



# **UNIVERSIDAD GENERAL SARMIENTO**

**PROGRAMACION 1 - TRABAJO PRACTICO IER SEMESTRE 2025**

**“EL CAMINO DE GONDOLF”**

**GRUPO 7**

**INTEGRANTES:**

**RAMIRO VEGA - 46736458**

**MAXIMO PAREDES - 46215082**

**DIRECCIONES DE MAIL:**

**ramirovegga05@gmail.com**

**maximoparedes12@gmail.com**

# **INTRODUCCION:**

**ESTE TRABAJO CONSISTE EN DESARROLLAR UN VIDEOJUEGO, DONDE EL JUGADOR TOMA EL CONTROL DE UN MAGO LLAMADO GONDOLF EL CUAL ESTA EN BUSCA DE UNA PIEZA DE JOYERIA CON FORMA CIRCULAR Y PROPIEDADES MAGICAS EN LO PROFUNDO DE LAS CATACUMBAS, GONDOLF SE VE ATRAPADO ENTRE ROCAS Y LOS MURCIELAGOS ENVIADOS POR OTRO MAGO LLAMADO SARAMUN.**

**EL JUGADOR DEBE MOVERSE POR EL MAPA EVITANDO CHOCARSE CON LAS ROCAS Y ENFRETANDO A SUS ENEMIGOS CON SUS DOS HECHIZOS DISPONIBLES: EL FUEGO PODEROSO PERO COSTOSO EN MANA, Y EL AGUA... GRATIS PERO NO TAN PODEROSA.**

# DESCRIPCION:

EL ENTORNO DEL VIDEOJUEGO YA ESTABA PROGRAMADO, EL CUAL SE ENCARGABA DE LA PARTE GRAFICA DEL JUEGO, POR LO QUE NUESTRA RESPONSABILIDAD ERA DARLE VIDA AL MISMO, CREANDO LAS CLASES, METODOS, FUNCIONES Y CARGANDOLE LAS IMAGENES.

SE PASA A DESCRIBIR TODAS LAS CLASES, SUS VARIABLES DE INSTANCIA Y METODOS.

## CLASE Juego:

EN ESTA CLASE SE DEFINEN TODOS LOS METODOS Y VARIABLES DE INSTANCIA, TAMBIEN SE DIBUJAN TODOS LOS OBJETOS, POR LO QUE ES LA CLASE CON MAS RESPONSABILIDADES YA QUE SE ENCARGA DE REUNIR EL RESTO DE CLASES Y DARLES UNA FUNCIONALIDAD.

**public class** Juego **extends** InterfaceJuego{} (Contiene a toda la interfaz del juego, junto con las variables, metodos y objetos).

**private** Entorno **entorno**; (Esta variable define el objeto entorno).

**private boolean** **juegoTerminado**; (Esta variable de instancia determina cuando el juego termina en “has ganado” y “has perdido”).

**private boolean** **poderDeFuegoActivo**; (Esta variable determina si el poder de fuego esta activo “True” o inactivo “False”).

**private** Poderes **poderDeFuegoActual**; (Contiene los valores y area de efecto del poder fuego).

**private boolean** **poderDeAguaListoParaLanzar**; (Esta variable determina si el poder de agua esta activo “True” o inactivo “False”).

**private** Poderes **poderDeAguaActual**; (Contiene los valores y area de efecto del poder agua).

**private int** **enemigosAsesinados**; (Guarda el conteo de enemigos asesinados).

**private void** reiniciarJuego() (Este metodo reinicia el juego una vez que el jugador pierde o gana).

**private boolean** colisionMagoEnemigo() (Este metodo crea la colision entre el mago y los murcielagos).

**private boolean** hayColision() (Este metodo verifica la colision del mago con las rocas).

**public void** movimiento() (Este metodo es el que controla los movimientos del mago y utiliza el metodo hayColision() para detener al mago cuando se encuentre con una roca, segun el movimiento que haga el jugador con WASD, utiliza una imagen para cada caso (lo dibuja)).

**public void** moverEnemigos() (Este metodo controla el movimiento de los enemigos; la aceleracion de los mismos, si chocan con el entorno rebotan, y tambien dibuja a los mismos).

**public void** tick() (Este metodo utiliza el rango de tiempo en ticks de 300 milseg, cada ese rango de tiempo maneja a todos los metodos del juego; dibuja los objetos en el juego asi como tambien el fondo, y las acciones; enemigos que siguen al mago, botones que se presionan, el control de vida del mago y su energia, tambien controla las colisiones y los poderes que lanza el mago).

## CLASE BarraLateral:

EN ESTA CLASE SE DEFINE LA BARRA LATERAL DEL JUEGO, QUE SE ENCARGA DE SEPARAR EL MAPA DE RECORRIDO CON LOS HECHIZOS Y BARRAS DE VIDA Y ENERGIA DEL MAGO JUNTO CON LOS BOTONES Y EL CONTADOR DE ENEMIGOS.

**public class** BarraLateral{} (Define la clase BarraLateral y se le otorga un punto x, y un punto y, y una imagen).

**public** BarraLateral() (Crea el objeto BarraLateral que recibe una cordenada x, y, y se le carga la imagen).

**public void** dibujar()(Este metodo dibuja la barra lateral en el entorno, en el punto x,y elegido, se le otorga una medida y un grado de inclinacion).

**public void** dibujarContadorEnemigos() (Este metodo dibuja el contador de enemigos junto con la barra lateral).

## CLASE Mago:

EN ESTA CLASE SE DEFINE AL MAGO GONDOLF JUNTO CON SU VIDA Y ENERGIA (MANA).

