {
                sprite = sprites.get(CAMINANDO_DERECHA);
                orientacion = DERECHA;
            }
            if (velocidadX < 0) {
                sprite = sprites.get(CAMINANDO_IZQUIERDA);
                orientacion = IZQUIERDA;
            }

        }
    }

    public void girar(){
        velocidadX = velocidadX*-1;
    }

    public void dibujar(Canvas canvas){
        sprite.dibujarSprite(canvas, (int) x - Nivel.scrollEjeX,
(int) y - Nivel.scrollEjeY);
    }
    public void destruir (){
```

```
        velocidadX = 0;
        estado = Estado.INACTIVO;
    }

}
```

Ahora pasaremos a crear las clases hijas, cada una con sus diferencias.

Clase **EnemigoBásico**.

```
public class EnemigoBasico extends Enemigo {
    public EnemigoBasico(Context context, double xInicial, double
yInicial) {
        super(context, xInicial, yInicial);
        cDerecha = 15;
        cIzquierda = 15;
        cArriba = 20;
        cAbajo = 20;

        estado = Estado.ACTIVO;

        inicializar();
    }
}
```

Creación de los nuevos tipos de disparo

Como ahora los enemigos de la ampliación necesitarán disparar crearemos una clase base **Disparo** con lo común y sus hijas serán **DisparoEnemigo** y **DisparoJugador.**

Clase **Disparo.**

```
public class Disparo extends Modelo {

    protected Sprite sprite;
    public double velocidadX = 10;

    public Disparo(Context context, double xInicial, double
yInicial) {
        super(context, xInicial, yInicial, 35, 35);

    }

    public void inicializar (){
        sprite= new Sprite(
                CargadorGraficos.cargarDrawable(context,
                        R.drawable.animacion_disparo3),
                ancho, altura,
                24, 5, true);
    }
    public void actualizar (long tiempo) {
        sprite.actualizar(tiempo);
    }

    public void dibujar(Canvas canvas){
        sprite.dibujarSprite(canvas, (int) x - Nivel.scrollEjeX,
(int) y - Nivel.scrollEjeY);
```

```
    }
}
```

Clase **DisparoEnemigo**.

```
public class DisparoEnemigo extends Disparo{
    public DisparoEnemigo(Context context, double xInicial, double
yInicial, int orientacion) {
        super(context, xInicial, yInicial);

        if (orientacion == Enemigo.IZQUIERDA)
            velocidadX = velocidadX*-1;

        cDerecha = 6;
        cIzquierda = 6;
        cArriba = 6;
        cAbajo = 6;

        inicializar();
    }
}
```

Clase **DisparoJugador**.

```
public class DisparoJugador extends Disparo {

    public DisparoJugador(Context context, double xInicial, double
yInicial,int orientacion) {
        super(context, xInicial, yInicial);

        if (orientacion == Jugador.IZQUIERDA)
            velocidadX = velocidadX*-1;

        cDerecha = 6;
        cIzquierda = 6;
        cArriba = 6;
        cAbajo = 6;

        inicializar();
    }

    public void inicializar (){
        sprite= new Sprite(
                CargadorGraficos.cargarDrawable(context,
                        R.drawable.animacion_disparo3),
                ancho, altura,
                24, 5, true);
    }

    public void actualizar (long tiempo) {
        sprite.actualizar(tiempo);
    }

    public void dibujar(Canvas canvas){
        sprite.dibujarSprite(canvas, (int) x - Nivel.scrollEjeX,
(int) y - Nivel.scrollEjeY);
    }
}
```

Con todo esto creado podremos pasar a crear la clase **EnemigoAmpliacion.**

Creación del nuevo tipo de enemigo

Clase **EnemigoAmpliacion**.

```
public class EnemigoAmpliacion extends Enemigo {

    protected int cadenciaDisparo = 3000;
    protected long milisegundosDisparo = 0;

    public EnemigoAmpliacion(Context context, double xInicial,
double yInicial) {
        super(context, xInicial, yInicial);

        velocidadX = 1.8;
        cDerecha = 15;
        cIzquierda = 15;
        cArriba = 20;
        cAbajo = 20;

        estado = Estado.ACTIVO;

        inicializar();
    }

    @Override
    public void inicializar() {
        Sprite caminandoDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyrunrightampliacion),
                ancho, altura,
                4, 4, true);
        sprites.put(CAMINANDO_DERECHA, caminandoDerecha);

        Sprite caminandoIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyrunampliacion),
                ancho, altura,
                4, 4, true);
        sprites.put(CAMINANDO_IZQUIERDA, caminandoIzquierda);


        Sprite muerteDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydierightampliacion),
                ancho, altura,
                3, 6, false);
        sprites.put(MUERTE_DERECHA, muerteDerecha);

        Sprite muerteIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydieampliacion),
                ancho, altura,
                3, 6, false);
        sprites.put(MUERTE_IZQUIERDA, muerteIzquierda);



        sprite = caminandoDerecha;
    }
```

```
    public Disparo disparar(long milisegundos) {
        if (milisegundos - milisegundosDisparo> cadenciaDisparo
                + Math.random()* cadenciaDisparo) {
            milisegundosDisparo = milisegundos;
            return new DisparoEnemigo(context, x, y,orientacion);
        }
        return null;
    }
}
```

Ya tenemos todo lo necesario para incluirlo en el juego ahora solo habrá que modificar la clase **Nivel.**

Inclusión en el Nivel

En esta clase habrá que modificar el método aplicarReglasDeMovimiento, ya que este es el que se encargara de gestionar tanto los nuevos enemigos, como los disparos que estos realizaran. Además debemos gestionar como incluir los enemigos en el mapa.

1º Se crea una lista con los disparos de los enemigos.

2º Se inicializa en el método inicializar.

3* Se gestiona la inclusión de nuevos tipos de enemigo.

```
case 'N':
    // Enemigo
    // Posición centro abajo
    int xCentroAbajoTileN = x * Tile.ancho + Tile.ancho / 2;
    int yCentroAbajoTileN = y * Tile.altura + Tile.altura;
    enemigos.add(new EnemigoAmpliacion(context, xCentroAbajoTileN,
yCentroAbajoTileN));

    return new Tile(null, Tile.PASABLE);
```

4º Se incluye en el método aplicarReglasDeMovimiento la creación de los disparos de los enemigos dentro del iterador de enemigos, justo cuando estos están a un rango del jugador.

```
if (tileXJugadorIzquierda - rango < tileXEnemigoIzquierda &&
        tileXJugadorIzquierda + rango > tileXEnemigoIzquierda){

…

long tiempo = System.currentTimeMillis();
if (enemigo.estado == Estado.ACTIVO){
        if(enemigo instanceof EnemigoAmpliacion) {
            DisparoEnemigo disparo = (DisparoEnemigo)
((EnemigoAmpliacion)enemigo).disparar(tiempo);
            if (disparo != null) {
                disparosEnemigos.add(disparo);
            }
        }
    }
```

5º Se añade la colisión con este tipo de disparos para el jugador.

```
for (Iterator<Disparo> iterator = disparosEnemigos.iterator();
iterator.hasNext();) {
    DisparoEnemigo disparoEnemigo = (DisparoEnemigo)
iterator.next();

Mismo código para recorrer los disparos del jugador, solo cambia el
if final.
…

    if (disparoEnemigo.colisiona(jugador)){
        if(jugador.golpeado() <= 0){
            nivelPausado = true;
            mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.you_lose);
            jugador.restablecerPosicionInicial();
            inicializar();
            return;
        }
     }
}
```

Con esto los nuevos enemigos ya dispararían si están en un rango cercano al jugador, tendrían un sprite diferente y se moverían más rápido. Con lo que la ampliación estaría completa.

Enemigo que salta o vuela

Descripción

Incluir un nuevo tipo de enemigo capaz de saltar o volar (no puede volar siempre en línea recta, ha de moverse en el eje X e Y).

Creación del nuevo tipo de enemigo

Seguiremos la misma manera de creación de un nuevo tipo de enemigo vista en Nuevos tipos de enemigos. Pero en este caso contendrá unos ligeros la clase **EnemigoVolador**.

