TP GRUPAL :D

"Al rescate de los gnomos"
Versión Spooky

Fernandez Facundo

DNI: 45132844

Mail: facundofer2607@gmail.com

Fernandez Facundo

Introducción

El presente trabajo tiene como objetivo el desarrollo de un videojuego, enfocándose en su partido lógico y principalmente en la creación y comunicación entre los diferentes objetos de las clases que componen el programa. Para ello, hacemos uso de un entorno grafico, cumplir con los siguientes requerimientos:

1. Al comenzar el juego, deben mostrarse entre 4 y 6 filas de islas flotantes y la isla con la casa de los gnomos en el sector central superior de la pantalla. Las dos filas superiores de islas flotantes deben ocupar solo un sector de la pantalla (aproximadamente la mitad) y deben estar intercaladas con sectores vacíos. Las islas en las filas inferiores también deben intercalarse con espacios vacíos ocupando todo el ancho de la pantalla. Pep debe
2. estar ubicado sobre una de las islas en la fila inferior. (Ver Figura 1).
3. Mientras este sobre las islas flotantes, Pep puede moverse usando las teclas izquierda y derecha, o las teclas ’a’ y ’d’, y puede saltar con la tecla flecha arriba o ’w’. Si no se presiona ninguna tecla, Pep debe permanecer parado sobre la isla. Si Pep no está sobre una isla flotante o no está saltando, debe caer hasta hacer colisión con una isla o caer al precipicio.
4. Pep tiene el poder del sol en sus manos, lo que le permite arrojar pequeñas bolas de fuego que van a ras del piso para acabar con sus enemigos. Para lanzar un disparo se debe presionar la tecla ’c’ o el botón izquierdo del mouse, la bola de fuego se moverá hacia la dirección donde se movió Pep la última vez.
5. Los gnomos salen de su casa ubicada en la isla más alta y se mueven hacia la izquierda o derecha. Cuando caen a una isla de un nivel más bajo, cambian su dirección de manera aleatoria. Debe haber entre 2 y 4 gnomos en pantalla, reponiéndose cada cierto tiempo ya que serán rescatados por Pep, aniquilados por las tortugas, o caerán al precipicio.
6. Un gnomo es rescatado cuando colisiona con Pep. Pero solo podrá rescatarlo en las islas inferiores (en las primeras dos filas desde abajo hacia arriba). Si un gnomo cae al precipicio o es colisionado por una tortuga, se considera un gnomo perdido. En ambos casos, se debe eliminar la instancia del gnomo.
7. Las tortugas aparecen cayendo desde el borde superior de la pantalla en una posición aleatoria pero no sobre la isla de la casa de los gnomos. Cuando una tortuga llega a una isla flotante empieza a moverse de izquierda a derecha (o al revés) hasta el borde de la isla, luego debe cambiar de dirección para permanecer en la misma. Si una tortuga hace colisión con Pep o con los gnomos los aniquila ya que son muy venenosas.
8. El juego se gana cuando Pep haya rescatado una cantidad determinada de gnomos a elección del grupo. El juego se pierde cuando se pierde una cierta cantidad de gnomos, o bien cuando Pep cae al precipicio o es aniquilado por las tortugas. Aclaramos que la destrucción o desaparición de un objeto de la pantalla, implica eliminar explícitamente el objeto (debe ser nulo), no vale ´únicamente ocultar la imagen o posicionarlo fuera de la pantalla.

9. Durante todo el juego deberá mostrarse en algún lugar de la pantalla: El tiempo transcurrido del juego, la cantidad de gnomos rescatados y perdidos, y la cantidad de enemigos

Se han modificado sin embargo, algunos apartados en los requisitos referente sobre todo a la estética general del juego y a la elección de algunas teclas. Por el lado estético, se ha cambiado a Pep por un caballero; a los gnomos por monedas; y a las tortugas por esqueletos. En cuanto a la elección de teclas, se ha decidido establecer a la tecla espacio como botón para saltar, y a la tecla ctr izquierdo como botón para disparar las bolas de fuego. Por lo demás, no se modifican ninguno de los comportamientos establecidos en los requerimientos del juego.