**public class** Mago{} (Define la clase mago y le otorga una coordenada x,y imagenes para cada movimiento, un ancho y alto, tambien las barras de vida y energia,)

**public int** vida; (Otorga la variable vida).

**public final int** VIDA\_MAXIMA; (Define la cantidad de vida inicial del mago).

**public int** energia; (Otorga la variable energia).

**public final int** ENERGIA\_MAXIMA; (Define la cantidad de energia con la que comienza el mago).

**public** Mago() (Crea el objeto mago con todas las características ya definidas).

**public void** recibirDaño() (Controla el desgaste de la vida del mago hasta 0).

**public boolean** estaVivo() (Controla que el mago este vivo si su vida es mayor a 0).

**public void** gastarEnergia() (Controla el desgaste de la energia del mago hasta 0)

**public boolean** tieneSuficienteEnergia() (Controla si el mago tiene suficiente energia para lanzar el hechizo fuego).

**public void** dibujarBarraDeVida() (Dibuja un rectangulo para la barra de vida)

**public void** dibujarBarraDeEnergia() (Dibuja un rectangulo para la barra de energia).

## CLASE Roca:

EN ESTA CLASE SE DEFINEN A LAS ROCAS QUE SE ATRAVIESAN EN EL CAMINO DEL MAGO GONDOLF.

**public class** Roca{} (Define a la clase roca, le otorga una coordenada x,y, un ancho y un alto y una imagen).

**public** Roca() (Crea el objeto Roca que recibe una coordenda x,y y una imagen).

**public void** dibujar() (Dibuja la imagen de la roca en el entorno).

**public boolean** colisionaCon() (Crea el area de colision del objeto roca).

## CLASE Enemigos:

EN ESTA CLASE SE DEFINEN A LOS ENEMIGOS ENVIADOS POR EL MAGO SARAMUN QUE ATACARAN AL MAGO GONDOLF.

**public class** Enemigo{} (Crea la clase Enemigo y le brinda variables x,y, velocidad, angulo, radio, y carga la imagen murcielago)

**public** Enemigo () (Crea el objeto Enemigo el cual recibe las variables ya mencionadas).



## CLASE Poderes:

EN ESTA CLASE SE CREAN LOS PODERES (AGUA Y FUEGO) QUE UTILIZARA EL MAGO GONDOLF PARA DERROTAR A SUS ENEMIGOS.

**public class** Poderes{} (Crea la clase poderes, se le otorga una coordenada x e y, un radio, un string que determina el tipo de poder, el tiempo de creacion y el tiempo de visibilidad, se le otorga las imagenes del poder agua y fuego).

**public** Poderes() (Se crea el objeto poderes y se le cargan las variables ya mencionadas).

**public void** dibujarEfecto() (Este metodo dibuja el efecto de los poderes (agua y fuego) en el entorno y hace que sea visible para el jugador utilizando la variable de visibilidad).

**public int** aplicarEfecto() (Este metodo aplica el efecto de los poderes a los enemigos en el area de uso del poder por el area que ocupa el poder usado, los enemigos mueren (null) y se crean nuevos segun el conteo de la cantidad).

**private** Enemigo crearNuevoEnemigo() (Este metodo crea nuevos enemigos que salen de un lugar al azar en los bordes del entorno, luego de asesinados)

**public boolean** yaTermino() (Este metodo controla el tiempo de duracion de los efectos de los poderes luego de lanzados).

## CLASE Boton:

EN ESTA CLASE SE DEFINEN LOS BOTONES QUE SE UTILIZARAN EN EL JUEGO, CON LOS CUALES EL USUARIO ACTIVARA LOS PODERES.

**public class** Boton{} (Se define la clase boton la cual recibe variables x,y ancho y alto, y la imagen “fuego” y “agua” que serian los botones diseñados).

**public** Boton() (Se crea el objeto Boton que recibe las variables ya mencionadas).

**public void** dibujar() (Dibuja al boton de Fuego).

**public void** dibujar1() (Dibuja al boton de Agua).

**public boolean** estaPresionado() (Este metodo verifica si el boton esta presionado, retorna “True” o “False” si se hizo click dentro del area de algun boton).

