

SPACE INVADERS



Iñigo Cemborain y Eider Labiano 1ºB DAM

Índice

| | |
|---|-----------|
| Implementación, prueba de sprites y creación de niveles | Pág 3-8 |
| Creación de clases e implementación de interfaces | Pág 9-17 |
| Creación de enumeraciones | Pág 18 |
| Implementación de audios | Pág 19-21 |

Implementación de las fotos (Sprites) al proyecto, cada una con su atributo:
Sprites hecho en: <https://www.piskelapp.com/p/create/sprite>

```
AppConsts.java x
27 //endregion
28
29 //region EnemyGenerator
1 usage
30 final static int LEVELS = 8;
8 usages
```

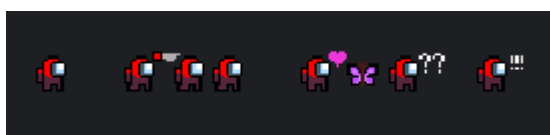
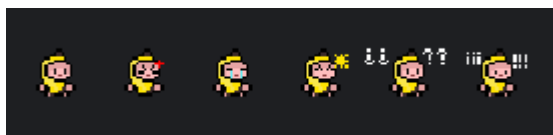
La constante muestra los distintos niveles existentes, como tenemos 8 distintos pondremos 8.

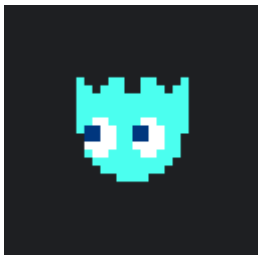
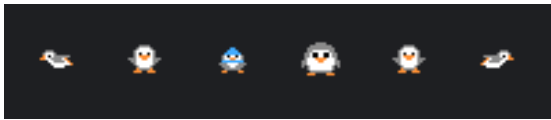
```
AppConsts.java x
18 usages
35 static final Image CAT = new Image(Main.class.getResource( name: "sprite/cat2.0.png").toString());
20 usages
36 static final Image AMONUS = new Image(Main.class.getResource( name: "sprite/AMONGUS.png").toString());
12 usages
37 static final Image PINGU = new Image(Main.class.getResource( name: "sprite/pingu.png").toString());
20 usages
```

```
AppConsts.java x
2 usages
53 public static final Image FANTASMA = new Image(Main.class.getResource( name: "sprite/Fantasma.png").toString());
16 usages
54 static final Image PACMAN = new Image(Main.class.getResource( name: "sprite/Pac.png").toString());
1 usage
55 static final Image EXPLOSION_PACMAN = new Image(Main.class.getResource( name: "sprite/Packman Muerte.png").toString());
56
```

Los enemigos se implementan en la clase AppConsts porque no va a variar el sprite, sin embargo sus cualidades se describirán en otras clases.