Clases del juego

- BolaDeFuego
- Cofre
- Moneda
- Caballero
- Esqueleto
- Isla
- Juego

Caballero

La clase caballero, tiene cuatro atributos principales, los cuales son las variables de tipo int x e y; dedicadas a establecer la posición del personaje dentro del entorno del juego. A su vez, posee otras dos variables, también de tipo int llamadas ancho y alto, a través de las que se determina el ancho y alto del rectángulo utilizado en el juego para representar al caballero, y al resto de objetos en general. Por último, tanto el caballero, como otros objetos con movimiento, como pueden ser el esqueleto o la moneda, poseen una variable int llamada velocidad, por la cual modificaremos los valores x e y, simulando así el movimiento de los personajes.

En cuanto a los métodos de instancia que posee esta clase, podríamos dividirlos en dos grandes grupos. Siendo el primero, los métodos relacionados al movimiento y el segundo los métodos encargados de detectar colisiones con otros objetos.

Entre los métodos dedicados al movimiento del caballero, nos encontramos los siguientes:

- moverDerecha ()
- moverIzquierda ()
- saltar ()
- caer ()

Los mismos no hacen más que incrementar los valores de x o y, según corresponda a través de la variable velocidad. Cuanto más alto sea el valor de la velocidad, mayor será este incremento.

Un detalle importante a tener en cuenta es que dentro del entorno del juego, y contrario a lo que podemos pensar en un primer momento, para incrementar la altura de nuestro objeto dentro del entorno, es necesario decrementar el valor de y, y por lo mismo para disminuir la altura es necesario incrementar el valor y.

Podemos ver este comportamiento en la propia implementación de los métodos saltar() y caer():

```
public void saltar(){
    this.y = this.y - velocidad*3;
}
public void caer(){
    this.y = this.y + velocidad;
}
```

En cuanto a los métodos relacionados con la detección de colisiones, nos encontramos los siguientes:

- tocaAbajo (Isla isla)
- tocaMoneda (Moneda moneda)

El método tocaAbajo (), comprueba si el caballero a impactado con la superficie de alguna isla. El método tocaMoneda () por su parte, se fija si se ha tocado alguna parte del rectángulo utilizado para representar a la moneda. En ambos métodos, es necesario pasar por parámetro el objeto con el que queremos comprobar si se ha hecho colisión.

Esqueleto

Como se dijo anteriormente, los objetos con movimiento como el caballero, moneda y esqueleto; comparten los mismos atributos, y en gran medida los mismos métodos. Entre ellos están:

- caer ()
- tocaAbajo (Isla isla)
- tocaMoneda(tocaMoneda)

Los métodos moverDerecha() y moverIzquierda(), se reemplazan por los métodos mover() y cambiarDireccion(). También se añade un nuevo método que es tocaCaballero(Caballero caballero) que comprueba la colisión del esqueleto con el caballero.

Moneda

La clase moneda sigue la misma línea que las anteriores clases. Lo único que la hace destacar es la utilización de clase Random para la creación de números aleatorios. Dentro de los atributos que conforman la clase Moneda, se encuentra un objeto de tipo Random llamado simplemente random. El cual es utilizado dentro del método direccionAleatoria () para generar y devolver un numero entre 0 y 1. Es mediante estos dos números que representaremos el movimiento de izquierda o derecha.

```
private Random random = new Random();

public int direccionAleatoria(){

    direccion = random.nextInt(2);

    return direccion;

  }
```

BolaDeFuego

La clase bola de fuego en sintonia con el resto de clases vistas, posee métodos para moverse tanto hacia la izquierda como hacia la derecha, asi como un metodo hecho para detectar la colicion con esqueletos, tocaEsqueleto()

Isla y Cofre

Ambas clases tienen la característica de no tener métodos, a excepción del método dibujarIsla(), que dibuja la isla en el entorno(sin contar claro, los métodos get). Sus atributos por otra parte son los mismos que los de anteriores clases, aunque en ambas clases, no esta presente la variable velocidad, y esto es simplemente por que son objetos que no se mueven y forman parte mas bien del escenario.