# **ALGUNOS PROBLEMAS SOLUCIONADOS:**

**\*ALGUNAS RESPONSABILIDADES DE JUEGO EN LAS CLASES DE MAGO, ROCAS Y ENEMIGOS.**

**\*ERRORES DE GENERACION DE ROCAS DETRAS DE LA BARRA.**

**\*PALABRAS FUEGO Y AGUA POR FUERA DE LOS BOTONES.**

**\*IMAGEN DE BARRA NO ENCAJABA CON EL ESTILO DEL JUEGO, Y APARECIA GIGANTE E ICLINADA.**

## **IMAGENES:**

**CLASE JUEGO:**

```
Juego.java X Boton.java Enemigo.java
1 package juego;
2
3 import java.awt.Color;
4
5
6
7
8
9 public class Juego extends InterfaceJuego
10 {
11
12     private Entorno entorno;
13
14     Mago mago;
15     Enemigo[] enemigo;
16     Image imgFondo;
17     Image imgNormal;
18     Image imgMurcielago;
19     Image imgRoca;
20     double anguloFondo;
21     Roca[] rocas = new Roca[5];
22     BarraLateral barra;
23     Boton boton; // Botón para el poder de fuego
24     Boton boton1; // Botón para el poder de agua
25     private boolean juegoTerminado;
26     private boolean poderDeFuegoActivo;
27     private Poderes poderDeFuegoActual;
28     private boolean poderDeAgualistoParaLanzar;
29     private Poderes poderDeAguaActual;
30     private int enemigosAsesinados; // Contador de enemigos asesinados
31
32     Juego()
33     {
34         // Inicializa el objeto entorno
35         this.entorno = new Entorno(this, "Proyecto para TP", 800, 600);
36         imgNormal = Herramientas.cargarImagen("mago base.png");
37         imgMurcielago = Herramientas.cargarImagen("murcielago.png");
38         anguloFondo = 0;
39         this.juegoTerminado = false;
40         this.poderDeFuegoActual = null;
41         this.poderDeFuegoActivo = false;
42         this.poderDeAgualistoParaLanzar = false;
43         this.poderDeAguaActual = null;
44         this.enemigosAsesinados = 0; // Inicializa el contador
45
46         mago = new Mago(400, 300);
47         enemigo = new Enemigo[10]; // El número máximo de enemigos es 10
48         imgFondo = Herramientas.cargarImagen("suelo.png");
49         barra = new BarraLateral(895, 300);
50         boton = new Boton(752, 330, 90, 35);
51         boton1 = new Boton(752, 390, 90, 35);
52
53         for (int i = 0; i < enemigo.length; i++) {
54             double x;
55             double y;
56             int radio = 13;
57             int bordeDeAparicion = (int) (Math.random() * 4);
58             if (bordeDeAparicion == 0) {
59                 x = Math.random() * entorno.ancho();
60                 y = -radio;
61             } else if (bordeDeAparicion == 1) {
62                 x = Math.random() * entorno.ancho();
63                 y = entorno.alto() + radio;
64             } else if (bordeDeAparicion == 2) {
65                 x = -radio;
66                 y = Math.random() * entorno.alto();
67             } else if (bordeDeAparicion == 3) {
68                 x = entorno.ancho() + radio;
69                 y = Math.random() * entorno.alto();
70             } else {
71                 x = 0; y = 0;
72             }
73         }
74     }
75 }
```

```
Juego.java X Boton.java Enemigo.java
71         x = 0; y = 0;
72     }
73
74     double velocidad = 2 + Math.random();
75     double angulo = Math.random() * 2 * Math.PI;
76
77     enemigo[i] = new Enemigo(x, y, velocidad, angulo, radio);
78 }
79 for (int i = 0; i < rocas.length; i++) {
80     double x = Math.random() * 685;
81     double y = Math.random() * 600;
82     rocas[i] = new Roca(x, y);
83 }
84 }
85
86
87 this.entorno.iniciar();
88 //inicia el juego
89 }
90
91 private void reiniciarJuego() {
92     mago = new Mago(400, 300); // Esto ya inicializa la vida y energía en la clase Mago
93     enemigo = new Enemigo[10]; // El número máximo de enemigos es 10
94     for (int a = 0; a < enemigo.length; a++) {
95         double x;
96         double y;
97         int radio = 13;
98         int bordeDeAparicion = (int) (Math.random() * 4);
99
100         if (bordeDeAparicion == 0) {
101             x = Math.random() * entorno.ancho();
102             y = -radio;
103         } else if (bordeDeAparicion == 1) {
104             x = Math.random() * entorno.ancho();
105             y = entorno.alto() + radio;
106         } else if (bordeDeAparicion == 2) {
107             x = -radio;
108             y = Math.random() * entorno.alto();
109         } else {
110             x = entorno.ancho() + radio;
111             y = Math.random() * entorno.alto();
112         }
113         double velocidad = 2 + Math.random();
114         double angulo = Math.random() * 2 * Math.PI;
115         enemigo[a] = new Enemigo(x, y, velocidad, angulo, radio);
116     }
117     juegoTerminado = false;
118
119     poderDeFuegoActivo = false;
120     poderDeFuegoActual = null;
121     poderDeAguaListoParaLanzar = false;
122     poderDeAguaActual = null;
123     this.enemigosAsesinados = 0; // Reinicia el contador
124 }
125
126 // MÉTODO DE COLISIÓN Mago-Enemigo
127 private boolean colisionMagoEnemigo(Mago mago, Enemigo enemigo) {
128     double closestX = Math.max(mago.x - mago.ancho / 2, Math.min(enemigo.x, mago.x + mago.ancho / 2)); // Corregido: mago.ancho
129     double closestY = Math.max(mago.y - mago.alto / 2, Math.min(enemigo.y, mago.y + mago.alto / 2));
130
131     double distX = enemigo.x - closestX;
132     double distY = enemigo.y - closestY;
133     double distanciaAlCuadrado = (distX * distX) + (distY * distY);
134
135     return distanciaAlCuadrado < (enemigo.radio * enemigo.radio);
136 }
137
138
```

```

139 //COLISION ROCAS Y ENTORNO
140
141 private boolean hayColision(double xNuevo, double yNuevo, int ancho, int alto, Roca[] rocas) {
142     double mitadW = ancho / 2.0;
143     double mitadH = alto / 2.0;
144     if (xNuevo < mitadW || xNuevo > 700 - mitadW || yNuevo < mitadH || yNuevo > entorno.alto() - mitadH) {
145         return true;
146     }
147     for (Roca roca : rocas) {
148         if (roca.colisionaCon(xNuevo - ancho / 2, yNuevo - alto / 2, ancho, alto)) {
149             return true;
150         }
151     }
152     return false;
153 }
154
155 //MOVIMIENTO MAGO
156
157 public void movimiento(Entorno entorno, Roca[] rocas) {
158     double paso = 4;
159     double nuevaX = mago.x;
160     double nuevaY = mago.y;
161     double MitadW = mago.ancho / 2.0;
162     double MitadH = mago.alto / 2.0;
163     if (entorno.estaPresionada('w')) {
164         nuevaY -= paso;
165         if (!hayColision(nuevaX, nuevaY, mago.ancho, mago.alto, rocas)) {
166             mago.y = nuevaY;
167             nuevaX = Math.max(MitadW, Math.min(nuevaX, entorno.ancho() - MitadW));
168             nuevaY = Math.max(MitadH, Math.min(nuevaY, entorno.alto() - MitadH));
169             //entorno.dibujarRectangulo(nuevaX, nuevaY, 30, 40, 0, Color.MAGENTA);
170         }
171         entorno.dibujarImagen(mago.imgEspalda, mago.x, mago.y, anguloFondo, 0.08);
172     }
173     else if (entorno.estaPresionada('s')) {
174         nuevaY += paso;
175         if (!hayColision(nuevaX, nuevaY, mago.ancho, mago.alto, rocas)) {
176             mago.y = nuevaY;
177             nuevaX = Math.max(MitadW, Math.min(nuevaX, entorno.ancho() - MitadW));
178             nuevaY = Math.max(MitadH, Math.min(nuevaY, entorno.alto() - MitadH));
179             //entorno.dibujarRectangulo(nuevaX, nuevaY, 30, 40, 0, Color.MAGENTA);
180         }
181         entorno.dibujarImagen(mago.imgFrente, mago.x, mago.y, anguloFondo, 0.08);
182     }
183     else if (entorno.estaPresionada('a')) {
184         nuevaX -= paso;
185         if (!hayColision(nuevaX, nuevaY, mago.ancho, mago.alto, rocas)) {
186             mago.x = nuevaX;
187             nuevaX = Math.max(MitadW, Math.min(nuevaX, entorno.ancho() - MitadW));
188             nuevaY = Math.max(MitadH, Math.min(nuevaY, entorno.alto() - MitadH));
189             //entorno.dibujarRectangulo(nuevaX, nuevaY, 30, 40, 0, Color.MAGENTA);
190         }
191         entorno.dibujarImagen(mago.imgDerecha, mago.x, mago.y, anguloFondo, 0.08);
192     }
193     else if (entorno.estaPresionada('d')) {
194         nuevaX += paso;
195         if (!hayColision(nuevaX, nuevaY, mago.ancho, mago.alto, rocas)) {
196             mago.x = nuevaX;
197             nuevaX = Math.max(MitadW, Math.min(nuevaX, entorno.ancho() - MitadW));
198             nuevaY = Math.max(MitadH, Math.min(nuevaY, entorno.alto() - MitadH));
199             //entorno.dibujarRectangulo(nuevaX, nuevaY, 30, 40, 0, Color.MAGENTA);
200         }
201         entorno.dibujarImagen(mago.imgIzquierda, mago.x, mago.y, anguloFondo, 0.08);
202     }
203     else {
204         entorno.dibujarImagen(imgNormal, mago.x, mago.y, anguloFondo, 0.08);
205         //entorno.dibujarRectangulo(nuevaX, nuevaY, 50, 75, 0, Color.MAGENTA);
206     }
}

