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)

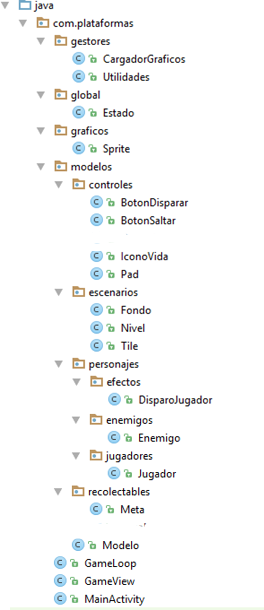
[Completar la interfaz del juego 6](#_Toc465613544)

[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

## Enemigo que salta o vuela

## Enemigo que te persigue

## Scroll en el eje Y

## 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;  }   @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

## Puertas

## Disparo con gravedad

## Disparo direccional

## Completar la interfaz del juego

## Multijugador con teclado

## Otras ampliaciones propuestas por el alumno

# Video de muestra de ampliaciones