Juego

Metodos utilizados en la clase Juego:

- generarIslas()
- dibujarIslas()
- int generarNumeroAleatorio()
- Isla esqueletoEstaTocandoAlgunaIsla()
- Isla caballeroEstaTocandoAlgunaIsla()
- Isla monedaEstaTocandoAlgunaIsla()
- Boolean faltaAlgunaMoneda()
- Void generarMoneda()
- Void generarEsqueleto()
- Void dibujarMoneda()

- Void dibujarEsqueleto()

Explicación de los métodos utilizados:

El método generarIslas() instancia todas las islas que se encuentran dentro de la lista que contiene las islas. La posición de cada isla dentro del entorno, esta declarada manualmente. El método dibujarIslas() permite dibujar todas estas islas, y lo hace recorriendo la lista con las islas y dibujándolas una por una de manera automática.

Los métodos esqueletoEstaTocandoAlgunaIsla(), caballeroEstaTocandoAlgunaIsla() y moneda EstaTocandoAlgunaIsla(), recorren la lista que contiene todas las islas del juego, ejecutando el metodo tocaAbajo(), presentes en las tres clases, Esqueleto, Caballero y Moneda. En cualquier caso, si se toca efectivamente la superficie de una isla, el metodo devuelve el propio objeto Isla que toco; y en caso contrario simplemente devuelve null. La razón de devolver la isla, y no simplemente un booleano, es para saber que isla se tocó, y poder hacer distintos cálculos con ella en caso de ser necesario.

El método faltaAlgunaMoneda() se encarga de revisar la lsita monedas y comprobar si alguna de sus posiciones esta en null, de ser asi retorna un true.

Los métodos generarEsqueleto() y generarMoneda(), comprueban si la lista de esqueletos y de monedas respectivamente, tienen una posición nula, y de ser así, instancian su respectivo objeto.

Los métodos dibujarMoneda() y dibujarEsqueleto(), recorren sus respectivas listas de objetos, dibujando todos sus elementos.

Explicación del comportamiento de los objetos

Siendo la clase más importante, y en donde interactúan el resto de clases, se procederá a explicar la estructura de la misma para poder entender mejor como están divididos los distintos comportamientos del programa.

A grandes rasgos la clase se divide en dos constructores. El primero Juego (), en donde se instancian todos los objetos de las distintas clases creadas anteriormente. Es aquí en donde definiremos la posición de los distintos objetos, sus proporciones y velocidad. Es también dentro de este apartado de la clase, en donde se instancia el entorno del juego, junto con su resolución.

Los distintos atributos de la clase Juego, que son los siguientes

- boolean estaCayendo = true;
- Image imagenFondo;
- Cofre cofre;
- BolaDeFuego boladefuego;
- Moneda[] monedas = new Moneda[3];
- Isla[] islas = new Isla[15];
- Caballero caballero;

- Esqueleto esqueleto;
- Esqueleto[] esqueletos = new Esqueleto[3];
- boolean confirmar = false;
- int monedasObtenidasPorCaballero;
- int esqueletosEliminados;
- int monedasPerdidas;
- int maxAlturaAlcanzada = 0;
- Image gameover = Herramientas.cargarImagen("images/gameover.jpg");

El segundo constructor es el constructor Tick(), que es el encargado de procesar un instante del juego equivalente a milisegundos de la vida real. Es dentro de este constructor que se desarrolla el apartado lógico del juego, relacionado con el comportamiento y la interacción de los distintos objetos dentro del entorno del juego.

Dado lo grande que es esta sección del código, se decidió dividirla a su vez en tres secciones indicadas a través de líneas de comentario. Las mismas son la sección dedicada al comportamiento de los esqueletos, la sección dedicada al comportamiento de las monedas, y la sección dedicada al movimiento del caballero (jugador) del que a su vez se desprende la sección dedicada al comportamiento de la bola de fuego. Pasaremos a explicar cada una de estas secciones.

Sección esqueleto

Dentro de esta sección lo que se hace es recorrer la lista que contiene todos los esqueletos del juego, comprobando primero si dicho esqueleto no esta en un estado nulo, ósea si fue instanciado. De ser así, se creará un nuevo objeto de tipo esqueleto. De esta manera nos aseguramos de que se estén creando esqueletos constantemente.

Mientras se recorre la lista, el programa se va fijando si el esqueleto revisado se encuentra tocando alguna isla; de ser así el esqueleto comenzara a moverse hacia un extremo de la misma, y al llegar a dicho extremo se ejecutará el método cambiar Dirección (), haciendo que comience a moverse en sentido contrario. En caso de que nunca se toque una isla, el esqueleto caerá al vacío, momento en el que será eliminado, y reemplazado por otro esqueleto en el escenario.