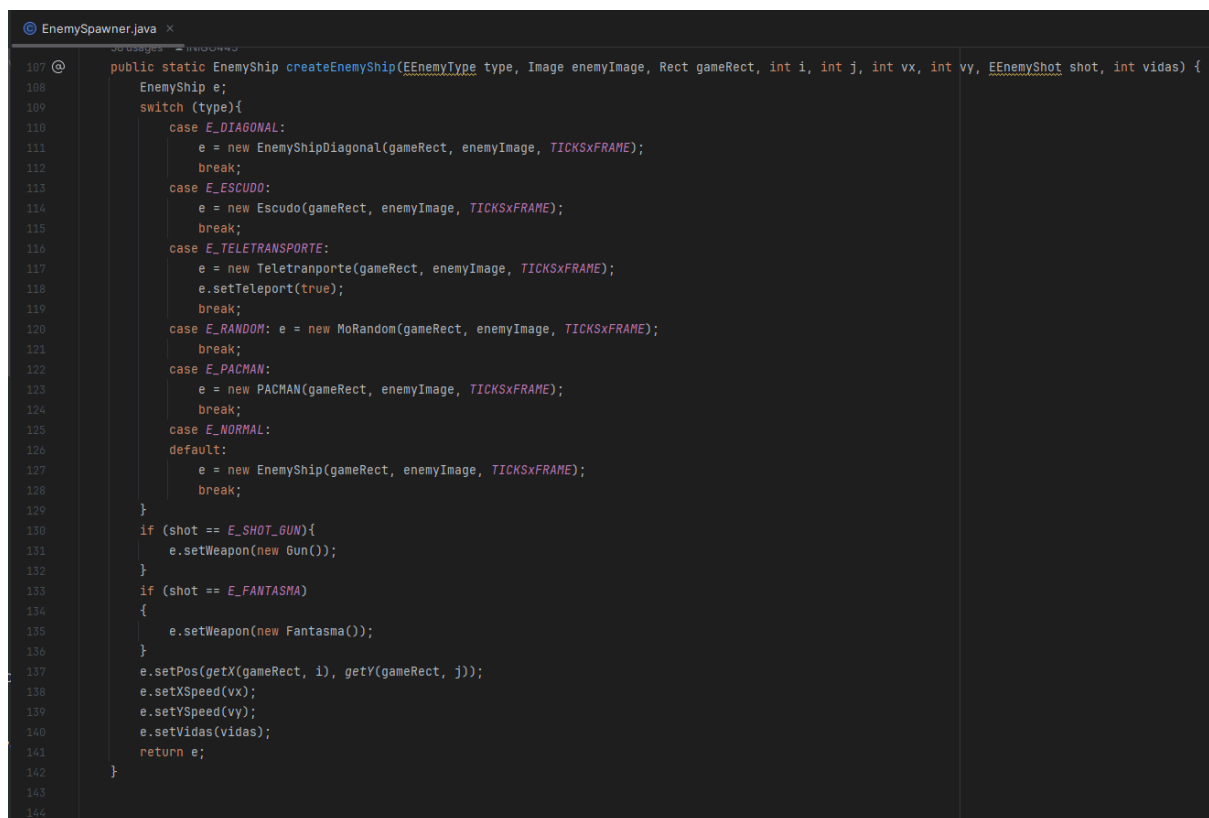
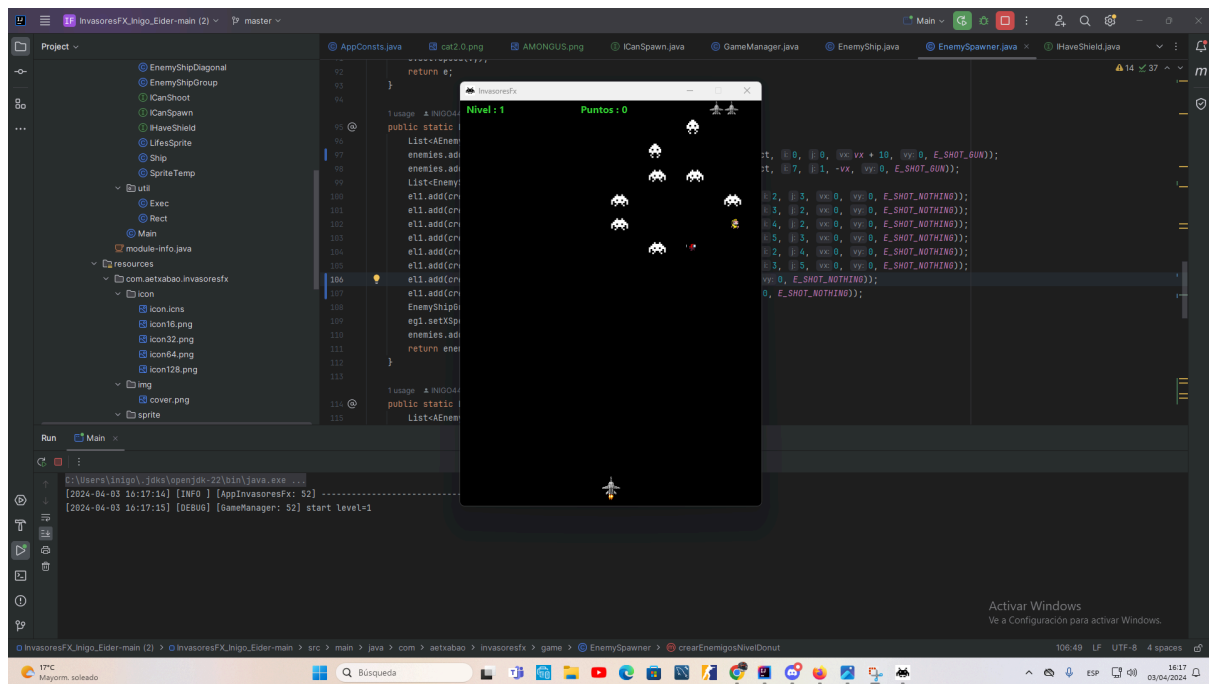
Imágenes a tamaño 262 x 32