```
public class EnemigoVolador extends Enemigo {

    public double velocidadY = 1.0;
    public double yInicial;
    public double yDestino;
    public boolean direccion;

    public EnemigoVolador(Context context, double xInicial, double
yInicial) {
        super(context, xInicial, yInicial);

        this.yInicial = y;
        velocidadX = 1.8;
        cDerecha = 15;
        cIzquierda = 15;
        cArriba = 20;
        cAbajo = 20;

        estado = Estado.ACTIVO;
```

```
        inicializar();
    }

    @Override
    public void inicializar() {
        Sprite caminandoDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyrightvolador),
                ancho, altura,
                3, 6, true);
        sprites.put(CAMINANDO_DERECHA, caminandoDerecha);

        Sprite caminandoIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemyvolador),
                ancho, altura,
                3, 6, true);
        sprites.put(CAMINANDO_IZQUIERDA, caminandoIzquierda);


        Sprite muerteDerecha = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydievolador),
                ancho, altura,
                3, 6, false);
        sprites.put(MUERTE_DERECHA, muerteDerecha);

        Sprite muerteIzquierda = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.enemydievolador),
                ancho, altura,
                3, 6, false);
        sprites.put(MUERTE_IZQUIERDA, muerteIzquierda);



        sprite = caminandoDerecha;
    }
    public void volar() {
        if (direccion && y >= yDestino) {
            y -= velocidadY;
            if (y <= yDestino) {
                direccion = false;
                yDestino = y + calcularMovimiento();
                if (yDestino > yInicial + 33)
                    yDestino = yInicial + 33;
            }
        }
        if (!direccion && y <= yDestino) {
            y += velocidadY;
            if (y >= yDestino) {
                direccion = true;
                yDestino = y - calcularMovimiento();
                if (yDestino < yInicial - 33)
                    yDestino = yInicial - 33;
            }
        }
        if (yDestino == 0 && y == yInicial) {
            int movimiento = calcularMovimiento();
```

```
            int posicion = (int) Math.floor(Math.random() * (2 - 0 +
1) + (0));
            if (posicion == 0) {
                yDestino = y - movimiento;
                direccion = true;
            } else {
                yDestino = movimiento + y;
                direccion = false;
            }
        }
    }

    private int calcularMovimiento(){
        return (int)Math.floor(Math.random()*(25-1+1)+(1));
    }
```

Inclusión del tipo de enemigo en el nivel

Al igual que con los anteriores enemigos que hemos añadido se debe añadir los nuevos tipos de enemigo.

```
case 'V':
    // Enemigo
    // Posición centro abajo
    int xCentroAbajoTileV = x * Tile.ancho + Tile.ancho / 2;
    int yCentroAbajoTileV = y * Tile.altura + Tile.altura;
    enemigos.add(new EnemigoVolador(context, xCentroAbajoTileV,
yCentroAbajoTileV));

    return new Tile(null, Tile.PASABLE);
```

Posteriormente deberemos añadir las nuevas reglas de movimiento para este tipo de enemigo.

Cuando la velocidad de X del enemigo sea > 0

```
…
}else if(tileXEnemigoDerecha + 1 <= anchoMapaTiles() - 1 &&
        mapaTiles[tileXEnemigoDerecha +
1][tileYEnemigoInferior].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoDerecha +
1][tileYEnemigoCentro].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoDerecha +
1][tileYEnemigoSuperior].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoDerecha + 1][tileYEnemigoInferior +
1].tipoDeColision ==
                Tile.PASABLE && enemigo instanceof EnemigoVolador){

    enemigo.x += enemigo.velocidadX;
    ((EnemigoVolador) enemigo).volar();

    // Sino, me acerco al borde del que estoy
}
…
```

Cuando la velocidad de X del enemigo sea < 0

```
…
}else if(tileXEnemigoIzquierda - 1 >= 0 &&
        mapaTiles[tileXEnemigoIzquierda -
1][tileYEnemigoInferior].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoIzquierda -
1][tileYEnemigoCentro].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoIzquierda -
1][tileYEnemigoSuperior].tipoDeColision ==
                Tile.PASABLE &&
        mapaTiles[tileXEnemigoIzquierda - 1][tileYEnemigoInferior +
1].tipoDeColision
                == Tile.PASABLE && enemigo instanceof
EnemigoVolador){

    enemigo.x += enemigo.velocidadX;
    ((EnemigoVolador) enemigo).volar();
}
….
```

Con esto ya tendríamos al nuevo enemigo volador.

Scroll en el eje Y

Descripción

Permitir que los mapas tengan más altura que la propia pantalla, se debe aplicar un scroll en el eje Y similar al que se aplicó en el eje X.

Modificación de la clase Nivel

Esta ampliación será similar al scroll del eje X.

1º Se crea la nueva variable scrollEjeY inicializada a 0

2º Después de iniciar el mapa de tiles se le da el valor inicial al scroll del eje Y, pero antes de esto debe valer 0.

```
scrollEjeY = 0;
mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.description);
……
inicializarMapaTiles();
scrollEjeY = (int) (altoMapaTiles() -
tilesEnDistanciaY(GameView.pantallaAlto))*Tile.altura;
```

3º El método dibujar tiles deberá tener las condiciones del scroll del eje y.

```
…
int tileYJugador = (int) jugador.y / Tile.altura;

int arriba = (int) (tileYJugador - tilesEnDistanciaY(jugador.y -
scrollEjeY));
arriba = Math.max(0,arriba);

if ( jugador .y  <
        altoMapaTiles()* Tile.altura - GameView.pantallaAlto*0.3 )
```

```
    if( jugador .y- scrollEjeY > GameView.pantallaAlto * 0.7 )
        scrollEjeY += (int) ((jugador .y - scrollEjeY) -
GameView.pantallaAlto* 0.7);


if ( jugador .y  > GameView.pantallaAlto*0.3 )
    if( jugador .y - scrollEjeY < GameView.pantallaAlto *0.3 )
        scrollEjeY -= (int) (GameView.pantallaAlto*0.3 -(jugador .y
- scrollEjeY));


int abajo = arriba +
        GameView.pantallaAlto / Tile.altura + 1;

abajo = Math.min(abajo, altoMapaTiles() - 1);


for (int y = arriba; y <= abajo; ++y) {
    for (int x = izquierda; x <= derecha; ++x) {
        if (mapaTiles[x][y].imagen != null) {
            // Calcular la posición en pantalla correspondiente
            // izquierda, arriba, derecha , abajo

            mapaTiles[x][y].imagen.setBounds(
                    (x * Tile.ancho) - scrollEjeX,
                    (y * Tile.altura) - scrollEjeY,
                    (x * Tile.ancho) + Tile.ancho - scrollEjeX,
                    (y * Tile.altura + Tile.altura) - scrollEjeY);

            mapaTiles[x][y].imagen.draw(canvas);
        }
    }
}
```

Con todo esto ya estaría acabado de añadir el scroll del eje Y en el nivel, ahora habría que añadirlo a cada clase que este dibujada en el nivel, como el jugador, los enemigos, etc… como se hizo con el scroll en el eje X.

Items recolectables

Descripción

Incluir ítems recolectable inmovibles que puedan ser recolectados por el Jugador al colisionar con ellos. Debemos incluir un contador de ítems que muestre el número en pantalla. El Ítem debe ser animado, utilizando el siguiente sprite:

El número y posición de los ítems se deberán poder especificar en el mapa del Nivel.

Creación de la clase recolectable

Dentro del paquete recolectables crear la clase **Recolectable.**

```
package com.plataformas.modelos.recolectables;

import android.content.Context;
import android.graphics.Canvas;

import com.plataformas.R;
import com.plataformas.gestores.CargadorGraficos;
import com.plataformas.global.Estado;
import com.plataformas.graficos.Sprite;
import com.plataformas.modelos.Modelo;
import com.plataformas.modelos.escenarios.Nivel;

import java.util.HashMap;

/**
 * Created by uo227602 on 05/10/2016.
 */
public class Recolectable extends Modelo {

    public static final String GEMA_GIRANDO = "Gema_Girando";
    public static final String GEMA_DESAPARECIENDO =
"Gema_desapareciendo";

    //Puntero sprite actual
    private Sprite sprite;

    private HashMap<String,Sprite> sprites = new
HashMap<String,Sprite> ();

    double xInicial;
    double yInicial;

    int estado;


    public Recolectable(Context context, double xInicial, double
yInicial) {
        super(context, xInicial,yInicial, 40, 40);

        this.xInicial=xInicial;
        this.yInicial=yInicial;

        inicializar();
    }

    private void inicializar() {

        Sprite gemaGirando = new Sprite(
                CargadorGraficos.cargarDrawable(context,
R.drawable.gem),
                ancho, altura,
                4, 8, true);
        sprites.put(GEMA_GIRANDO, gemaGirando);

        Sprite gemaDesapareciendo = new Sprite(
                CargadorGraficos.cargarDrawable(context,
```

```
R.drawable.item_on_collected),
                ancho, altura,
                5, 10, false);
        sprites.put(GEMA_DESAPARECIENDO, gemaDesapareciendo);
        estado = Estado.ACTIVO;
        sprite = gemaGirando;
    }

    public void dibujar(Canvas canvas){
        sprite.dibujarSprite(canvas, (int)x - Nivel.scrollEjeX ,
(int)y );
    }

    public void actualizar(long tiempo){
        boolean finSprite = sprite.actualizar(tiempo);

        if (estado == Estado.INACTIVO) {
            sprite = sprites.get(GEMA_DESAPARECIENDO);
        }
        else{
            sprite = sprites.get(GEMA_GIRANDO);
        }
        if ( estado == Estado.INACTIVO && finSprite){
            estado = Estado.ELIMINAR;
        }
    }
    public void destruir(){
        estado = Estado.INACTIVO;
    }
    public int getEstado() {
        return estado;
    }
}
```

Creación de la clase contador

Dentro del paquete controles crear la clase **Contador.**

```
package com.plataformas.modelos.controles;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

import com.plataformas.GameView;
import com.plataformas.R;
import com.plataformas.gestores.CargadorGraficos;
import com.plataformas.modelos.Modelo;

/**
 * Created by uo227602 on 05/10/2016.
 */
public class Contador extends Modelo {

    int puntuacion=0;

    public Contador(Context context) {
        super(context, GameView.pantallaAncho*0.90 ,
```

```
GameView.pantallaAlto*0.1,
                GameView.pantallaAlto, GameView.pantallaAncho);

        altura = 40;
        ancho = 40;
        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.score);
    }

    public void actualizarPuntuacion(int puntuacion){
        this.puntuacion+=puntuacion;
    }
    public void reiniciarPuntuacion(){
        this.puntuacion = 0;
     }


    @Override
    public void dibujar(Canvas canvas){
        int yArriva = (int)  y - altura / 2;
        int xIzquierda = (int) x - ancho / 2;
        Paint paint = new Paint();
        paint.setColor(Color.WHITE);
        paint.setAntiAlias(true);
        paint.setTextSize(20);
        canvas.drawText(String.valueOf(puntuacion), xIzquierda -
ancho, yArriva + altura/2, paint);
        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }

}
```

Modificación del nivel

En el nivel habrá que añadir la lista de recolectables, cargar los recolectables en esta lista y gestionar las colisiones con estos en el método aplicar reglas de movimiento.