```

```

207     }
208
209     //MOVIMIENTO ENEMIGO
210
211     public void moverEnemigos(Enemigo i) {
212         i.y += i.velocidad * Math.sin(i.angulo);
213         i.x += i.velocidad * Math.cos(i.angulo);
214     }
215
216     public boolean chocasteCon(Entorno e) {
217         for (Enemigo i : enemigo) {
218             if (i.x <= i.radio || i.y <= i.radio || i.x >= e.anch() - i.radio || i.y >= e.alto() - i.radio) {
219                 return true;
220             }
221         }
222         return false;
223     }
224
225     public void rebotar(Enemigo i) {
226         i.angulo += Math.PI/2;
227     }
228     public void acelerar(Enemigo i) {
229         i.velocidad += 0.001;
230     }
231
232     public void dibujoEnemigos(Enemigo i) {
233         entorno.dibujarImagen(i.imgMurcielago, i.x, i.y, i.angulo, 0.08); // Dibuja el murciélago
234     }
235     /**
236      * Durante el juego, el método tick() será ejecutado en cada instancia y
237      * por lo tanto es el método más importante de esta clase. Aquí se debe
238      * actualizar el estado interno del juego para simular el paso del tiempo
239      * (ver el enunciado del TP para mayor detalle).
240      */
241     public void tick(){
242         // Condición de victoria o derrota
243         if (!imago.estaVivo()) {
244             juegoTerminado = true;
245         }
246
247         if (this.enemigosAsesinados >= 50) { // Si se eliminaron 50 o más enemigos, ganas
248             juegoTerminado = true;
249         }
250
251         if (juegoTerminado) {
252             entorno.cambiarFont("Arial", 50, Color.RED);
253             if (this.enemigosAsesinados >= 50) {
254                 entorno.escribirTexto("¡HAS GANADO!", entorno.anch() / 2 - 150, entorno.alto() / 2);
255             } else {
256                 entorno.escribirTexto("HAS MUERTO", entorno.anch() / 2 - 150, entorno.alto() / 2);
257             }
258             entorno.cambiarFont("Arial", 20, Color.WHITE);
259             entorno.escribirTexto("Presiona Q para reiniciar", entorno.anch() / 2 - 120, entorno.alto() / 2 + 50);
260
261             if (entorno.estaPresionada('q')) {
262                 reiniciarJuego();
263             }
264             return; // Detiene la ejecución del tick si el juego ha terminado
265         }
266
267         entorno.dibujarImagen(imgFondo, 300, 300, anguloFondo, 1.0);
268         movimiento(entorno, rocas);
269
270         // Bucle for para poder remolazar enemigos
271         for (int i = 0; i < enemigo.length; i++) {
272             Enemigo currentEnemigo = enemigo[i];
273
274             currentEnemigo.angulo = Math.atan2(mago.y - currentEnemigo.y, mago.x - currentEnemigo.x);