Ahora, los añadimos (Para probar que la imagen animada aparece) a un nivel.

```
102 return e;  
103 }  
104  
105 //image: AMONGUS.png  
106 public static List<Enemy> createEnemiesNivelDonut(Rect gameRect) {  
107     List<Enemy> enemies = new ArrayList<>();  
108     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 0, vx + 10, vy + 0, E_SHOT_GUN));  
109     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 7, -vx, vy + 0, E_SHOT_GUN));  
110     List<EnemyShip> e11 = new ArrayList<>();  
111     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 2, 3, vx, 0, E_SHOT_NOTHING));  
112     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 3, 2, vx, 0, E_SHOT_NOTHING));  
113     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 4, 2, vx, 0, E_SHOT_NOTHING));  
114     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 5, 3, vx, 0, E_SHOT_NOTHING));  
115     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 2, 4, vx, 0, E_SHOT_NOTHING));  
116     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 3, 5, vx, 0, E_SHOT_NOTHING));  
117     e11.add(createEnemyShip(E_NORMAL, AMONGUS, gameRect, 4, 5, vx, 0, E_SHOT_NOTHING));  
118     e11.add(createEnemyShip(E_NORMAL, CAT, gameRect, 5, 4, vx, 0, E_SHOT_NOTHING));  
119     EnemyShipGroup egl = new EnemyShipGroup(gameRect, e11);  
120     egl.setSpeed(vx);  
121     enemies.add(egl);  
122     return enemies;  
123 }  
124  
125 //image: AMONGUS.png  
126 public static List<Enemy> createEnemiesNivelPaquito(Rect gameRect) {  
127     List<Enemy> enemies = new ArrayList<>();  
128     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 0, vx, vy + 0, E_SHOT_GUN));  
129     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 7, -vx, vy + 0, E_SHOT_GUN));  
130     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 3, 2, vx, vy + 0, E_SHOT_GUN));  
131     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_2, gameRect, 0, 4, 3, -vx, vy + 0, E_SHOT_GUN));  
132     List<EnemyShip> e11 = new ArrayList<>();  
133     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_1, gameRect, 2, 4, vx, vy + 0, E_SHOT_NOTHING));  
134     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_1, gameRect, 3, 4, vx, vy + 0, E_SHOT_NOTHING));  
135     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_1, gameRect, 4, 4, vx, vy + 0, E_SHOT_NOTHING));  
136     e11.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_1, gameRect, 5, 4, vx, vy + 0, E_SHOT_NOTHING));  
137     EnemyShipGroup egl = new EnemyShipGroup(gameRect, e11);  
138     egl.setSpeed(vx);  
139     enemies.add(egl);  
140     return enemies;  
141 }  
142
```



Como se observa, tenemos un segundo constructor que será sobrecargado porque:

- En el caso escudo crea un enemigo de tipo escudo
- En el caso teletransporte crea un enemigo que puede teletransporte al recibir daño
- En el caso random el enemigo que recibe daño cambia la velocidad de movimiento
- En el caso pacman se genera un enemigo pacman que tendrá como peculiaridad disparar fantasmas (de tipo fantasma) a diferencia de otros enemigos

```
EnemySpawner.java x
43 @ > private static int getY(Rect gameRect, int j) { return gameRect.top + j*gameRect.height()/ m; }
46
2 usages INIGO445
47 public static List<AEnemy> createEnemies(Rect gameRect, int level) {
48     List<AEnemy> enemies = new ArrayList<>();
49     level = level % LEVELS;
50
51     switch (level){
52         case 1:
53             enemies = crearEnemigosNivelEiderLevel(gameRect);
54             break;
55         case 2:
56             enemies = crearEnemigosNivelCemboLevel(gameRect);
57             break;
58         case 3:
59             enemies = nivelFantasmas(gameRect);
60             break;
61         case 4:
62             enemies = nivelDificilNAI(gameRect);
63             break;
64         case 5:
65             enemies = crearEnemigosNivelFrenesi(gameRect);
66             break;
67         case 6:
68             enemies = crearEnemigosNivelDonut(gameRect);
69             break;
70         case 7:
71             enemies = crearEnemigosNivelPaquito(gameRect);
72             break;
73         default:
74             enemies = crearEnemigosNivelPulpo(gameRect);
75             break;
76     }
77     return enemies;
78 }
79
```