```
private List<Recolectable> recolectables = new ArrayList<>();
```

```
case 'a':
    int xCentroAbajoTileR = x * Tile.ancho + Tile.ancho/2;
    int yCentroAbajoTileR = y * Tile.altura + Tile.altura;
    recolectables.add(new Recolectable(context, xCentroAbajoTileR,
yCentroAbajoTileR));

    return new Tile(null, Tile.PASABLE);
```

```
for(Iterator<Recolectable> iterator =
recolectables.iterator();iterator.hasNext();){
    Recolectable recolectable = iterator.next();
    if(jugador.colisiona(recolectable) &&
recolectable.getEstado()==Estado.ACTIVO){
        recolectable.destruir();
```

```
            gameView.contador.actualizarPuntuacion(1);
            break;
        }
        if (recolectable.getEstado() == Estado.ELIMINAR){
            iterator.remove();
            continue;
        }
    }
```

Plataformas móviles

Descripción

Incluir plataformas móviles con movilidad en el eje X.

Deben moverse con una velocidad constante preestablecida hasta que colisionen con algún elemento. El jugador y los enemigos deben poder caminar sobre estas plataformas.

Las plataformas se representarán en el mapa de nivel con el carácter 'X'.

Creación de la clase Plataforma

Dentro del paquete escenarios.

```
public class Plataforma extends Modelo {

    double velocidadX = 3;

    public Plataforma(Context context, double x, double y) {
        super(context, x, y, 32, 40);

        this.y = y -altura/2;


        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.platform);
    }

    public boolean girar(){
        velocidadX = velocidadX*-1;
        return true;
    }

    public void dibujar(Canvas canvas){
        int yArriva = (int)  y - altura/2  - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho/2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }
}
```

Modificación de la clase Nivel

Primero se incluirá una lista de plataformas, esta lista se inicializará con lo leído de los ficheros del nivel, además se deberá dibujar cada elemento en el nivel.

Posterior a esto se debe crear una variable global booleana denominada giradoPlataforma. La usaremos para saber cuándo gira la plataforma.

Posteriormente a esto se modifica el método aplicarReglasMovimiento.

```
……
for (Iterator<Plataforma> iterator = plataformas.iterator();
iterator.hasNext(); ) {
    Plataforma plataforma = iterator.next();

    int tileXPlataformaIzquierda =
            (int) (plataforma.x - (plataforma.ancho / 2 - 1)) /
Tile.ancho;
    int tileXPlataformaDerecha =
            (int) (plataforma.x + (plataforma.ancho / 2 - 1)) /
Tile.ancho;
    int tileYPlataformaInferior =
            (int) (plataforma.y + (plataforma.altura / 2 - 1)) /
Tile.altura;
    int tileYPlataformaCentro =
            (int) plataforma.y / Tile.altura;
    if (tileXPlataformaDerecha == tileXPlataformaIzquierda &&
giradoPlataforma) {
        giradoPlataforma = false;
        plataforma.x += plataforma.velocidadX;
    } else {
        if (plataforma.velocidadX > 0) {
            if (tileXPlataformaDerecha + 1 <= anchoMapaTiles() - 1
&&
                    mapaTiles[tileXPlataformaDerecha +
1][tileYPlataformaInferior].tipoDeColision ==
                            Tile.PASABLE &&
                    mapaTiles[tileXPlataformaDerecha +
1][tileYPlataformaCentro].tipoDeColision ==
                            Tile.PASABLE){

                plataforma.x += plataforma.velocidadX;

mapaTiles[tileXPlataformaDerecha][tileYPlataformaCentro] = new
Tile(null, Tile.SOLIDO);
                if (!giradoPlataforma)
                    mapaTiles[tileXPlataformaDerecha -
1][tileYPlataformaCentro] = new Tile(null, Tile.PASABLE);
            } else if (tileXPlataformaDerecha + 1 <=
anchoMapaTiles() - 1) {
                int TileEnemigoDerecho = tileXPlataformaDerecha *
Tile.ancho + Tile.ancho;
                double distanciaX = TileEnemigoDerecho -
(plataforma.x + plataforma.ancho / 2);

                if (distanciaX > 0) {
                    double velocidadNecesaria = Math.min(distanciaX,
plataforma.velocidadX);
                    plataforma.x += velocidadNecesaria;
                } else {
                    mapaTiles[tileXPlataformaDerecha -
1][tileYPlataformaCentro] = new Tile(null, Tile.PASABLE);
                    giradoPlataforma = plataforma.girar();
                }
            } else {
```

```
                plataforma.girar();
            }
        }
    }
    if (tileXPlataformaDerecha == tileXPlataformaIzquierda &&
giradoPlataforma) {
        giradoPlataforma = false;
        plataforma.x += plataforma.velocidadX;
    } else {
        if (plataforma.velocidadX < 0) {
            if (tileXPlataformaIzquierda - 1 >= 0 &&
                    mapaTiles[tileXPlataformaIzquierda -
1][tileYPlataformaInferior].tipoDeColision ==
                            Tile.PASABLE &&
                    mapaTiles[tileXPlataformaIzquierda -
1][tileYPlataformaCentro].tipoDeColision ==
                            Tile.PASABLE){

                plataforma.x += plataforma.velocidadX;

mapaTiles[tileXPlataformaIzquierda][tileYPlataformaCentro] = new
Tile(null, Tile.SOLIDO);
                if (!giradoPlataforma)
                    mapaTiles[tileXPlataformaIzquierda +
1][tileYPlataformaCentro] = new Tile(null, Tile.PASABLE);
            } else if (tileXPlataformaIzquierda - 1 >= 0) {
                giradoPlataforma = false;
                int TileEnemigoIzquierdo = tileXPlataformaIzquierda
* Tile.ancho;
                double distanciaX = (plataforma.x - plataforma.ancho
/ 2) - TileEnemigoIzquierdo;
                if (distanciaX > 0) {
                    double velocidadNecesaria =
                            Utilidades.proximoACero(-distanciaX,
plataforma.velocidadX);
                    plataforma.x += velocidadNecesaria;
                } else {
                    mapaTiles[tileXPlataformaIzquierda +
1][tileYPlataformaCentro] = new Tile(null, Tile.PASABLE);
                    giradoPlataforma = plataforma.girar();
                }
            } else {
                plataforma.girar();
            }
        }
    }
    if(jugador.colisiona(plataforma)){
        jugador.x+=plataforma.velocidadX;
    }
    for(Enemigo enemigo:enemigos){
        if(enemigo.colisiona(plataforma)){
            enemigo.x+=plataforma.velocidadX;
        }
    }
}
……
```

Caja que se puedan arrastrar

Descripción

Incluir cajas que puedan ser arrastradas por el nivel cuando el jugador colisiona con ellos en el eje X.

Este tipo de elemento se representará en el mapa de nivel con la letra Q.

Creación de la caja

Esta clase será creada dentro del paquete escenarios

```
public class Caja extends Modelo {

    double velocidadX = 3;

    public Caja(Context context, double x, double y) {
        super(context, x, y, 32, 40);

        this.y = y -altura/2;


        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.platform_dos);
    }


    public void dibujar(Canvas canvas){
        int yArriva = (int)  y - altura/2  - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho/2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }
}
```

Modificación del nivel

Primero se creará una serie de cajas, se inicializarán, se asignarán al mapa de tiles y se dibujarán. Además de una variable booleana para saber cuándo esta encima de las cajas.

Posteriormente se modificará el método aplicarReglasMovimiento, estas modificaciones controlaran cuando el jugador arrastra la caja desde la izquierda o derecha, si esta encima de ella para que sea un espacio solido para este, y por ultimo para que los enemigo choquen contra esta.