```

```

273
274     currentEnemigo.angulo = Math.atan2(mago.y - currentEnemigo.y, mago.x - currentEnemigo.x);
275     moverEnemigos(currentEnemigo);
276     if (chocasteCon(entorno)) {
277         rebotar(currentEnemigo);
278     }
279     dibujarEnemigos(currentEnemigo);
280     acelerar(currentEnemigo);
281
282     if (colisionMagoEnemigo(mago, currentEnemigo)) {
283         // Cuando el mago choca, le baja vida.
284         mago.recibirDaño(5); // <-- línea reintroducida para que le baje vida
285
286         // Crea un nuevo enemigo y reemplaza al que colisionó
287         double x;
288         double y;
289         int radio = 13;
290         int bordeDeAparicion = (int) (Math.random() * 4);
291
292         if (bordeDeAparicion == 0) {
293             x = Math.random() * entorno.ancho();
294             y = -radio;
295         } else if (bordeDeAparicion == 1) {
296             x = Math.random() * entorno.ancho();
297             y = entorno.alto() + radio;
298         } else if (bordeDeAparicion == 2) {
299             x = -radio;
300             y = Math.random() * entorno.alto();
301         } else {
302             x = entorno.ancho() + radio;
303             y = Math.random() * entorno.alto();
304         }
305         double velocidad = 2 + Math.random();
306         double angulo = Math.random() * 2 * Math.PI;
307         enemigo[i] = new Enemigo(x, y, velocidad, angulo, radio);
308     }
309 }
310
311 for (Roca roca : rocas) {
312     roca.dibujar(entorno);
313 }
314
315 //Dibujo y gestión del poder fuego
316 if (this.poderDeFuegoActual != null) {
317     this.poderDeFuegoActual.dibujarEfecto(entorno);
318     if (this.poderDeFuegoActual.yaTermino()) {
319         this.poderDeFuegoActual = null;
320     }
321 }
322
323
324 //Dibujo y gestión del Poder Agua
325 if (this.poderDeAguaActual != null) {
326     this.poderDeAguaActual.dibujarEfecto(entorno);
327     if (this.poderDeAguaActual.yaTermino()) {
328         this.poderDeAguaActual = null;
329     }
330 }
331 barra.dibujar(entorno);
332 boton.dibujar(entorno);
333 boton1.dibujar1(entorno);
334 mago.dibujarBarraDeVida(entorno);
335 mago.dibujarBarraDeEnergia(entorno);
336
337 // Nueva la responsabilidad de dibujar el contador a BarraLateral
338 // entorno.cambiarFont("Arial", 20, Color.WHITE);
339 // entorno.escribirTexto("Enemigos: " + this.enemigosAsesinados + "/50", 10, 30);
340

```



```

340
341
342 // NUEVO: Pasa el contador a la barra lateral para que lo dibuje
343 barra.dibujarContadorEnemigos(this.enemigosAsesinados, entorno);
344
345
346 // lógica de clics y poderes
347 if(entorno.sePresionoBoton(entorno.BOTON_IZQUIERDO)) {
348     double mx = entorno.mouseX();
349     double my = entorno.mouseY();
350     if (boton.estaPresionado(mx, my)) {
351         this.poderDeFuegoActivo = true;
352         System.out.println("Poder de Fuego activado, haz clic en pantalla para lanzar.");
353         this.poderDeAguaListoParaLanzar = false;
354     } else if (boton1.estaPresionado(mx, my)) {
355         this.poderDeAguaListoParaLanzar = true;
356         System.out.println("Poder de Agua activado, haz clic en pantalla para lanzar.");
357         this.poderDeFuegoActivo = false;
358     }
359     // lógica para lanzar el poder si está activo y se hace clic en la pantalla
360     else if (this.poderDeFuegoActivo) {
361         int costoPoderFuego = 20;
362         if (mago.tieneSuficienteEnergia(costoPoderFuego)) {
363             double clickX = entorno.mouseX();
364             double clickY = entorno.mouseY();
365             this.poderDeFuegoActual = new Poderes(clickX, clickY, 130, "fuego");
366             // Captura el valor de retorno de enemigos afectados y lo suma al contador principal
367             this.enemigosAsesinados += this.poderDeFuegoActual.aplicarEfecto(enemigo, entorno);
368             mago.gastarEnergia(costoPoderFuego);
369             this.poderDeFuegoActivo = false;
370             System.out.println("¡Poder de fuego lanzado en: (" + clickX + ", " + clickY + ")!");
371         } else {
372             System.out.println("No tienes suficiente energía para lanzar el poder de fuego.");
373             this.poderDeFuegoActivo = false;
374         }
375     }
376     // lógica para lanzar el poder de agua
377     else if (this.poderDeAguaListoParaLanzar) {
378         double clickX = entorno.mouseX();
379         double clickY = entorno.mouseY();
380         this.poderDeAguaActual = new Poderes(clickX, clickY, 80, "agua");
381         // El poder de agua ahora también contribuye al contador de enemigos asesinados
382         this.enemigosAsesinados += this.poderDeAguaActual.aplicarEfecto(enemigo, entorno);
383         this.poderDeAguaListoParaLanzar = false;
384         System.out.println("¡Poder de agua lanzado en: (" + clickX + ", " + clickY + ")!");
385     }
386 }
387
388
389
390
391 @SuppressWarnings("unused")
392 public static void main(String[] args)
393 {
394     Juego juego = new Juego();
395 }
396
397 }

```

## CLASE BARRALATERAL:

```
File Edit Source Refactor Navigate Search Project Run Window Help
Juego.java Boton.java Enemigo.java BarraLateral.java X
1 package juego;
2
3 import java.awt.Image;
4
5
6
7
8 public class BarraLateral {
9     double x;
10    double y;
11    Image imgBarra;
12
13    public BarraLateral(double x, double y) {
14        this.x = x;
15        this.y = y;
16        imgBarra = Herramientas.cargarImagen("barralateral.jpg");
17    }
18
19    public void dibujar(entorno.Entorno entorno) {
20        entorno.dibujarImagen(imgBarra, x, y, 3.141, 2.135);
21    }
22
23
24    public void dibujarContadorEnemigos(int enemigosAsesinados, Entorno entorno) {
25        int rectX = 740;
26        int rectY = 20;
27        int rectAncho = 100;
28        int rectAlto = 20;
29        entorno.dibujarRectangulo(rectX, rectY, rectAncho, rectAlto, 0, Color.GRAY);
30        int textoX = rectX - rectAncho / 2 + 5;
31        int textoY = rectY + rectAlto / 2 + 5;
32        entorno.cambiarFont("Arial", 10, Color.WHITE);
33        entorno.escribirTexto("Enemigos: " + enemigosAsesinados + "/50", textoX, textoY);
34    }
35 }
```