Paralelamente también se comprueba si el esqueleto colisiona con alguna moneda. Esto se logra recorriendo la lista que contiene las monedas, y ejecutando constantemente el método tocaMoneda (). En caso de que efectivamente se haya colisionado con una moneda, la misma será eliminada de la lista.

Similar es lo que se hace con el caballero, solo que para esto no es necesario recorrer ninguna isla; basta con que se pase el objeto de tipo Caballero, y se ejecute el método toca Caballero() sobre el esqueleto en cuestión.

```
for (int i = 0; i < esqueletos.length; i++) {
```

```
                Esqueleto esqueleto = esqueletos[i];

                if(esqueletos[i] != null){

                    if(esqueletos[i].getY() >= 600){
                        esqueletos[i] = null;
                    }
                    if(esqueletoEstaTocandoAlgunaIsla(islas, esqueleto)
!= null){
                        Isla isla =
esqueletoEstaTocandoAlgunaIsla(islas,esqueleto); //Guarda la isla que el
esqueleto toco

                        //Comprueban si el esqueleto llego al borde
derecho o izquierdo de la isla
                        boolean tocaDerecha = esqueleto.getX() +
esqueleto.getAncho()/2 == isla.getX() + isla.getAncho()/2;
                        boolean tocaIzquierda = esqueleto.getX() -
esqueleto.getAncho()/2 == isla.getX() - isla.getAncho()/2;


                        if(tocaDerecha || tocaIzquierda){
                            esqueleto.cambiarDireccion();
                            esqueleto.mover(); //Es importante para que
el esqueleto reactive su movimiento al llegar a un borde, y no se quede
quieto en el mismo.
                        }else{
                            esqueleto.mover();
                        }
                    }else{
                        esqueleto.caer();
                    }
                    //Comprueba si el esqueleto toca alguna moneda
                    for (int j = 0; j < monedas.length; j++) {

                        Moneda moneda = monedas[i];

                        if(esqueleto.tocaMoneda(moneda)){

                            this.monedasPerdidas ++;
                            monedas[i] = null;

                        }
                    }
                    //Comprueba si el esqueleto toca al caballero
                    if(esqueleto.tocaCaballero(caballero)){
                        caballero = null;
                    }
```

```
                }else{
                    generarEsqueleto(esqueletos);
                    dibujarEsqueletos(esqueletos);
                }

            }
```

Sección Moneda

El comportamiento de las monedas es muy similar al del esqueleto, al igual que estos, las monedas poseen su propia lista así como métodos propios para saber si se colisiona con las islas. La principal diferencia es que cuando una moneda toca una isla, la dirección hacia la que se dirige es aleatoria, y que a diferencia del esqueleto, esta continua su movimiento hasta caer hasta otra isla, o en su defecto hacia el vacío.

Para recrear este movimiento aleatorio, se ejecuta el método direccionAleatoria () cada vez que la moneda este cayendo en el aire, para que al caer a la isla esta dirección ya este definida. Se hace mientras se cae en el aire y no una vez que toca el suelo, por que de hacerlo de este modo, se estaría cambiando continuamente la dirección de la moneda.

A su vez, mientras se recorre la lista, comprueba si el caballero toca alguna moneda, a través del método tocaMoneda() del caballero, y de ser así la moneda es eliminada y contabilizada dentro de la variable monedasObtenidasPorCaballero.

```
for (int i = 0; i < monedas.length; i++) {

                Moneda moneda = monedas[i];


                if(monedas[i] != null){

                    //Si se esta tocando alguna isla..
                    if (monedaEstaTocandoAlgunaIsla(islas, moneda) !=
null) {

                        //La direccion a la que se mueve la moneda, se
indica mediante numeros. (1 == izquierda) (0 == derecha)
                        if(moneda.getDireccion() == 1){
                            moneda.moverIzquierda();
                        }else{
                            moneda.moverDerecha();
                        }
                    /*
                     * La direccion a la que se mueve la moneda es
aleatoria, y se decide mientras la misma cae.
                     *  Esto es devido a que si la moneda decidiese la
direccion a la que se deve mover mientras esta
```

```
                    *      tocando alguna isla, la misma estaria
cambiando todo le tiempo.
                    */
                   }else{

                       moneda.direccionAleatoria();
                       moneda.caer();
                   }
                   //Si se supera el margen de la pantalla se elimina la
moneda
                   if(moneda.getY() >= 600){
                       this.monedasPerdidas ++;
                       monedas[i] = null;
                   }
                   /*
                    * Las monedas podran ser recogidas por el caballero,
unicamente si se encuentran por la mitad inferior del entorno.
                    *  Es decir, se podran recoger unicamente en las
islas inferiores. Se evito utilizar metodos que comprueben si
                    *  la moneda o el caballero se encuentran tocando
las islas inferiores, debido a que de hacerlo, no se podrian
                    *  recoger las monedas que estan cayendo en el aire.
                    */
                   if(caballero.tocaMoneda(moneda) && moneda.getY() >
300){
                       monedas[i] = null;
                       monedasObtenidasPorCaballero++;
                   }

               }else{
                   generarMonedas(monedas);
                   dibujarMonedas(monedas);
               }

           }
```