```
…
for (Iterator<Caja> iterator = cajas.iterator(); iterator.hasNext();
) {

    Caja caja = iterator.next();

    int tileXCajaIzquierda =
            (int) (caja.x - (caja.ancho / 2 - 1)) / Tile.ancho;
    int tileXCajaDerecha =
            (int) (caja.x + (caja.ancho / 2 - 1)) / Tile.ancho;
    int tileYCajaInferior =
```

```
            (int) (caja.y + (caja.altura / 2 - 1)) / Tile.altura;
    int tileYCajaCentro =
            (int) caja.y / Tile.altura;
    int tileYCajaSuperior = (int) (caja.y - (caja.altura / 2 - 1)) /
Tile.altura;
    int tileJugador = (int) jugador.x / Tile.ancho;
    if (jugador.getVelocidadX() > 0) {
        if (jugador.colisiona(caja) && tileJugador ==
tileXCajaIzquierda &&
mapaTiles[tileXCajaDerecha][tileYCajaCentro].tipoDeColision ==
Tile.PASABLE
                && mapaTiles[tileXCajaDerecha][tileYCajaInferior +
1].tipoDeColision == Tile.SOLIDO && tileXCajaDerecha + 1 <=
anchoMapaTiles() - 1) {
            caja.x += jugador.getVelocidadX();
            encimaCaja = false;
        } else if (jugador.colisiona(caja) && tileYJugadorInferior
== tileYCajaCentro && !encimaCaja && jugador.enElAire) {
            jugador.y = (caja.y - (caja.altura));
            jugador.enElAire = false;
            encimaCaja = true;
        } else if (jugador.colisiona(caja) && encimaCaja) {
            jugador.enElAire = false;
            encimaCaja = true;
        } else if (!jugador.colisiona(caja) && (tileJugador >=
tileXCajaDerecha || tileYJugadorCentro < tileYCajaSuperior+1)) {
            jugador.enElAire = true;
            encimaCaja = false;
        }
    } else if (jugador.getVelocidadX() < 0) {
        if (jugador.colisiona(caja) && tileJugador ==
tileXCajaDerecha &&
mapaTiles[tileXCajaIzquierda][tileYCajaCentro].tipoDeColision ==
Tile.PASABLE
                && mapaTiles[tileXCajaIzquierda][tileYCajaInferior +
1].tipoDeColision == Tile.SOLIDO && tileXCajaIzquierda - 1 >= 0) {
            caja.x += jugador.getVelocidadX();
            encimaCaja = false;
        } else if (jugador.colisiona(caja) && tileYJugadorInferior
== tileYCajaCentro && !encimaCaja && jugador.enElAire) {
            jugador.y = (caja.y - (caja.altura));
            jugador.enElAire = false;
            encimaCaja = true;
        } else if (jugador.colisiona(caja) && encimaCaja) {
            jugador.enElAire = false;
            encimaCaja = true;
        } else if (!jugador.colisiona(caja) && (tileJugador <=
tileXCajaIzquierda || tileYJugadorCentro < tileYCajaSuperior+1)) {
            jugador.enElAire = true;
            encimaCaja = false;
        }
    } else {
        if (jugador.colisiona(caja) && tileYJugadorInferior ==
tileYCajaCentro && !encimaCaja && jugador.enElAire) {
            jugador.y = (caja.y - (caja.altura));
            jugador.enElAire = false;
            encimaCaja = true;
        } else if (jugador.colisiona(caja) && encimaCaja) {
            jugador.enElAire = false;
            encimaCaja = true;
```

```
        } else if (tileJugador==tileYCajaCentro &&
!jugador.colisiona(caja) && tileYJugadorCentro <
tileYCajaSuperior+1) {
            jugador.enElAire = true;
            encimaCaja = false;
        }
    }
    for (Enemigo enemigo : enemigos) {
        if (enemigo.colisiona(caja)) {
            enemigo.girar();
        }
    }
    for (Plataforma plataforma : plataformas) {
        if (plataforma.colisiona(caja)) {
            plataforma.girar();
        }
    }
}
…

if (jugador.getVelocidadY() >= 0 && !encimaCaja
) {
…
```

Tiles destruibles

Descripción

Incluir tiles que desaparezcan a los pocos segundos de que el Jugador entre en contacto con ellos (este colocado encima).

Este tipo de tiles se representará en el mapa de nivel con la letra W.

Creacion de Destructible

La creación de esta ira dentro del paquete escenarios

```
public class Destructible extends Modelo {

    long tiempoPiso;

    public Destructible(Context context, double x, double y) {
        super(context, x, y, 32, 40);

        this.y = y -altura/2;


        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.platform_tres);
    }

    public void dibujar(Canvas canvas){
        int yArriva = (int)  y - altura/2  - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho/2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }
```

```
}
```

Modificacion del Nivel

Se añadirá una lista de destructibles, se inicializarán, se añadirán para que se puedan reconocer al cargar el mapa de tiles y por ultimo se dibujaran.

Posteriormente a esto se modificará el aplicarReglasMovimiento, donde se controlara el tiempo posterior al haberla pisado para que se convierta en un tile pasable.

```
for(Iterator<Destructible> iterator = destructibles.iterator();
iterator.hasNext();){
    Destructible destructible = iterator.next();

    int tileXDestruc = (int) destructible.x / Tile.ancho;
    int tileYDestruc = (int) destructible.y / Tile.altura;
    if(jugador.colisiona(destructible) &&
destructible.tiempoPiso==0){
        destructible.tiempoPiso = System.currentTimeMillis();
    }
    if(destructible.tiempoPiso>0 && System.currentTimeMillis()-
destructible.tiempoPiso>=1000){
        mapaTiles[tileXDestruc][tileYDestruc] = new
Tile(null,Tile.PASABLE);
        iterator.remove();
        continue;
    }
}
```

Tiles escalera

Descripción

Incluir tiles de tipo escalera que permitan al jugador subir en el eje Y en línea recta.

Este tipo de tiles se representará en el mapa de nivel con la letra R

Creacion de la Escalera

Este se creará dentro del paquete escenarios.

```
public class Escalera extends Modelo {


    public Escalera(Context context, double x, double y) {
        super(context, x, y, 32, 40);

        this.y = y -altura/2;


        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.escalera);
    }


    public void dibujar(Canvas canvas){
```

```
        int yArriva = (int)  y - altura/2  - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho/2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }
}
```

Modificacion del Nivel

Se incluirá una lista de escaleras, se inicializara, se incluirá el código para poder ser reconocida y se crearan las variables booleanas padAbajoPulsado, padArribaPulsado

Posteriormente se modificará el método aplicarReglasMovimiento, donde este controlara la orientación del pad en el eje y para subir o bajar por la escalera.

```
…
for(Escalera escalera:escaleras){
    if(jugador.colisiona(escalera) && orientacionPadY>0 &&
padArribaPulsado &&
mapaTiles[tileXJugador][tileYJugadorSuperior].tipoDeColision==Tile.P
ASABLE){
        jugador.setVelocidadY(0);
        jugador.y -= 3;
        jugador.setVelocidadY(0);
        padArribaPulsado=false;
    }
    else if(jugador.colisiona(escalera) && orientacionPadY<0 &&
padAbajoPulsado){
        jugador.setVelocidadY(0);
        jugador.y += 3;
        jugador.setVelocidadY(0);
        padAbajoPulsado=false;
    }
    else if(jugador.colisiona(escalera) && orientacionPad!=0){
        jugador.setVelocidadY(0);
    }
    else if(jugador.colisiona(escalera)){
        jugador.setVelocidadY(0);
    }

}
…
```

Modificacion del GameView

Habra que definir cuando esta pulsado arriba o abajo, además de modificar como se reconocen las teclas por teclado cambiando el salto de la W a la E.

Cuando esta pulsado

```
if (accion[i] != ACTION_UP) {
    if(orientacion!=0) {
        pulsacionPadMover = true;
```

```
            nivel.orientacionPad = orientacion;
        }
        else if(orientacionY<0){
            pulsacionPadMover=false;
            nivel.orientacionPadY = orientacionY;
            nivel.entraPuerta=true;
            nivel.padAbajoPulsado=true;
        }
        else if(orientacionY>0){
            pulsacionPadMover=false;
            nivel.orientacionPadY = orientacionY;
            nivel.entraPuerta=true;
            nivel.padArribaPulsado=true;
        }
        else if(orientacionY==0){
            pulsacionPadMover=false;
            nivel.orientacionPadY = orientacionY;
            nivel.entraPuerta=false;
            nivel.padArribaPulsado=false;
        }
    }
```

Redefinir teclado

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    Log.v("Tecla","Tecla pulsada: "+keyCode);

    if( keyCode == 32) {
        nivel.orientacionPad = -0.5f;
    }
    if( keyCode == 29) {
        nivel.orientacionPad = 0.5f;
    }
    if( keyCode == 47) {
        nivel.orientacionPadY = -0.5f;
        nivel.padAbajoPulsado = true;
    }
    if( keyCode == 51) {
        nivel.orientacionPadY = 0.5f;
        nivel.padArribaPulsado = true;
    }
    if( keyCode == 33) {
        nivel.botonSaltarPulsado = true;
    }
    if( keyCode == 62) {
        nivel.botonDispararPulsado = true;
    }
    return super.onKeyDown(keyCode, event);
}
```

Enemigos más inteligentes

Descripción

Modificar el comportamiento de los enemigos para que actúen de una manera más inteligente, por ejemplo: que se muevan directamente hacia el jugador, traten de esquivar sus disparos, etc.