El switch ordena los distintos niveles creados, como hemos cambiado la constante, podemos añadir hasta 8 niveles diferentes que se repetirán.

```

190     return enemies;
191 }
192 @ 1 usage  INIGO445
193 public static List<AEnemy> crearEnemigosNivelCemboLevel(Rect gameRect)
194 {
195     List<AEnemy> enemies = new ArrayList<>();
196     enemies.add(createEnemyShip(E_ESCUDO, PINGU, gameRect, 0, 0, vx, vy: 0, E_SHOT_GUN, vidas: 5));
197     enemies.add(createEnemyShip(E_ESCUDO, PINGU, gameRect, 2, 0, vx, vy: 0, E_SHOT_GUN, vidas: 5));
198     enemies.add(createEnemyShip(E_ESCUDO, PINGU, gameRect, 4, 0, vx, vy: 0, E_SHOT_GUN, vidas: 5));
199     enemies.add(createEnemyShip(E_ESCUDO, PINGU, gameRect, 6, 0, vx, vy: 0, E_SHOT_GUN, vidas: 5));
200     enemies.add(createEnemyShip(E_TELETRANSPORTE, AMONUS, gameRect, 1, 0, vx, vy: 0, E_SHOT_GUN, vidas: 2));
201     enemies.add(createEnemyShip(E_TELETRANSPORTE, AMONUS, gameRect, 3, 0, vx, vy: 0, E_SHOT_GUN, vidas: 2));
202     enemies.add(createEnemyShip(E_TELETRANSPORTE, AMONUS, gameRect, 5, 0, vx, vy: 0, E_SHOT_GUN, vidas: 2));
203     enemies.add(createEnemyShip(E_TELETRANSPORTE, AMONUS, gameRect, 7, 0, vx, vy: 0, E_SHOT_GUN, vidas: 2));
204     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 3, 2, vx, vy: 0, E_SHOT_GUN, vidas: 5));
205     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 6, 2, vx, vy: 0, E_SHOT_GUN, vidas: 5));
206     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 0, 4, vx, vy: 0, E_SHOT_GUN));
207     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 2, 4, vx, vy: 0, E_SHOT_GUN));
208     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 4, 4, vx, vy: 0, E_SHOT_GUN));
209     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 6, 4, vx, vy: 0, E_SHOT_GUN));
210     return enemies;
211 }
212 @ 1 usage  INIGO445
213 public static List<AEnemy> crearEnemigosNivelFiderLevel(Rect gameRect)
214 {
215     List<AEnemy> enemies = new ArrayList<>();
216     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 1, 4, vx: 0, vy: 0, E_SHOT_GUN));
217     enemies.add(createEnemyShip(E_NORMAL, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 7, 4, vx: 0, vy: 0, E_SHOT_GUN));
218     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 0, 0, vx, vy: 0, E_SHOT_GUN));
219     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 2, 0, vx, vy: 0, E_SHOT_GUN));
220     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 4, 0, vx, vy: 0, E_SHOT_GUN));
221     enemies.add(createEnemyShip(E_NORMAL, CAT, gameRect, 6, 0, vx, vy: 0, E_SHOT_GUN));
222     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 2, 3, vx, vy, E_SHOT_NOTHING));
223     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 3, 2, -vx, vy, E_SHOT_NOTHING));
224     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 4, 2, vx, vy, E_SHOT_NOTHING));
225     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 5, 3, -vx, vy, E_SHOT_NOTHING));
226     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 2, 4, vx, vy, E_SHOT_NOTHING));
227     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 3, 5, -vx, vy, E_SHOT_NOTHING));
228     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 4, 5, vx, vy, E_SHOT_NOTHING));
229     enemies.add(createEnemyShip(E_DIAGONAL, AMONUS, gameRect, 5, 4, -vx, vy, E_SHOT_NOTHING));

```

```

233 @ public static List<AEnemy> crearEnemigosNivelFrenesi(Rect gameRect)
234 {
235     List<AEnemy> enemies = new ArrayList<>();
236     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 0, 0, vx, vy, E_SHOT_GUN, vidas: 5));
237     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 2, 0, vx, vy, E_SHOT_GUN, vidas: 5));
238     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 4, 0, vx, vy, E_SHOT_GUN, vidas: 5));
239     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 6, 0, vx, vy, E_SHOT_GUN, vidas: 5));
240     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 0, 1, vx, vy, E_SHOT_GUN, vidas: 5));
241     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 2, 1, vx, vy, E_SHOT_GUN, vidas: 5));
242     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 4, 1, vx, vy, E_SHOT_GUN, vidas: 5));
243     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 6, 1, vx, vy, E_SHOT_GUN, vidas: 5));
244     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 0, 2, vx, vy, E_SHOT_GUN, vidas: 5));
245     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 2, 2, vx, vy, E_SHOT_GUN, vidas: 5));
246     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 4, 2, vx, vy, E_SHOT_GUN, vidas: 5));
247     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 6, 2, vx, vy, E_SHOT_GUN, vidas: 5));
248     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 0, 3, vx, vy, E_SHOT_GUN, vidas: 5));
249     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 2, 3, vx, vy, E_SHOT_GUN, vidas: 5));
250     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 4, 3, vx, vy, E_SHOT_GUN, vidas: 5));
251     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 6, 3, vx, vy, E_SHOT_GUN, vidas: 5));
252     return enemies;
253 }
1 usage  ± INIGO445
254 @ public static List<AEnemy> nivelFantasmas(Rect gameRect)
255 {
256     List<AEnemy> enemies = new ArrayList<>();
257     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 0, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
258     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 1, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
259     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 2, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
260     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 3, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
261     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 4, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
262     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 5, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
263     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 6, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
264     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 7, 0, vx, vy: 0, E_FANTASMA, vidas: 10));
265     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 0, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
266     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 1, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
267     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 2, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
268     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 3, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
269     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 4, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
270     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 5, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
271     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 6, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
272     enemies.add(createEnemyShip(E_PACMAN, PACMAN, gameRect, 7, 1, vx, vy: 0, E_FANTASMA, vidas: 10));
273     return enemies;