Sección caballero

Esta sección dista bastante de las dos anteriores por el simple hecho de que el movimiento del caballero es determinado por el jugador, y no por un comportamiento previamente programado. Primeramente se decidió que el salto del caballero, se hiciese al mantener presionada la tecla espacio, haciendo que el valor y del caballero incremente de forma escalonada, en contraposición a hacer que el valor de y incremente a un valor fijo al tocar una única vez la tecla espacio (como se pensó hacer originalmente). Esto por dos motivos, uno es que el movimiento de esta manera se siente menos mecánico, y segundo que al incrementar el valor y poco a poco y no de golpe, es más fácil detectar colisiones con otras islas.

Siguiendo esta lógica, lo siguiente que se hizo, para que el personaje no pareciera estar volando en el escenario en vez de saltando; fue poner un límite máximo de altura a la que se podía llegar manteniendo la tecla de saltar apretada. Este valor máximo de altura, no corresponde a un valor de y, (dato importante) sino simplemente a un numero acumulativo. Podría verse también como un limite en cuanto tiempo podemos mantener apretado la tecla de saltar, mas que como un límite de altura. En cualquier caso, cuando este limite es superado, el caballero cae automáticamente al vacío, sin posibilidad de reactivar su salta, a excepción de que se toque alguna isla.

```
if(caballeroEstaTocandoAlgunaIsla(islas)==null &&
!entorno.estaPresionada(entorno.TECLA_ESPACIO)){
                caballero.caer();
                estaCayendo = true;
            }
            if(caballeroEstaTocandoAlgunaIsla(islas)!= null){
                maxAlturaAlcanzada = 0;
                estaCayendo = false;
            }
            if(maxAlturaAlcanzada > 15){
                caballero.caer();
                estaCayendo = true;
            }
            if(estaCayendo == false){
                if(entorno.estaPresionada(entorno.TECLA_ESPACIO) &&
maxAlturaAlcanzada <= 15){
                    caballero.saltar();
                    maxAlturaAlcanzada++;

                }
            }

            if(caballero.getY() > 600){
                caballero = null;
            }
            if(estaPresionadaDerecha){
                caballero.moverDerecha();

            }
            if (estaPresionadaIzquierda) {
                caballero.moverIzquierda();
            }
```

Sección Bola de fuego

La sección de la bola de fuego se divide en dos partes, y estas son cuando la bola de fuego es creada, y cuando la bola de fuego es dibujada y se mueve.

Cuando el jugador acciona la tecla ctrl, que es la encargada de generar el disparo, la bola de fuego es creada en las mismas coordenadas que el caballero. A su vez se guarda la última dirección a la que se movió el caballero, y además se cambia a true la variable ***confirmar*** que es la que nos permitirá controlar en qué momento podemos o no dibujar y mover la bola de fuego.

Cuando esta variable ***confirmar*** es accionada (está en true), la bola de fuego se dibujará y comenzará a moverse en la misma dirección que nos movimos por última vez con el caballero. La bola de fuego continuara su movimiento hasta que se salga de los márgenes del entorno, o hasta que colisione con algún esqueleto, momento en que el estado de la variable ***confirma*** pasara a false. Con respecto a la eliminación de los esqueletos, similar a lo hecho con las monedas, se recorre la lista que contiene los esqueletos, ejecutando el método tocaEsqueleto() de la clase BolaDeFuego.