Modificaciones en Enemigo

Añadir dos variables nuevas a la clase enemigo para que estos puedan saltar

```
public boolean enElAire;
public double yAntes;
```

Modificaciones en Nivel

Donde se controla a los enemigos, cuando este en el rango del jugador y estén frente a el dispararan, si no están a la misma altura, es decir el jugador en el aire estos saltaran y verán si llegan a su altura.

```
if (tileXJugadorIzquierda - rango < tileXEnemigoIzquierda &&
        tileXJugadorIzquierda + rango > tileXEnemigoIzquierda) {

    if (jugador.colisiona(enemigo)) {
        if (Opciones.efectos)

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_JUGADOR_GOL
PEADO);
        if (jugador.golpeado() <= 0) {
            nivelPausado = true;
            mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.you_lose);
            inicializar();
            jugador.restablecerPosicionInicial();
            return;
        }
    }
    if (tileYJugadorCentro == tileYEnemigoCentro &&
jugador.getVelocidadX() > 0 && enemigo.velocidadX < 0) {
        long tiempo = System.currentTimeMillis();
        if (enemigo.estado == Estado.ACTIVO) {
            if (enemigo instanceof EnemigoAmpliacion) {
                DisparoEnemigo disparo = (DisparoEnemigo)
((EnemigoAmpliacion) enemigo).disparar(tiempo);
                if (disparo != null) {
                    disparosEnemigos.add(disparo);
                }
            }
        }
    }else if (tileYJugadorCentro == tileYEnemigoCentro &&
jugador.getVelocidadX() < 0 && enemigo.velocidadX > 0) {
        long tiempo = System.currentTimeMillis();
        if (enemigo.estado == Estado.ACTIVO) {
            if (enemigo instanceof EnemigoAmpliacion) {
                DisparoEnemigo disparo = (DisparoEnemigo)
((EnemigoAmpliacion) enemigo).disparar(tiempo);
                if (disparo != null) {
                    disparosEnemigos.add(disparo);
                }
            }
```

```
            }
        }else if (jugador.enElAire && jugador.getVelocidadX() < 0 &&
enemigo.velocidadX > 0 && !enemigo.enElAire) {
            enemigo.yAntes = enemigo.y;
            enemigo.y -= enemigo.altura;
            enemigo.enElAire = true;
        }else if (jugador.enElAire && jugador.getVelocidadX() > 0 &&
enemigo.velocidadX < 0 && !enemigo.enElAire) {
            enemigo.yAntes = enemigo.y;
            enemigo.y -= enemigo.altura;
            enemigo.enElAire = true;
        }
    }
    …
    if(enemigo.enElAire){
        if(enemigo.y + velocidadGravedad*2 < enemigo.yAntes) {
            enemigo.y += velocidadGravedad*2;
        }
        else {
            enemigo.y = enemigo.yAntes;
            enemigo.enElAire = false;
        }

    }
```

Punto de salvado

Descripción

Incluir un elemento que al colisionar con el permita al jugador salvar esa posición, en caso de perder el jugador debería comenzar desde ese punto del mapa (en lugar de desde el punto inicial del nivel). Este tipo de elemento se representará en el mapa de nivel con la letra A.

Creación del objeto SavePoint

Dentro del paquete recolectables creamos la clase SavePoint.

```
public class SavePoint extends Modelo {


    double xSalvada;
    double ySalvada;
    boolean salvado;

    public SavePoint(Context context, double x, double y) {
            super(context, x, y, 32,40);

            this.y =  y - altura/2;;

            xSalvada = x;
            ySalvada = y-Tile.altura;

            imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.flagyellow_down);
    }


    public void dibujar(Canvas canvas){
```

```
        int yArriba = (int)  y - altura / 2 - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho / 2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriba, xIzquierda
                + ancho, yArriba + altura);
        imagen.draw(canvas);

    }

    public double getxSalvada(){
        return xSalvada;
    }
    public double getySalvada(){
        return  ySalvada;
    }
    public boolean getSalvado(){
        return salvado;
    }
    public void setSalvado(boolean salvado){
        this.salvado=salvado;
    }

    public void changeImage() {
        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.flagyellow);
    }
}
```

Modificación de la clase Nivel

Se creará una lista de SavePoints ya que podrá haber más de uno por nivel y se rellenará con los SavePoints determinados en el mapa de tiles con una A. Se añadirá también una variable booleana para saber que ha sido pisada al menos un SavePoint.

Se modificará el método inicializar.

```
if(!checkPoint) {
    recolectables = new ArrayList<>();
    savePoints = new ArrayList<>();
    scrollEjeX = 0;
    scrollEjeY = 0;
    mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.description);
    nivelPausado = true;
    GameView.contador.reiniciarPuntuacion();
    enemigos = new LinkedList<Enemigo>();
    disparosJugador = new LinkedList<Disparo>();
    disparosEnemigos = new LinkedList<Disparo>();
    fondos = new Fondo[2];
    fondos[0] = new Fondo(context,
CargadorGraficos.cargarBitmap(context,
            R.drawable.capa1), 0);
    fondos[1] = new Fondo(context,
CargadorGraficos.cargarBitmap(context,
            R.drawable.capa2), 1f);
    iconosVida = new IconoVida[3];

    iconosVida[0] = new IconoVida(context, GameView.pantallaAncho *
```

```
0.05,
            GameView.pantallaAlto * 0.1);
    iconosVida[1] = new IconoVida(context, GameView.pantallaAncho *
0.15,
            GameView.pantallaAlto * 0.1);
    iconosVida[2] = new IconoVida(context, GameView.pantallaAncho *
0.25,
            GameView.pantallaAlto * 0.1);
    inicializarMapaTiles();
    scrollEjeY = (int) (altoMapaTiles() -
tilesEnDistanciaY(GameView.pantallaAlto)) * Tile.altura;
}
```

Se modificará el método aplicarReglasDeMovimiento.

```
...

for (Iterator<SavePoint> iterator = savePoints.iterator();
iterator.hasNext();) {
    SavePoint savePoint = iterator.next();
    if(jugador.colisiona(savePoint) && !savePoint.getSalvado()){

jugador.setPosicionInicial(savePoint.getxSalvada(),savePoint.getySal
vada());
        savePoint.setSalvado(true);
        savePoint.changeImage();
        checkPoint=true;
    }
}

…
```

Con esto ya tendríamos los puntos de guardado insertados en el videojuego.

Puertas

Descripción

Tipo de elemento especifico que trasladan al jugador de un punto a otro del nivel.

Las puertas se definirán en pareja, siempre estarán conectadas de dos en dos. Este tipo de elemento se representará en el mapa de nivel con los números 9,8,7,5,4.

Un número para cada par de puertas conectadas, por lo tanto podría haber un máximo de 10 puertas en el nivel.

Creación del objeto Puerta

Se creará dentro del paquete escenarios.

```
public class Puerta extends Modelo {

    public int numero;

    public Puerta(Context context, double x, double y, int numero) {
        super(context, x, y, 64, 40);
```

```
        this.numero = numero;
        imagen = CargadorGraficos.cargarDrawable(context,
R.drawable.puerta);
    }

    public double getX(){
        return x;
    }
    public double getY(){
        return y;
    }

    public void dibujar(Canvas canvas){
        int yArriva = (int)  y - altura  - Nivel.scrollEjeY;
        int xIzquierda = (int) x - ancho/2 - Nivel.scrollEjeX;

        imagen.setBounds(xIzquierda, yArriva, xIzquierda
                + ancho, yArriva + altura);
        imagen.draw(canvas);
    }
}
```

Modificación del Nivel

Primero se añadirán un conjunto de puertas, se inicializará vacío, posteriormente se dibujarán antes siempre que el personaje, se añadirán también a la inicialización del mapa de tiles, una variable booleana para saber cuándo entra por la puerta y posteriormente se modificara el método aplicarReglasMovimiento.

Estas modificaciones controlaran cuando el jugador puede entrar por las puertas, para ello debe colisionar con la puerta, pulsar abajo y haberla usado hace menos de 2 segundos.

```
for(Puerta puerta:puertas){
    for(Puerta puerta2:puertas){
        if(puerta!=puerta2){
            if(puerta.numero==puerta2.numero){
                if(jugador.colisiona(puerta) && entraPuerta &&
System.currentTimeMillis()-tiempoEspera >=2000){
                    jugador.x = puerta2.getX();
                    jugador.y = puerta2.getY()-Tile.altura;
                    entraPuerta=false;
                    tiempoEspera = System.currentTimeMillis();
                }
                else if(jugador.colisiona(puerta2) && entraPuerta &&
System.currentTimeMillis()-tiempoEspera >=2000){
                    jugador.x = puerta.getX();
                    jugador.y = puerta.getY()-Tile.altura;
                    entraPuerta=false;
                    tiempoEspera = System.currentTimeMillis();
                }
            }
        }
    }
}
```

Modificación del Pad

Habrá que crear una pequeña área para poder pulsar solo en el eje y para poder entrar en la puerta.

```
public int getOrientacionX(float cliclX) {
    int xPulsado = (int) (x - cliclX);
    return xPulsado>15 || xPulsado<-15 ? xPulsado : 0;
}
public int getOrientacionY(float clicY){
    int yPulsado = (int) (y - clicY);
    return yPulsado>15 || yPulsado<-15 ? yPulsado : 0;
}
```

Modificación del GameView

Por ultimo cuando se pulsa el pad habrá que controlar donde se pulsa para saber si debe entrar por la puerta, en caso de que así sea se cambiara la variable de entrar puerta a true.

```
float orientacion = pad.getOrientacionX(x[i]);
float orientacionY = pad.getOrientacionY(y[i]);
// Si almenosuna pulsacion está en el pad
if (accion[i] != ACTION_UP) {
    if(orientacion!=0) {
        pulsacionPadMover = true;
        nivel.orientacionPad = orientacion;
    }
    else if(orientacionY!=0){
        pulsacionPadMover=false;
        nivel.entraPuerta=true;
    }
}
```

Disparo con gravedad

Descripción

Aplicar la fuerza de gravedad al disparo del Jugador, debe "rebotar" contra el suelo e ir perdiendo la velocidad progresivamente, cuando pierde toda la velocidad desaparece.