```

```

275 @ public static List<AEnemy> nivelDificilNAI(Rect gameRect)
276 {
277     List<AEnemy> enemies = new ArrayList<>();
278     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 0, 0, vx, vy, E_SHOT_GUN, vidas: 5));
279     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 2, 0, -vx, vy, E_SHOT_GUN, vidas: 5));
280     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 4, 0, -vx, vy, E_SHOT_GUN, vidas: 5));
281     enemies.add(createEnemyShip(E_RANDOM, PINGU, gameRect, 6, 0, vx, vy, E_SHOT_GUN, vidas: 5));
282     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 0, 1, vx, vy, E_SHOT_GUN, vidas: 5));
283     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 2, 1, -vx, vy, E_SHOT_GUN, vidas: 5));
284     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 4, 1, -vx, vy, E_SHOT_GUN, vidas: 5));
285     enemies.add(createEnemyShip(E_RANDOM, AMONUS, gameRect, 6, 1, vx, vy, E_SHOT_GUN, vidas: 5));
286     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 0, 2, vx, vy, E_SHOT_GUN, vidas: 5));
287     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 2, 2, -vx, vy, E_SHOT_GUN, vidas: 5));
288     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 4, 2, -vx, vy, E_SHOT_GUN, vidas: 5));
289     enemies.add(createEnemyShip(E_RANDOM, CAT, gameRect, 6, 2, vx, vy, E_SHOT_GUN, vidas: 5));
290     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 0, 3, vx, vy, E_SHOT_GUN, vidas: 5));
291     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 2, 3, -vx, vy, E_SHOT_GUN, vidas: 5));
292     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 4, 3, -vx, vy, E_SHOT_GUN, vidas: 5));
293     enemies.add(createEnemyShip(E_RANDOM, ENEMYSHIP_SPRITE_IMAGE_3, gameRect, 6, 3, vx, vy, E_SHOT_GUN, vidas: 5));
294     return enemies;
295 }
296
297 }
298

```

Aquí se crean los distintos niveles.

Ahora creamos las clases específicas para cada enemigo nuevo

Clases Pacman y Fantasma:

```
PACMAN.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 > import ...
4
5
6 public class PACMAN extends EnemyShip implements SoyPacman{
7     1 usage INIGO445
8     > public PACMAN(Rect gameRect, Image img, int N) { super(gameRect, img, N); }
9
10
11     1 usage INIGO445
12     @Override
13     public boolean impact() { return true; }
14
15 }
16
```

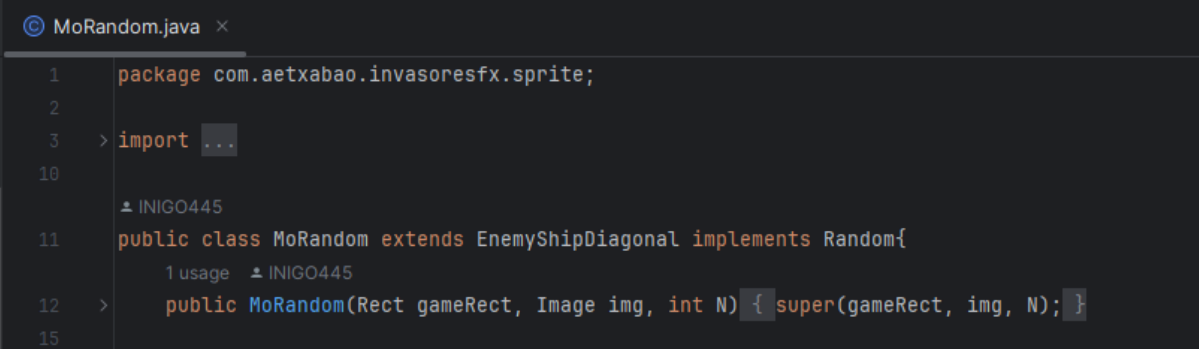
```
SoyPacman.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 1 implementation INIGO445
4 public interface SoyPacman extends IHaveShield{
5     1 usage 1 implementation INIGO445
6     @Override
7     boolean impact();
8 }
9
```

La clase Pacman hereda de una clase padre EnemyShip que además implementa una interface llamada SoyPacman con su atributo impact, que debe ser true para que el enemigo se le retire una vida (mas adelante).

```
Fantasma.java x
1 package com.aetxabao.invasoresfx.sprite.weaponry;
2
3 > import ...
4
5
6
7
8
9
10 public class Fantasma extends AWeapon{
11     1 usage  INIGO445
12     @Override
13     public ArrayList<AShot> shoot(ASprite sprite) {
14         ArrayList<AShot> list = new ArrayList<>();
15         AShot shot = new Cannonball(FANTASMA);
16         shot.setPos(sprite.getRect().centerX(), sprite.getRect().bottom);
17         list.add(shot);
18         return list;
19     }
20 }
```

La clase Fantasma hereda de la clase padre AWeapon. Crea una arma que dispara fantasmas.