La bola de fuego no puede volver a ser creada si no se cumplió alguno de los requisitos anteriores, ya que para crear la bola de fuego es necesario que la variable ***confirmar*** sea igual a false. De esta manera evitamos que se puedan spamear bolas de fuego de manera indiscriminada.

```
if(entorno.sePresiono(entorno.TECLA_CTRL) && confirmar == false){

                //Controla cuando se crea la bola de fuego, para que no
cree nuevas bolas de fuego mientras hay una en pantalla.
                confirmar = true;
                boladefuego = new BolaDeFuego(caballero.getX(),
caballero.getY(),10, 10, 10);
                if(entorno.estaPresionada(entorno.TECLA_IZQUIERDA) ||
entorno.sePresiono(entorno.TECLA_IZQUIERDA)){
                    caballero.setDireccion(1);
                }
                if(entorno.estaPresionada(entorno.TECLA_DERECHA) ||
entorno.sePresiono(entorno.TECLA_DERECHA)){
                    caballero.setDireccion(2);
                }

            }

            if(confirmar == true){

                if(caballero.getDireccion() == 1){
                    if(boladefuego.x > 0){
                        boladefuego.dibujarBolaDeFuego(entorno);
                        boladefuego.moverIzquierda();

                    }else{
                        confirmar = false; //Cuando la bola de fuego
supera el margen del entorno, se nos permite volver a generar otra bola
de fuego
```

```
                    }
                }else{
                    if(boladefuego.x <= entorno.ancho()){
                        boladefuego.dibujarBolaDeFuego(entorno);
                        boladefuego.moverDerecha();

                    }else{
                        confirmar = false; //Cuando la bola de fuego
supera el margen del entorno, se nos permite volver a generar otra bola
de fuego
                    }
                }

                //Seccion dedicada a comprobar si la bola de fuego
impacta con algun esqueleto
                for (int i = 0; i < esqueletos.length; i++) {

                    Esqueleto esqueleto = esqueletos[i];

                    if(boladefuego.tocaEsqueleto(esqueleto)){

                        this.esqueletosEliminados ++;
                        esqueletos[i] = null;
                        confirmar = false; //Al eliminar un esuqueleto,
se nos permite volver a generar otra bola de fuego
                    }
                }
            }
        }else{
            //Se activa la pantalla de game over
            entorno.dibujarImagen(gameover, 400, 300, 0, 1);
        }
```