Modificación del disparoJugador

Se añadirá una variable rebotes.

```
public int rebotes = 5;
```

Modificación del Nivel

Se modifica el método aplicarReglasMovimiento a la altura de los disparos del jugador. Estas modificaciones harán que el disparo vaya perdiendo velocidad y al final choque contra el suelo y pueda rebotar un 5 veces a lo sumo.

```
for (Iterator<Disparo> iterator = disparosJugador.iterator();
iterator.hasNext(); ) {

    DisparoJugador disparoJugador = (DisparoJugador)
```

```
iterator.next();

    int tileXDisparo = (int) disparoJugador.x / Tile.ancho;
    int tileYDisparoInferior =
            (int) (disparoJugador.y + disparoJugador.cAbajo) /
Tile.altura;

    int tileYDisparoSuperior =
            (int) (disparoJugador.y - disparoJugador.cArriba) /
Tile.altura;

    if (disparoJugador.velocidadX > 0) {
        // Tiene delante un tile pasable, puede avanzar.
        if (tileXDisparo + 1 <= anchoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo +
1][tileYDisparoInferior].tipoDeColision
                        == Tile.PASABLE &&
                mapaTiles[tileXDisparo +
1][tileYDisparoSuperior].tipoDeColision
                        == Tile.PASABLE) {

            disparoJugador.x += disparoJugador.velocidadX;
            disparoJugador.y += velocidadGravedad;

        }else if (tileXDisparo + 1 <= anchoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo +
1][tileYDisparoInferior].tipoDeColision
                        == Tile.SOLIDO &&
                mapaTiles[tileXDisparo +
1][tileYDisparoSuperior].tipoDeColision
                        == Tile.PASABLE && disparoJugador.rebotes>0
) {

            disparoJugador.x += disparoJugador.velocidadX/2;
            disparoJugador.y -= velocidadGravedad;
            disparoJugador.rebotes--;

        } else if (tileXDisparo <= anchoMapaTiles() - 1) {

            int TileDisparoBordeDerecho = tileXDisparo * Tile.ancho
+ Tile.ancho;
            double distanciaX =
                    TileDisparoBordeDerecho - (disparoJugador.x +
disparoJugador.cDerecha);

            if (distanciaX > 0) {
                double velocidadNecesaria =
                        Math.min(distanciaX,
disparoJugador.velocidadX);
                disparoJugador.x += velocidadNecesaria;
            } else {
                iterator.remove();
                continue;
            }
        }
    }

    // izquierda
    if (disparoJugador.velocidadX <= 0) {
        if (tileXDisparo - 1 >= 0 &&
```

```
                tileYDisparoSuperior < altoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo -
1][tileYDisparoSuperior].tipoDeColision ==
                        Tile.PASABLE &&
                mapaTiles[tileXDisparo -
1][tileYDisparoInferior].tipoDeColision ==
                        Tile.PASABLE) {

            disparoJugador.x += disparoJugador.velocidadX;
            disparoJugador.y += velocidadGravedad;
            // No tengo un tile PASABLE detras
            // o es el INICIO del nivel o es uno SOLIDO
        }else if(tileXDisparo - 1 >= 0 &&
                tileYDisparoSuperior < altoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo -
1][tileYDisparoSuperior].tipoDeColision ==
                        Tile.PASABLE &&
                mapaTiles[tileXDisparo -
1][tileYDisparoInferior].tipoDeColision ==
                        Tile.SOLIDO && disparoJugador.rebotes>0){

            disparoJugador.x += disparoJugador.velocidadX / 2;
            disparoJugador.y -= velocidadGravedad;
            disparoJugador.rebotes--;

        }else if (tileXDisparo >= 0) {
            // Si en el propio tile del jugador queda espacio para
            // avanzar más, avanzo
            int TileDisparoBordeIzquierdo = tileXDisparo *
Tile.ancho;
            double distanciaX =
                    (disparoJugador.x - disparoJugador.cIzquierda) -
TileDisparoBordeIzquierdo;

            if (distanciaX > 0) {
                double velocidadNecesaria =
                        Utilidades.proximoACero(-distanciaX,
disparoJugador.velocidadX);
                disparoJugador.x += velocidadNecesaria;
            } else {
                iterator.remove();
                continue;
            }
        }
    }
    for (Enemigo enemigo : enemigos) {
        if (disparoJugador.colisiona(enemigo)) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_ENEMIGO_GOL
PEADO);
            enemigo.destruir();
            iterator.remove();
            break;
        }
    }
    for(Plataforma plataforma:plataformas){
        if(disparoJugador.colisiona(plataforma)){
            iterator.remove();
            continue;
        }
```

```
        }
        for(Caja caja:cajas){
            if(disparoJugador.colisiona(caja)){
                iterator.remove();
                continue;
            }
        }
        for(Puerta puerta:puertas){
            if(disparoJugador.colisiona(puerta)){
                iterator.remove();
                continue;
            }
        }
}
```

Disparo direccional

Descripción

Cambiar el botón de disparar por un joystick direccional que permita especificar el ángulo del disparo en el eje X e Y.

Modificaciones en Disparo

```
public double velocidadY = 5;
public int angulo;
public int impulsos=15;
```

Modificaciones BotonDisparo

Habra que añadir 3 metodos, dos para controlar donde se pulsan en x e y, un tercero para controlar el angulo con el que saldrá.

```
public int getOrientacionX(float cliclX) {
    return (int) (x - cliclX);

}
public int getOrientacionY(float clicY){
    return (int) (y - clicY);
}

public int calcularAngulo(float orientacionX,float orientacionY){
    if(orientacionX<15 && orientacionX>-15){
        if(orientacionY>25){
            return 90;
        }
        if(orientacionY<-20){
            return -90;
        }
    }
    else if(orientacionX>15 || orientacionX<-15){
        if(orientacionY>15){
            return 45;
        }
        if(orientacionY>5 && orientacionY<15){
            return 0;
        }
        if(orientacionY<-15){
            return -45;
```

```
            }
        }
        return 0;
    }
```

