Adrián Jiménez Villarreal

71904816P | UO227602@uniovi.es

Juego Plataformas

Ampliaciones

Contenido

[Ampliaciones 2](#_Toc465613524)

[Nuevos tipos de enemigos 3](#_Toc465613525)

[Enemigo que salta o vuela 3](#_Toc465613526)

[Enemigo que te persigue 3](#_Toc465613527)

[Scroll en el eje Y 3](#_Toc465613528)

[Items recolectables 3](#_Toc465613529)

[Descripción 3](#_Toc465613530)

[Creación de la clase recolectable 3](#_Toc465613531)

[Creación de la clase contador 5](#_Toc465613532)

[Modificación del nivel 5](#_Toc465613533)

[Plataformas móviles 6](#_Toc465613534)

[Caja que se puedan arrastrar 6](#_Toc465613535)

[Tiles destruibles 6](#_Toc465613536)

[Tiles escalera 6](#_Toc465613537)

[Tiles solidos con inclinación 6](#_Toc465613538)

[Enemigos más inteligentes 6](#_Toc465613539)

[Punto de salvado 6](#_Toc465613540)

[Puertas 6](#_Toc465613541)

[Disparo con gravedad 6](#_Toc465613542)

[Disparo direccional 6](#_Toc465613543)

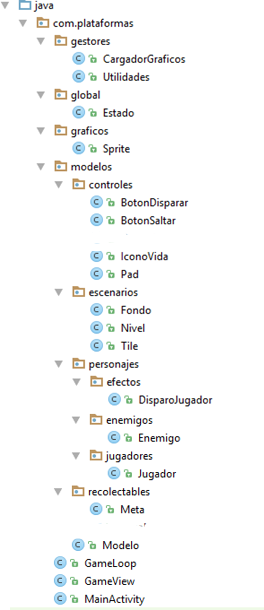
[Completar la interfaz del juego 6](#_Toc465613544)

[Multijugador con teclado 6](#_Toc465613545)

[Otras ampliaciones propuestas por el alumno 6](#_Toc465613546)

# 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.

## Enemigo que te persigue

## 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

## Caja que se puedan arrastrar

## Tiles destruibles

## Tiles escalera

## Tiles solidos con inclinación

## Enemigos más inteligentes

## 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. 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.getySalvada());  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

## Disparo con gravedad

## Disparo direccional

## Completar la interfaz del juego

## Multijugador con teclado

## Otras ampliaciones propuestas por el alumno

# Video de muestra de ampliaciones