Clase Random:



```
1 package com.aetxabao.invasoresfx.sprite;
2
3 > import ...
10
11 public class MoRandom extends EnemyShipDiagonal implements Random{
12     > public MoRandom(Rect gameRect, Image img, int N) { super(gameRect, img, N); }
15
```

La clase Random hereda de la clase padre EnemyShipDiagonal e implementará la interface Random.

```
© MoRandom.java x
28 @Override
29 public void cambio() {
30     int random = (int) (Math.random()*15 + 1);
31     int randomX = (int) (Math.random()*20 - 10);
32     int randomY = (int) (Math.random()*20 - 10);
33     if (getXSpeed() %2 != 0)
34     {
35         int randomA = (int) (Math.random()*2 + 1);
36         if (randomA == 1)
37         {
38             setXSpeed(getXSpeed() -1);
39         }
40         else
41         {
42             setXSpeed(getXSpeed() +1);
43         }
44     }
45     if (getYSpeed() %2 != 0)
46     {
47         int randomB = (int) (Math.random()*2 + 1);
48         if (randomB == 1)
49         {
50             setYSpeed(getYSpeed() -1);
51         }
52         else
53         {
54             setYSpeed(getYSpeed() +1);
55         }
56     }
57     if (getXSpeed() == 0)
58     {
59         setXSpeed(5);
60     }
61     if (getYSpeed() == 0)
62     {
63         setYSpeed(5);
64     }
```

Tenemos estas condiciones para que no de errores de double, en caso de esto ocurriera, el programa recalcula para que se sume uno o restarse 1 para que la división no de con coma.

El método `cambio` se encarga de generar los cambios de velocidad y de posición de los distintos enemigos.

```
© MoRandom.java x
67         case 1:
68             setXSpeed(getXSpeed() * 2);
69             break;
70         case 2:
71             int a = (int) getXSpeed() / 2;
72             setXSpeed(getXSpeed() / 2);
73             break;
74         case 3:
75             setYSpeed(getYSpeed() * 2);
76             break;
77         case 4:
78             setYSpeed(getYSpeed() / 2);
79             break;
80         case 5:
81             setYSpeed(-getYSpeed());
82             break;
83         case 6:
84             setXSpeed(-getXSpeed());
85             break;
86         case 7:
87             setXSpeed(0);
88             break;
89         case 8:
90             setYSpeed(0);
91             break;
92         case 9:
93             setXSpeed(0);
94             setYSpeed(0);
95             break;
96         case 10:
97             setYSpeed(-getYSpeed() * 2);
98             break;
```

```
© MoRandom.java x
99      case 11:
100          setYSpeed(-getYSpeed() / 2);
101          break;
102      case 12:
103          setXSpeed(-getXSpeed() * 2);
104          break;
105      case 13:
106          setXSpeed(-getXSpeed() / 2);
107          break;
108      case 14:
109          setXSpeed(randomX);
110          setYSpeed(randomY);
111          break;
112      case 15:
113          setXSpeed(randomX);
114          break;
115      case 16:
116          setYSpeed(randomY);
117          break;
118      case 17:
119          setXSpeed(randomX * 2);
120          break;
121      case 18:
122          setYSpeed(randomY * 2);
123          break;
124      case 19:
125          setXSpeed(randomX / 2);
126          break;
127      case 20:
128          setYSpeed(randomY / 2);
129          break;
130      default:
131          break;
132    }
133 }
```

El switch entra con un número random del 1 al 20 y hará lo que se describa en caso.

Clase Teletransporte:

```
Teletransporte.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 > import ...
10
11 public class Teletransporte extends EnemyShip implements ICanTeleport{
    6 usages INIGO445
```

La clase Teletransporte hereda de la clase padre EnemyShip e implementará la interface ICanTeleport.

```
92 > public Teletransporte(Rect gameRect, Image img, int N) { super(gameRect, img, N); }
95
1 usage INIGO445
96 @Override
97 > public boolean impact() { return true; }
100
1 usage INIGO445
101 @Override
102 > public void teleport() {
103     int randomX = (int) (Math.random()*440);
104     int randomY = (int) (Math.random()*500);
105     setPos(randomX,randomY);
106 }
```

El método teleport se encarga de generar una posición random para los enemigos para que a la hora de ser impactados generen números random y con esos cambiar la posición de los enemigos.

```
ICanTeleport.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 > public interface ICanTeleport extends IHaveShield{
    1 usage 1 implementation INIGO445
4     @Override
5     boolean impact();
6
    1 usage 1 implementation INIGO445
7     void teleport();
8 }
9
```