**CLASE MAGO:**

```
1 package juego;
2
3 import java.awt.Image;
4
5
6
7
8 public class Mago {
9
10     double x;
11     double y;
12     Image imgNormal;
13     Image imgDerecha;
14     Image imgIzquierda;
15     Image imgEspalda;
16     Image imgFrente;
17
18     int ancho = 50;
19     int alto = 75;
20
21     public int vida;
22     public final int VIDA_MAXIMA = 100;
23
24     public int energia;
25     public final int ENERGIA_MAXIMA = 100;
26
27     public Mago(int x, int y) {
28         this.x = x;
29         this.y = y;
30         this.vida = VIDA_MAXIMA;
31         this.energia = ENERGIA_MAXIMA;
32         imgNormal = Herramientas.cargarImagen("mago base.png");
33         imgDerecha = Herramientas.cargarImagen("magoderecha.png");
34         imgIzquierda = Herramientas.cargarImagen("magoizquierda.png");
35         imgEspalda = Herramientas.cargarImagen("magoespalda.png");
36         imgFrente = Herramientas.cargarImagen("magofrente.png");
37     }
38
39     public void recibirDaño(int cantidad) {
40         this.vida -= cantidad;
41         if (this.vida < 0) {
42             this.vida = 0;
43         }
44     }
45
46     public boolean estaVivo() {
47         return this.vida > 0;
48     }
49
50
51     public void gastarEnergia(int cantidad) {
52         this.energia -= cantidad;
53         if (this.energia < 0) {
54             this.energia = 0;
55         }
56     }
57
58     public boolean tieneSuficienteEnergia(int costo) {
59         return this.energia >= costo;
60     }
61
62     public void dibujarBarraDeVida(Entorno e) {
63         int anchoBarra = 80;
64         int altoBarra = 15;
65         int margenX = 715;
66         int margenY = 200;
67
68         // Fondo de la barra de vida
69         e.dibujarRectangulo(margenX + anchoBarra / 2.0, margenY + altoBarra / 2.0, anchoBarra, altoBarra, 0, Color.DARK_GRAY);
70
71         // Barra de vida actual (verde)
```

```

70
71 // Barra de vida actual (verde)
72 double porcentajeVida = (double)this.vida / this.VIDA_MAXIMA;
73 int anchoVidaActual = (int)(anchoBarra * porcentajeVida);
74
75 e.dibujarRectangulo(margenX + anchoVidaActual / 2.0, margenY + altoBarra / 2.0, anchoVidaActual, altoBarra, 0, Color.GREEN);
76 e.cambiarFont("Arial", 10, Color.WHITE);
77 e.escribirTexto("Vida: " + this.vida, margenX + 5, margenY + altoBarra / 2 + 5);
78 }
79
80 public void dibujarBarraDeEnergia(Entorno e) {
81     int anchoBarra = 80;
82     int altoBarra = 15;
83     int margenX = 715;
84     int margenY = 180;
85
86     e.dibujarRectangulo(margenX + anchoBarra / 2.0, margenY + altoBarra / 2.0, anchoBarra, altoBarra, 0, Color.DARK_GRAY);
87     double porcentajeEnergia = (double)this.energia / this.ENERGIA_MAXIMA;
88     int anchoEnergiaActual = (int)(anchoBarra * porcentajeEnergia);
89     e.dibujarRectangulo(margenX + anchoEnergiaActual / 2.0, margenY + altoBarra / 2.0, anchoEnergiaActual, altoBarra, 0, Color.blue);
90     e.cambiarFont("Arial", 10, Color.WHITE);
91     e.escribirTexto("Energia: " + this.energia, margenX + 5, margenY + altoBarra / 2 + 5);
92 }
93 }