Modificaciones Nivel

Habra que modificar como se comportarn los disparos al moverse, dependiendo del angulo con el que venga el disparo saldrá de una manera u otra, siempre controlando que no se salga de los limites del mapa. Ademas se le quitaran impulsos al disparo si este va hacia arriba, para que pueda caer posteriormente

```
for (Iterator<Disparo> iterator = disparosJugador.iterator();
iterator.hasNext(); ) {

    DisparoJugador disparoJugador = (DisparoJugador)
iterator.next();

    int tileXDisparo = (int) disparoJugador.x / Tile.ancho;
    int tileYDisparoInferior =
            (int) (disparoJugador.y + disparoJugador.cAbajo) /
Tile.altura;

    int tileYDisparoSuperior =
            (int) (disparoJugador.y - disparoJugador.cArriba) /
Tile.altura;

    if (disparoJugador.velocidadX > 0) {
        // Tiene delante un tile pasable, puede avanzar.
        if (tileXDisparo + 1 <= anchoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo +
1][tileYDisparoInferior].tipoDeColision
                        == Tile.PASABLE &&
                mapaTiles[tileXDisparo +
1][tileYDisparoSuperior].tipoDeColision
                        == Tile.PASABLE) {

            if(disparoJugador.angulo==0) {
    disparoJugador.x += disparoJugador.velocidadX;
}
else if(disparoJugador.angulo==45 && disparoJugador.y-3>0 &&
disparoJugador.impulsos>0) {
    disparoJugador.x += disparoJugador.velocidadX;
    disparoJugador.y -= disparoJugador.velocidadY;
    disparoJugador.impulsos--;
}
else if(disparoJugador.angulo==90 && disparoJugador.y-3>0 &&
disparoJugador.impulsos>0) {
    disparoJugador.y -= disparoJugador.velocidadY;
    disparoJugador.impulsos--;
}
else if(disparoJugador.angulo==-45 &&
disparoJugador.y+disparoJugador.velocidadY<GameView.pantallaAlto+60)
{
    disparoJugador.x += disparoJugador.velocidadX;
    disparoJugador.y += disparoJugador.velocidadY;
}
else if(disparoJugador.angulo==-90 &&
```

```
disparoJugador.y+disparoJugador.velocidadY<GameView.pantallaAlto+60)
{
    disparoJugador.y +=  disparoJugador.velocidadY-
velocidadGravedad;
}
else if((disparoJugador.angulo==45 || disparoJugador.angulo==90) &&
disparoJugador.y-3<0){
    iterator.remove();
    continue;
}
else if(disparoJugador.impulsos<=0 && disparoJugador.angulo!=90 &&
disparoJugador.angulo!=-90){
    disparoJugador.x += disparoJugador.velocidadX;
}
if(disparoJugador.y+velocidadGravedad<GameView.pantallaAlto+60)
    disparoJugador.y += velocidadGravedad;
else{
    iterator.remove();
    continue;
}

        }else if (tileXDisparo + 1 <= anchoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo +
1][tileYDisparoInferior].tipoDeColision
                        == Tile.SOLIDO &&
                mapaTiles[tileXDisparo +
1][tileYDisparoSuperior].tipoDeColision
                        == Tile.PASABLE && disparoJugador.rebotes>0
) {

            disparoJugador.x += disparoJugador.velocidadX/2;
            disparoJugador.y -= velocidadGravedad;
            disparoJugador.rebotes--;

        } else if (tileXDisparo <= anchoMapaTiles() - 1) {

            int TileDisparoBordeDerecho = tileXDisparo * Tile.ancho
+ Tile.ancho;
            double distanciaX =
                    TileDisparoBordeDerecho - (disparoJugador.x +
disparoJugador.cDerecha);

            if (distanciaX > 0) {
                double velocidadNecesaria =
                        Math.min(distanciaX,
disparoJugador.velocidadX);
                disparoJugador.x += velocidadNecesaria;
            } else {
                iterator.remove();
                continue;
            }
        }
    }

    // izquierda
    if (disparoJugador.velocidadX <= 0) {
        if (tileXDisparo - 1 >= 0 &&
                tileYDisparoSuperior < altoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo -
1][tileYDisparoSuperior].tipoDeColision ==
```

```
                        Tile.PASABLE &&
                mapaTiles[tileXDisparo -
1][tileYDisparoInferior].tipoDeColision ==
                        Tile.PASABLE) {

            if(disparoJugador.angulo==0) {
    disparoJugador.x += disparoJugador.velocidadX;
}
else if(disparoJugador.angulo==45 && disparoJugador.y-3>0 &&
disparoJugador.impulsos>0) {
    disparoJugador.x += disparoJugador.velocidadX;
    disparoJugador.y -= disparoJugador.velocidadY;
    disparoJugador.impulsos--;
}
else if(disparoJugador.angulo==90 && disparoJugador.y-3>0 &&
disparoJugador.impulsos>0) {
    disparoJugador.y -= 4;
    disparoJugador.impulsos--;
}
else if(disparoJugador.angulo==-45 &&
disparoJugador.y+disparoJugador.velocidadY<GameView.pantallaAlto+60)
{
    disparoJugador.x += disparoJugador.velocidadX;
    disparoJugador.y +=  disparoJugador.velocidadY;
}
else if(disparoJugador.angulo==-90 &&
disparoJugador.y+disparoJugador.velocidadY<GameView.pantallaAlto+60)
{
    disparoJugador.y +=  disparoJugador.velocidadY-
velocidadGravedad;
}
else if((disparoJugador.angulo==45 || disparoJugador.angulo==90) &&
disparoJugador.y-3<0){
    iterator.remove();
    continue;
}
else if(disparoJugador.impulsos<=0 && disparoJugador.angulo!=90 &&
disparoJugador.angulo!=-90){
    disparoJugador.x += disparoJugador.velocidadX;
}
if(disparoJugador.y+velocidadGravedad<GameView.pantallaAlto+60)
    disparoJugador.y += velocidadGravedad;
else{
    iterator.remove();
    continue;
}

            // No tengo un tile PASABLE detras
            // o es el INICIO del nivel o es uno SOLIDO
        }else if(tileXDisparo - 1 >= 0 &&
                tileYDisparoSuperior < altoMapaTiles() - 1 &&
                mapaTiles[tileXDisparo -
1][tileYDisparoSuperior].tipoDeColision ==
                        Tile.PASABLE &&
                mapaTiles[tileXDisparo -
1][tileYDisparoInferior].tipoDeColision ==
                        Tile.SOLIDO && disparoJugador.rebotes>0){

            disparoJugador.x += disparoJugador.velocidadX / 2;
            disparoJugador.y -= velocidadGravedad;
```

```
            disparoJugador.rebotes--;

        }else if (tileXDisparo >= 0) {
            // Si en el propio tile del jugador queda espacio para
            // avanzar más, avanzo
            int TileDisparoBordeIzquierdo = tileXDisparo *
Tile.ancho;
            double distanciaX =
                    (disparoJugador.x - disparoJugador.cIzquierda) -
TileDisparoBordeIzquierdo;

            if (distanciaX > 0) {
                double velocidadNecesaria =
                        Utilidades.proximoACero(-distanciaX,
disparoJugador.velocidadX);
                disparoJugador.x += velocidadNecesaria;
            } else {
                iterator.remove();
                continue;
            }
        }
    }
```

Modificaciones GameView

Por ultimo habrá que modificar como se disparara ya que se calculará un angulo aproximado según donde se pulse en el pad.

```
int anguloDisparo =
botonDisparar.calcularAngulo(orientacion,orientacionY);

if (accion[i] == ACTION_DOWN) {
    if(orientacion!=0 && orientacionY!=0) {
        nivel.botonDispararPulsado = true;
        nivel.orientacionDisparoY = orientacionY;
        nivel.anguloDisparo=anguloDisparo;
    }
    else if(orientacion!=0 && orientacionY==0) {
        nivel.botonDispararPulsado = true;
        nivel.orientacionDisparoY = orientacionY;
        nivel.anguloDisparo=anguloDisparo;
    }
    else{
        nivel.botonDispararPulsado=true;
    }

}
```

Completar la interfaz del juego

Descripción

Incluir un menú de inicio, selección de nivel y efectos de sonido durante el juego.

Creación de las pantallas

MenuActivity

```
public class MenuActivity extends Activity implements
View.OnClickListener {

    ImageButton nuevoJuego;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Establecer el control de sonido multimedia como
predefinido
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);

        // Pantalla completa, sin titulo
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(LayoutParams.FLAG_FULLSCREEN,
                LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_menu);

        nuevoJuego = (ImageButton) findViewById(R.id.imageButton);
        nuevoJuego.setOnClickListener(this);

    }


    @Override
    public void onClick(View v) {
        if(v==nuevoJuego){
            Intent actividadJuego = new Intent(MenuActivity.this,
                    SeleccionModo.class);
            startActivity(actividadJuego);
            finish();
        }
    }
}
```

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_menu"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.plataformas.MenuActivity"
    android:background="@drawable/fondo_menu_principal">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        app:srcCompat="@drawable/boton_play"
        android:id="@+id/imageButton"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/boton_play" />
</RelativeLayout>
```

SeleccionModoActivity

```
public class SeleccionModo extends Activity implements
View.OnClickListener {

    ImageButton unJugador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Establecer el control de sonido multimedia como
predefinido
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);

        // Pantalla completa, sin titulo
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_seleccion_modo);

        unJugador = (ImageButton)
findViewById(R.id.unJugadorButton);
        unJugador.setOnClickListener(this);

    }


    @Override
    public void onClick(View v) {
        if(v==unJugador){
            Intent actividadJuego = new Intent(SeleccionModo.this,
                    MainActivity.class);
            actividadJuego.putExtra("Modo","Un jugador");
            startActivity(actividadJuego);
            finish();
        }
    }
}
```

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_seleccion_modo"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.plataformas.SeleccionModo"
    android:background="@drawable/fondo_seleccionar_nivel">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/boton_one"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="54dp"
        android:id="@+id/unJugadorButton"
        android:background="@drawable/boton_one" />

</RelativeLayout>
```