Clase Escudo:

```
Escudo.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 import com.aetxabao.invasoresfx.sprite.weaponry.AShot;
4 import com.aetxabao.invasoresfx.sprite.weaponry.AWeapon;
5 import com.aetxabao.invasoresfx.util.Rect;
6 import javafx.scene.canvas.GraphicsContext;
7 import javafx.scene.image.Image;
8
9 import java.util.ArrayList;
10
11 public class Escudo extends EnemyShip implements IHaveShield{
12
13     1 usage  INIGO445
14     public Escudo(Rect gameRect, Image img, int N){ super(gameRect, img, N); }
```

La clase Escudo hereda de la clase padre EnemyShip e implementa la interface IHaveShield. Se encarga de crear enemigos con más de una vida .

```
1 usage  INIGO445
98 @Override
99 public boolean impact() {
100     return true;
101 }
102 }
103
```

```
IHaveShield.java x
1 package com.aetxabao.invasoresfx.sprite;
2
3 7 implementations  INIGO445
4 public interface IHaveShield {
5     1 usage  7 implementations  INIGO445
6     boolean impact();
7 }
8
```


Clase ASprite:

```
ASprite.java x
2 usages
15    protected int vidas;
```

```
89    > public void setVidas(int vidas) { this.vidas = vidas; }
92
7 usages 3 overrides INIGO445
93    > public int getVidas() { return vidas; }
96
2 usages 1 override INIGO445
97    > public void setTeleport(boolean teleport) { this.teleport = teleport; }
1 usage 1 override INIGO445
100    > public boolean isTeleport() { return teleport; }
106
```

El constructor de la clase ASprite no varía por lo que se añaden métodos.

Enums

```
EEnemyShot.java ×  
1 package com.aetxabao.invasoresfx.game.enums;  
2  
3 public enum EEnemyShot {  
4     E_SHOT_NOTHING, E_SHOT_GUN, E_FANTASMA  
5 }  
6
```

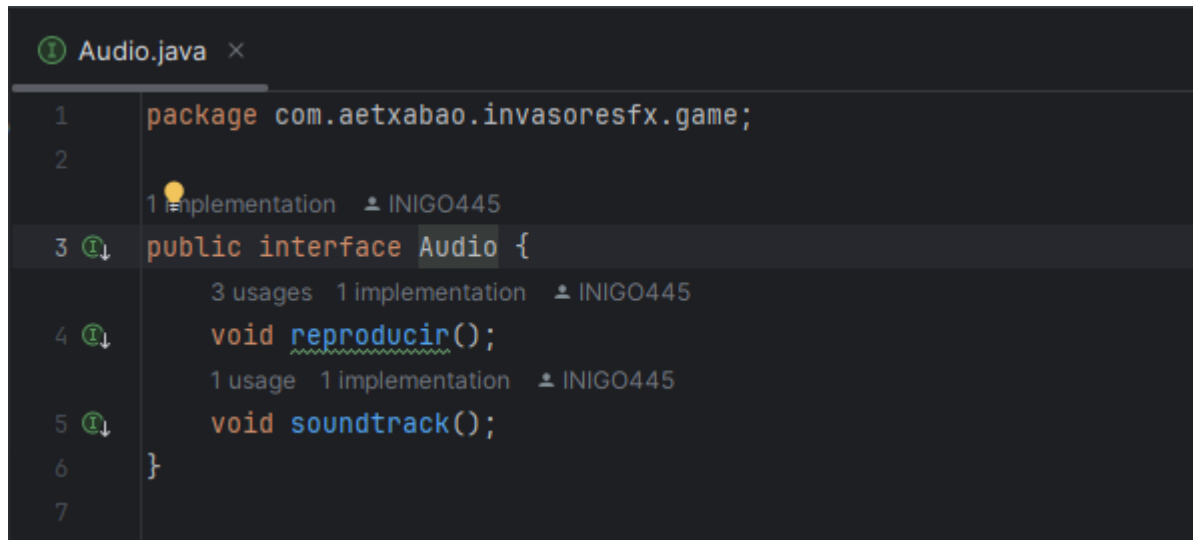
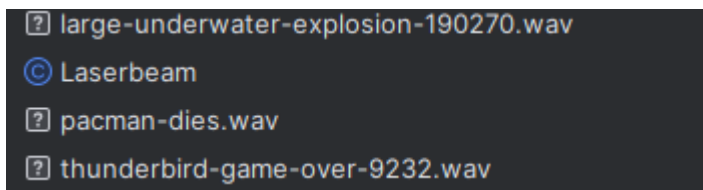
EEnemyShot son los tipos de disparos generados.

```
EEnemyType.java ×  
1 package com.aetxabao.invasoresfx.game.enums;  
2  
3 public enum EEnemyType {  
4     E_NORMAL, E_ONEWAY, E_DIAGONAL, E_SINU, E_ROCKET, E_BARRIER, E_BARRIERDOWN, E_TOWER, E_ESCUDO, E_TELETRANSPORTE, E_RANDOM, E_PACMAN  
5 }
```