```

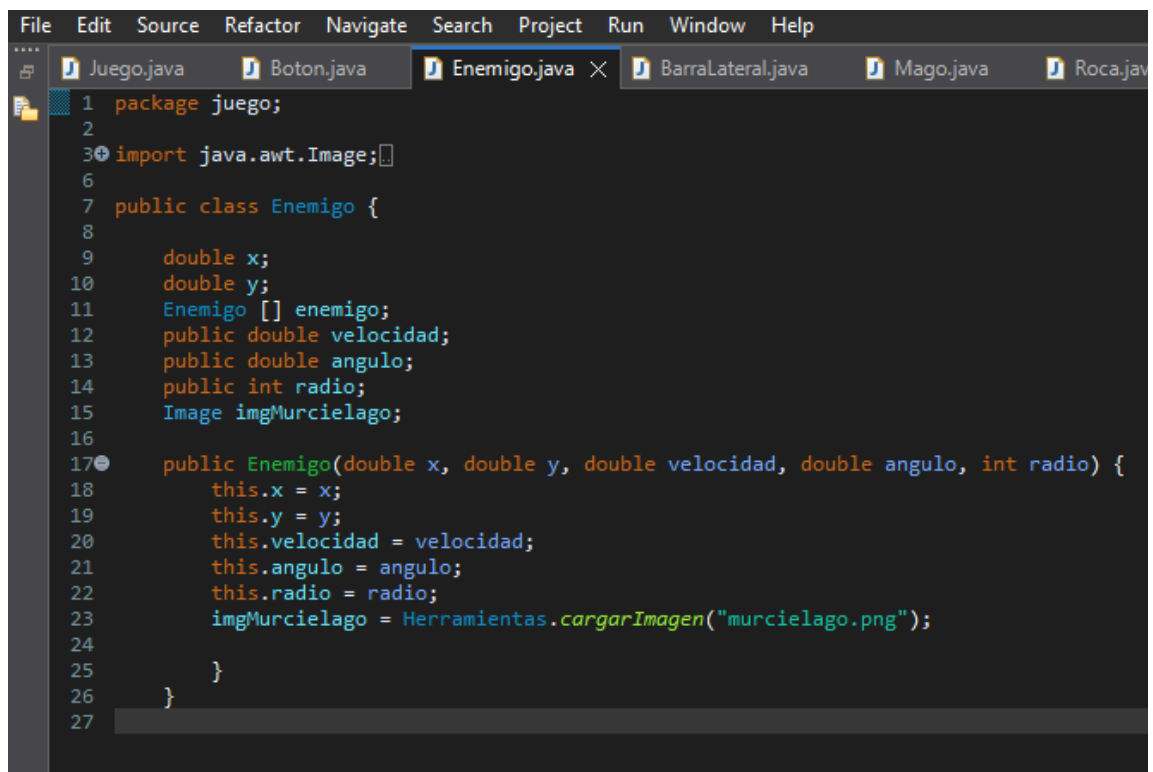
## CLASE ROCA:

```

File Edit Source Refactor Navigate Search Project Run Window Help
Juego.java Boton.java Enemigo.java BarraLateral.java Mago.java Roca.java X
1 package juego;
2
3
4 import java.awt.Image;
5
6
7 public class Roca {
8     double x;
9     double y;
10    int ancho = 30;
11    int alto = 35;
12    Image imgRoca;
13
14    public Roca(double x, double y) {
15        this.x = x;
16        this.y = y;
17        imgRoca = Herramientas.cargarImagen("roca.png");
18    }
19
20    public void dibujar(Entorno entorno) {
21        entorno.dibujarImagen(imgRoca, x, y, 0, 0.88);
22        //entorno.dibujarRectangulo(x, y, 35, 38, 0, Color.green);
23    }
24
25    public boolean colisionaCon(double mx, double my, int mancho, int malto) {
26        double rx = this.x - this.ancho / 2;
27        double ry = this.y - this.alto / 2;
28
29        return mx < rx + this.ancho &&
30               mx + mancho > rx &&
31               my < ry + this.alto &&
32               my + malto > ry;
33    }
34 }
35
36
37
38

```

## CLASE ENEMIGOS:



```
File Edit Source Refactor Navigate Search Project Run Window Help
Juego.java Boton.java Enemigo.java BarraLateral.java Mago.java Roca.java
1 package juego;
2
3 import java.awt.Image;
4
5
6
7 public class Enemigo {
8
9     double x;
10    double y;
11    Enemigo [] enemigo;
12    public double velocidad;
13    public double angulo;
14    public int radio;
15    Image imgMurcielago;
16
17    public Enemigo(double x, double y, double velocidad, double angulo, int radio) {
18        this.x = x;
19        this.y = y;
20        this.velocidad = velocidad;
21        this.angulo = angulo;
22        this.radio = radio;
23        imgMurcielago = Herramientas.cargarImagen("murcielago.png");
24    }
25
26 }
27
```