Creación del Gestor de audio

Dentro del paquete gestores.

```
public class GestorAudio implements MediaPlayer.OnPreparedListener {
    public static final int SONIDO_DISPARO_JUGADOR = 1;
    public static final int SONIDO_SALTO_JUGADOR = 2;
    public static final int SONIDO_ENEMIGO_GOLPEADO = 3;
    public static final int SONIDO_JUGADOR_GOLPEADO = 4;
    // Pool de sonidos, para efectos de sonido.
    // Suele fallar el utilizar ficheros de sonido demasiado grandes
    private SoundPool poolSonidos;
    private HashMap<Integer, Integer> mapSonidos;
    private Context contexto;
    // Media Player para bucle de sonido de fondo1.
    private MediaPlayer sonidoAmbiente;
    private AudioManager gestorAudio;

    private static GestorAudio instancia = null;

    public static GestorAudio getInstancia(Context contexto,
                                           int idMusicaAmbiente) {
        synchronized (GestorAudio.class) {
            if (instancia == null) {
                instancia = new GestorAudio();
                instancia.initSounds(contexto, idMusicaAmbiente);
            }
            return instancia;
        }
    }


    public static GestorAudio getInstancia() {
        return instancia;
    }

    private GestorAudio() {
    }
```

```
    public void initSounds(Context contexto, int idMusicaAmbiente) {
        this.contexto = contexto;
        poolSonidos = new SoundPool(4, AudioManager.STREAM_MUSIC,
100);
        mapSonidos = new HashMap<Integer, Integer>();
        gestorAudio = (AudioManager) contexto
                .getSystemService(Context.AUDIO_SERVICE);
        sonidoAmbiente = MediaPlayer.create(contexto,
idMusicaAmbiente);
        sonidoAmbiente.setLooping(true);
        sonidoAmbiente.setVolume(1, 1);
    }

    public void reproducirMusicaAmbiente() {
        try {
            if (!sonidoAmbiente.isPlaying()) {
                try {
                    sonidoAmbiente.setOnPreparedListener(this);
                    sonidoAmbiente.prepareAsync();
                } catch (Exception e) {
                }
            }
        } catch (Exception e) {
        }
    }

    public void pararMusicaAmbiente() {
        if (sonidoAmbiente.isPlaying()) {
            sonidoAmbiente.stop();
        }
    }

    @Override
    public void onPrepared(MediaPlayer mp) {
        mp.start();
    }

    public void registrarSonido(int index, int SoundID) {
        mapSonidos.put(index, poolSonidos.load(contexto, SoundID,
1));
    }

    public void reproducirSonido(int index) {
        float volumen =

gestorAudio.getStreamVolume(AudioManager.STREAM_MUSIC);

        volumen =
                volumen /
gestorAudio.getStreamMaxVolume(AudioManager.STREAM_MUSIC);

        poolSonidos.play(
                (Integer) mapSonidos.get(index),
                volumen, volumen, 1, 0, 1f);
    }

}
```

Modificación del GameView

Creación de la variable publica gestorAudio y su inicialización en el constructor llamando al siguiente método.

```
public void inicializarGestorAudio(Context context) {
    gestorAudio = GestorAudio.getInstancia(context,
R.raw.musica_fondo);
    gestorAudio.reproducirMusicaAmbiente();
    gestorAudio.registrarSonido(GestorAudio.SONIDO_DISPARO_JUGADOR,
            R.raw.lanzar_objeto);

gestorAudio.registrarSonido(GestorAudio.SONIDO_SALTO_JUGADOR,R.raw.s
alto_jugador);

gestorAudio.registrarSonido(GestorAudio.SONIDO_ENEMIGO_GOLPEADO,R.ra
w.disparo_golpea);

gestorAudio.registrarSonido(GestorAudio.SONIDO_JUGADOR_GOLPEADO,R.ra
w.jugador_golpeado);
}
```

Modificación del Nivel

Llamar a los diferentes sonidos donde se produzcan las interacciones

Saltar

```
if (botonSaltarPulsado) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_SALTO_JUGAD
OR);
    botonSaltarPulsado = false;
}
```

Disparar

```
if (botonDispararPulsado) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_DISPARO_JUG
ADOR);
    disparosJugador.add(new DisparoJugador(context, jugador.x,
jugador.y, jugador.orientacion));
    botonDispararPulsado = false;
}
```

Ser golpeado por disparo enemigo

```
if (disparoEnemigo.colisiona(jugador)) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_JUGADOR_GOL
PEADO);
    if (jugador.golpeado() <= 0) {
        nivelPausado = true;
        mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.you_lose);
        inicializar();
        jugador.restablecerPosicionInicial();
        return;
```

```
    }
}
```

Ser golpeado por enemigo

```
if (jugador.colisiona(enemigo)) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_JUGADOR_GOLPEADO);
    if (jugador.golpeado() <= 0) {
        nivelPausado = true;
        mensaje = CargadorGraficos.cargarBitmap(context,
R.drawable.you_lose);
        inicializar();
        jugador.restablecerPosicionInicial();
        return;
    }
}
```

Golpear enemigo con disparo

```
if (disparoJugador.colisiona(enemigo)) {

gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_ENEMIGO_GOLPEADO);
    enemigo.destruir();
    iterator.remove();
    break;
}
```

Otras ampliaciones propuestas por el alumno

Descripción

Menu de opciones previo al empezar el juego donde se puede elegir si quitar el volumen de la música, quitar el volumen de los efectos o aumentar o dismunir la dificultad, si se activa la dificultad menos vidas si no, mas vidas.

Modificacion de la pantalla Selección de Modo

```
if(v==unJugador){
    Intent actividadJuego = new Intent(SeleccionModo.this,
            OpcionesActivity.class);
    startActivity(actividadJuego);
    finish();
}
```

Creacion de la pantalla opciones

Activity

```
public class OpcionesActivity extends Activity implements
View.OnClickListener {

    ImageButton dificultad;
    ImageButton musica;
    ImageButton efectos;
    ImageButton empezar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Establecer el control de sonido multimedia como
predefinido
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);

        // Pantalla completa, sin titulo
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_opciones);

        dificultad = (ImageButton) findViewById(R.id.dificultad);
        musica = (ImageButton) findViewById(R.id.musica);
        efectos = (ImageButton) findViewById(R.id.efectos);
        empezar = (ImageButton) findViewById(R.id.empezar);
        dificultad.setOnClickListener(this);
        musica.setOnClickListener(this);
        efectos.setOnClickListener(this);
        empezar.setOnClickListener(this);

    }


    @Override
    public void onClick(View v) {
        if(v==empezar){
            Intent actividadJuego = new
Intent(OpcionesActivity.this,
                    MainActivity.class);
            startActivity(actividadJuego);
            finish();
        }
        if(v==dificultad) {
            Opciones.dificultad =  Opciones.dificultad? false:true;
            if( Opciones.dificultad)

dificultad.setBackgroundResource(R.drawable.dificultad_on);
            else

dificultad.setBackgroundResource(R.drawable.dificultad_off);
        }
        if(v==musica){
            Opciones.musica = Opciones.musica? false:true;
            if(Opciones.musica)

musica.setBackgroundResource(R.drawable.boton_music_on);
            else

musica.setBackgroundResource(R.drawable.boton_music_off);
```

```
        }
        if(v==efectos){
            Opciones.efectos = Opciones.efectos ?false:true;
            if(Opciones.efectos)

efectos.setBackgroundResource(R.drawable.boton_effects_on);
            else

efectos.setBackgroundResource(R.drawable.boton_effects_off);
        }
    }
}
```

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_seleccion_modo"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.plataformas.SeleccionModo"
    android:background="@drawable/fondo_opciones">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/boton_play"
        android:id="@+id/empezar"
        android:background="@drawable/boton_play"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/dificultad_off"
        android:id="@+id/dificultad"
        android:background="@drawable/dificultad_off"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/boton_effects_on"
        android:id="@+id/efectos"
        android:background="@drawable/boton_effects_on"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="50dp" />
```

```
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/boton_music_on"
        android:id="@+id/musica"
        android:background="@drawable/boton_music_on"
        android:layout_above="@+id/efectos"
        android:layout_alignLeft="@+id/efectos"
        android:layout_alignStart="@+id/efectos" />

</RelativeLayout>
```

Creacion del objeto Opciones

```
public class Opciones {

    public static boolean musica=true;
    public static boolean dificultad;
    public static boolean efectos=true;

}
```

Modificaciones GameView

```
protected void inicializar() throws Exception {
    contador = new Contador(context);
    pad = new Pad(context);
    botonSaltar = new BotonSaltar(context);
    botonDisparar = new BotonDisparar(context);
    nivel = new Nivel(context,numeroNivel);
    nivel.gameView = this;
    if(!Opciones.musica)
        gestorAudio.pararMusicaAmbiente();
}
```

Modificaciones Nivel

Efectos de sonido, donde haya una reproducción de un sonido insertar de la siguiente manera

```
if(Opciones.efectos)
gameView.gestorAudio.reproducirSonido(GestorAudio.SONIDO_ENEMIGO_GOL
PEADO);
```

Dificultad

```
if (!checkPoint)
    if(Opciones.dificultad)
        jugador = new Jugador(context, xCentroAbajoTile,
```

```
yCentroAbajoTile,1);
    else
        jugador = new Jugador(context, xCentroAbajoTile,
yCentroAbajoTile,3);
```

Video de muestra de ampliaciones

Video con sonido y sin activar la dificultad
https://unioviedo-my.sharepoint.com/personal/uo227602_uniovi_es/_layouts/15/guestaccess.aspx?guestaccesstoken=2E5kd%2bBojm1s9mhk14kd9nEfpS3i4ZNuUPjLyF8XPHU%3d&docid=16a4608f2eaa94507a6919fa1d97efd31&rev=1

Video sin sonidos y activando la dificultad
https://unioviedo-my.sharepoint.com/personal/uo227602_uniovi_es/_layouts/15/guestaccess.aspx?guestaccesstoken=FbdzbrYDVE200hkITxVwNR5pxtFM9PaVXX1zi9kPYEI%3d&docid=1d3a2ee4681ec4f0e8bbe266f2d0f6488&rev=1