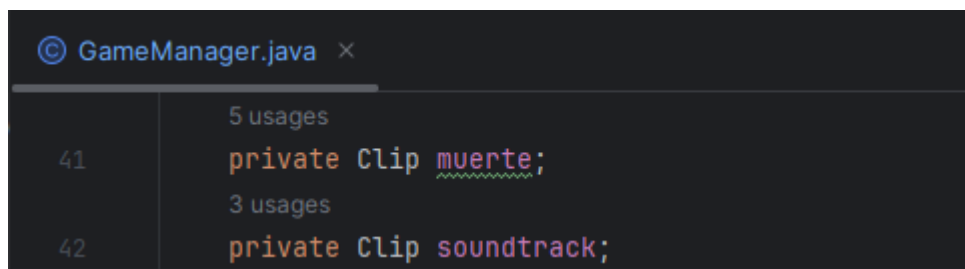
EEnemyType son los tipos de enemigos creados.

!!!Extra!!!

Soundtrack y audios:



La interface Audio servirá para implementar en la clase GameManager los audios para la muerte de los enemigos y del fondo del juego.



```

191         }
192     }else if (sprite instanceof IHaveShield){
193         if (((IHaveShield) sprite).impact()){
194             sprite.setVidas(sprite.getVidas() - 1);
195             if (sprite instanceof ICanTeleport && sprite.getVidas() > -1) {
196                 ((ICanTeleport) sprite).teleport();
197             }
198             if (sprite instanceof Random)
199             {
200                 ((Random) sprite).cambio();
201             }
202         } if (sprite.getVidas() == -1 && !(sprite instanceof SoyPacman)) {
203             cambio(path: "InvasoresFX_Inigo_Eider-main/src/main/java/com/aetxabao/invasoresfx/sprite/weaponry/large-underwater-explosion-190270.wav");
204             reproducir();
205             temps.add(new SpriteTemp(temps, sprite.getRect().centerX(), sprite.getRect().centerY(),
206                                     EXPLOSION_9_SPRITE_IMAGE, cols: 9));
207             itSprite.remove();
208         } else if (sprite.getVidas() == -1 && sprite instanceof SoyPacman)
209         {
210             cambio(path: "InvasoresFX_Inigo_Eider-main/src/main/java/com/aetxabao/invasoresfx/sprite/weaponry/pacman-dies.wav");
211             reproducir();
212             temps.add(new SpriteTemp(temps, sprite.getRect().centerX(), sprite.getRect().centerY(),
213                                     EXPLOSION_PACHAN, cols: 5));
214             itSprite.remove();
215         }
216     }
217 }
218
219 else{
220     cambio(path: "InvasoresFX_Inigo_Eider-main/src/main/java/com/aetxabao/invasoresfx/sprite/weaponry/large-underwater-explosion-190270.wav");
221     temps.add(new SpriteTemp(temps, sprite.getRect().centerX(), sprite.getRect().centerY(),
222                             EXPLOSION_9_SPRITE_IMAGE, cols: 9));
223     reproducir();
224     itSprite.remove();
225 }
226 score += AppConsts.PTS_ENEMYSHIP;
227 itBullet.remove();
228 break;
229 }

```

En el método se llama a la interface de audios mediante el atributo 'muerte' para que en las muertes de los enemigos suene una explosión. Cada enemigo que tenga vidas, al ser golpeado se le restará 1, dependiendo del enemigo hace un sonido u otro al morir.

```

3 usages  INIGO445
54 public void cambio(String path)
55 {
56     try {
57         File sonido = new File(path);
58         AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(sonido);
59         muerte = AudioSystem.getClip();
60         muerte.open(audioInputStream);
61     }
62     catch (Exception e)
63     {
64         e.printStackTrace();
65     }
66 }
67
1 usage  INIGO445
68 @Override
69 public void soundtrack() {
70     try {
71         File sonido = new File(pathname: "InvasoresFX_Inigo_Eider-main/src/main/java/com/aetxabao/invasoresfx/sprite/weaponry/thunderbird-game-over-9232.wav");
72         AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(sonido);
73         soundtrack = AudioSystem.getClip();
74         soundtrack.open(audioInputStream);
75     }
76     catch (Exception e)
77     {
78         e.printStackTrace();
79     }
80     soundtrack.start();
81 }

```

En el método soundtrack se añade el audio de fondo que sonará en bucle empezando desde el principio al iniciar partida.

```
© GameManager.java x
245     }
246     //Actualización de los enemigos
247     for (ASprite enemy : enemies) {
248         enemy.update();
249         if (enemy.getY() <= 0)
250         {
251             enemy.setXSpeed(5);
252             enemy.setYSpeed(3);
253         }
254     }
```

El for se encarga de evitar que los enemigos salgan del mapa, rebotando en los bordes, para evitar cualquier tipo de bug.