## CLASE PODERES:

```
****
Juego.java Boton.java Enemigo.java BarraLateral.java Mago.java Roca.java Poderes.java X
1 package juego;
2
3 import java.awt.Color;
4
5
6
7
8
9 public class Poderes {
10     double x;
11     double y;
12     public int radio;
13     String tipoPoder;
14     long tiempoCreacion;
15     final long DURACION_VISIBLE_MS = 300;
16     Image PoderFuego;
17     Image PoderAgua;
18
19     public Poderes(double x, double y, int radio, String tipoPoder) {
20         this.x = x;
21         this.y = y;
22         this.radio = radio;
23         this.tipoPoder = tipoPoder;
24         this.tiempoCreacion = System.currentTimeMillis();
25         PoderFuego = Herramientas.cargarImagen("PoderFuego.png");
26         PoderAgua = Herramientas.cargarImagen("PoderAgua.png");
27     }
28
29     public void dibujarEfecto(Entorno e) {
30         if (System.currentTimeMillis() - this.tiempoCreacion < DURACION_VISIBLE_MS) {
31             if (this.tipoPoder.equals("fuego")) {
32                 e.dibujarImagen(PoderFuego, x, y, 0, 0.15); // Dibuja la imagen del poder de fuego
33             }
34             if (this.tipoPoder.equals("agua")) {
35                 e.dibujarCirculo(this.x, this.y, this.radio, Color.CYAN);
36                 e.dibujarImagen(PoderAgua, x, y, 0, 0.15);
37             }
38         }
39     }
40
41
42     public int aplicarEfecto(Enemigo[] enemigos, Entorno e) {
43         int enemigosAfectados = 0;
44
45         // Lógica para el poder de fuego
46         if (this.tipoPoder.equals("fuego")) {
47             for (int i = 0; i < enemigos.length; i++) {
48                 Enemigo currentEnemigo = enemigos[i];
49                 if (currentEnemigo != null) {
50                     double distanciaX = this.x - currentEnemigo.x;
51                     double distanciaY = this.y - currentEnemigo.y;
52                     double distancia = Math.sqrt(distanciaX * distanciaX + distanciaY * distanciaY);
53                     if (distancia < this.radio + currentEnemigo.radio) {
54                         enemigos[i] = crearNuevoEnemigo(e);
55                         enemigosAfectados++;
56                     }
57                 }
58             }
59         }
60
61         // Lógica para el poder de agua
62         if (this.tipoPoder.equals("agua")) {
63             for (int i = 0; i < enemigos.length; i++) {
64                 Enemigo currentEnemigo = enemigos[i];
65                 if (currentEnemigo != null) {
66                     double distanciaX = this.x - currentEnemigo.x;
67                     double distanciaY = this.y - currentEnemigo.y;
68                     double distancia = Math.sqrt(distanciaX * distanciaX + distanciaY * distanciaY);
69                     if (distancia < this.radio + currentEnemigo.radio) {
70                         enemigos[i] = crearNuevoEnemigo(e);
71                         enemigosAfectados++;
72                     }
73                 }
74             }
75         }
76     }
77 }
```



```
File Edit Source Refactor Navigate Search Project Run Window Help
Juego.java Boton.java Enemigo.java BarraLateral.java Mago.java Roca.java Poderes.java X
40
41
42 public int aplicarEfecto(Enemigo[] enemigos, Entorno e) {
43     int enemigosAfectados = 0;
44
45     // lógica para el poder de fuego
46     if (this.tipoPoder.equals("fuego")) {
47         for (int i = 0; i < enemigos.length; i++) {
48             Enemigo currentEnemigo = enemigos[i];
49             if (currentEnemigo != null) {
50                 double distanciaX = this.x - currentEnemigo.x;
51                 double distanciaY = this.y - currentEnemigo.y;
52                 double distancia = Math.sqrt(distanciaX * distanciaX + distanciaY * distanciaY);
53                 if (distancia < this.radio + currentEnemigo.radio) {
54                     enemigos[i] = crearNuevoEnemigo(e);
55                     enemigosAfectados++;
56                 }
57             }
58         }
59     }
60
61     // lógica para el poder de agua
62     if (this.tipoPoder.equals("agua")) {
63         for (int i = 0; i < enemigos.length; i++) {
64             Enemigo currentEnemigo = enemigos[i];
65             if (currentEnemigo != null) {
66                 double distanciaX = this.x - currentEnemigo.x;
67                 double distanciaY = this.y - currentEnemigo.y;
68                 double distancia = Math.sqrt(distanciaX * distanciaX + distanciaY * distanciaY);
69                 if (distancia < this.radio + currentEnemigo.radio) {
70                     enemigos[i] = crearNuevoEnemigo(e);
71                     enemigosAfectados++;
72                 }
73             }
74         }
75     }
76
77     return enemigosAfectados;
78 }
79
80
81 private Enemigo crearNuevoEnemigo(Entorno e) {
82     double xNuevo, yNuevo;
83     int radioEnemigo = 13;
84     int bordeDeAparicion = (int) (Math.random() * 4);
85
86     if (bordeDeAparicion == 0) {
87         xNuevo = Math.random() * e.ancho();
88         yNuevo = -radioEnemigo;
89     } else if (bordeDeAparicion == 1) {
90         xNuevo = Math.random() * e.ancho();
91         yNuevo = e.alto() + radioEnemigo;
92     } else if (bordeDeAparicion == 2) {
93         xNuevo = -radioEnemigo;
94         yNuevo = Math.random() * e.alto();
95     } else { // Derecha
96         xNuevo = e.ancho() + radioEnemigo;
97         yNuevo = Math.random() * e.alto();
98     }
99     double velocidadNueva = 2 + Math.random();
100     double anguloNuevo = Math.random() * 2 * Math.PI;
101     return new Enemigo(xNuevo, yNuevo, velocidadNueva, anguloNuevo, radioEnemigo);
102 }
103
104 public boolean yaTermino() {
105     return System.currentTimeMillis() - this.tiempoCreacion >= DURACION_VISIBLE_MS;
106 }
107 }
```

CLASE BOTON:

```
File Edit Source Refactor Navigate Search Project Run Window Help
Juego.java Boton.java X Enemigo.java BarraLateral.java Mago.java Roca.java Poderes.java

1 package juego;
2
3 import entorno.Entorno;
4
5
6 public class Boton {
7     double x;
8     double y;
9     int ancho;
10    int alto;
11    Image fuego;
12    Image agua;
13    public Boton(int x, int y, int ancho, int alto) {
14        this.x = x;
15        this.y = y;
16        this.ancho = ancho;
17        this.alto = alto;
18        fuego = Herramientas.cargarImagen("fuego.png");
19        agua = Herramientas.cargarImagen("agua.png");
20    }
21    public void dibujar(Entorno e) {
22        e.dibujarImagen(fuego, x, y, 0, 0.07);
23    }
24    public void dibujar1(Entorno e) {
25        e.dibujarImagen(agua, x, y, 0, 0.07);
26    }
27    public boolean estaPresionado(double px, double py) {
28        return px >= x - ancho / 2 && px <= x + ancho / 2 && py >= y - alto / 2 && py <= y + alto / 2;
29    }
30 }
31
32
33
34
35 }
```



# **CONCLUSION:**

**PARA CONCLUIR ESTE INFORME PODEMOS DECIR QUE EL DESAFIO DE DESARROLLAR ESTE JUEGO EN JAVA NOS BRINDO DE UNA GRAN EXPERIENCIA A LA HORA DE COMPRENDER COMO FUNCIONA ESTE LENGUAJE, APRENDIMOS A DEFINIR LOS OBJETOS, A UTILIZAR LOS METODOS DE FORMA CORRECTA, SI BIEN RECONOCEMOS QUE HEMOS TENIDO PROBLEMAS A LA HORA DE DISTRIBUIR LAS RESPONSABILIDADES A CADA CLASE EN ALGUNOS CASOS, PUDIMOS SOLUCIONAR LA MAYORIA DE ELLOS Y MANTENER DENTRO DE LO POSIBLE... UN CODIGO LIMPIO, LO MAS GRATIFICANTE QUE NOS LLEVAMOS LUEGO DE LA FINALIZACION DEL PROYECTO, ES EL HABER PODIDO CREAR ALGO QUE FUNCIONE Y QUE TAMBIEN SEA DIVERTIDO A LA HORA DE JUGARLO.**

**SIN DUDAS NOS QUEDAMOS CON LAS GANAS DE SEGUIR DESARROLLANDO MAS, QUIZAS LAS CUESTIONES DE TIEMPO, Y EL RESOLVER LOS PROBLEMAS QUE SURGIAN MIENTRAS DESARROLLABAMOS NOS DEJARON CON LAS GANAS...**