Código completo

Caballero

```
package juego;

import java.awt.Color;
import java.awt.Image;
import entorno.Entorno;
import entorno.Herramientas;

public class Caballero {

    public BolaDeFuego boladefuego;
    public int alto,ancho;
    private int x,y;
    private int velocidad;
    private int direccion = 0;
    private int direccionImagen = 0;
    public Image imagenCaballero =
Herramientas.cargarImagen("images/caballero.png");
    public Image imagenIzquierda =
Herramientas.cargarImagen("images/caballero-izquierda.png");

    public Caballero(int x, int y, int velocidad,int ancho, int alto){
        this.x = x;
        this.y = y;
        this.velocidad = velocidad;
        this.ancho = ancho;
        this.alto = alto;

    }

    public void moverDerecha(){
        this.x = this.x + velocidad;
        this.direccionImagen = 2;
    }
    public void moverIzquierda(){
        this.x = this.x - velocidad;
        this.direccionImagen = 1;

    }
    public void saltar(){
        this.y = this.y - velocidad*3;
    }
    public void caer(){

        this.y = this.y + velocidad;
    }
    public boolean tocaAbajo(Isla isla){
```

```
        boolean tocaX = this.x >= isla.getX() - isla.getAncho()/2 &&
this.x <= isla.getX() + isla.getAncho()/2;
        /*
         * Este margen de +1. Es que, dado a que el numero y, se
incrementa o decrementa en base a la velocidad, no siempre
         *  esta dara justo en el valor y de de la isla, por lo que es
necesario ponerle un pequeño margen
        */
        boolean tocaY = this.y + this.alto/2 >= (isla.getY() -
isla.getAlto()/2) -1 && this.y + this.alto/2 <= (isla.getY() -
isla.getAlto()/2) +1;
        return tocaY && tocaX;
    }
    public boolean tocaArriba(Isla isla){
        boolean tocaX = this.x >= isla.getX() - isla.getAncho()/2 &&
this.x <= isla.getX() + isla.getAncho()/2;
        boolean tocaY = this.y - this.alto/2 >= (isla.getY() +
isla.getAlto()/2) -2 && this.y - this.alto/2 <= (isla.getY() +
isla.getAlto()/2) +2;
        return tocaY && tocaX;
    }

    public boolean tocaMoneda(Moneda moneda){
        boolean tocaX = this.x >= moneda.getX() - moneda.getAncho()/2 &&
this.x <= moneda.getX() + moneda.getAncho()/2;
        boolean tocaY = this.y >= moneda.getY() - moneda.getAlto()/2 &&
this.y <= moneda.getY() + moneda.getAlto()/2;
        return tocaX && tocaY;
    }
    public void dibujarCaballero(Entorno entorno){
        //entorno.dibujarRectangulo(x, y, ancho, alto, 0, Color.RED);
        if(direccionImagen == 1){
            entorno.dibujarImagen(this.imagenIzquierda, this.x, this.y,
0, 0.08);
        }else{
            entorno.dibujarImagen(this.imagenCaballero, this.x, this.y,
0, 0.08);
        }
        //entorno.dibujarImagen(this.imagenCaballero, this.x, this.y, 0,
0.08);
    }

    ////////Setter y Getter///////
    //Getters
    public int getX(){
        return this.x;
    }
    public int getY(){
        return this.y;
```

```
    }
    public int getAncho(){
        return this.ancho;
    }
    public int getAlto(){
        return this.alto;
    }
    public int getVelocidad(){
        return this.velocidad;
    }
    public Image getImagenCaballero(){
        return this.imagenCaballero;
    }
    public int getDireccion(){
        return this.direccion;
    }

    //Setters
    public void setX(int x){
        this.x = x;
    }
    public void setY(int y){
        this.y = y;
    }
    public void setVelocidad(int velocidad){
        this.velocidad = velocidad;
    }
    public void setDireccion(int direccion){
        this.direccion = direccion;
    }
}
```

Esqueleto

```
package juego;

import java.awt.Color;
import java.awt.Image;
import entorno.Entorno;
import entorno.Herramientas;

public class Esqueleto {

    private int x,y;
    private int ancho,alto;
    private int velocidad;
    boolean estaVivo;  // Atributo que indica si el esqueleto está vivo
```

```
    Image imagenEsqueleto;
    public Esqueleto(int x, int y, int ancho, int alto, int velocidad){
        this.x = x;
        this.y = y;
        this.ancho = ancho;
        this.alto = alto;
        this.velocidad = velocidad;
        this.imagenEsqueleto =
Herramientas.cargarImagen("images/esqueleto.png");
    }
    public Esqueleto() {
        estaVivo = true;  // Inicialmente el esqueleto está vivo
    }

    public void morir() {
        estaVivo = false;  // Cambia el estado a muerto cuando el
esqueleto es destruido
    }

    public void saltar(){
        this.y = this.y - velocidad;
    }
    public void caer(){

        this.y = this.y + velocidad;
    }
    public void mover(){
        this.x = this.x + this.velocidad;
    }
    public void cambiarDireccion(){
        this.imagenEsqueleto =
Herramientas.cargarImagen("images/esqueleto1.png");
        this.velocidad = this.velocidad *(-1);
    }
    public boolean tocaAbajo(Isla isla){
        boolean tocaX = this.x >= isla.getX() - isla.getAncho()/2 &&
this.x <= isla.getX() + isla.getAncho()/2;
        boolean tocaY = this.y + this.alto/2 == isla.getY() -
isla.getAlto()/2;
        return tocaY && tocaX;
    }
    public boolean tocaMoneda(Moneda moneda){
        boolean tocaX = this.x >= moneda.x - moneda.ancho/2 && this.x <=
moneda.x + moneda.ancho/2;
        boolean tocaY = this.y >= moneda.y - moneda.alto/2 && this.y <=
moneda.y + moneda.alto/2;
        return tocaX && tocaY;
    }
    public boolean tocaCaballero(Caballero caballero){
```

```
        boolean tocaX = this.x >= caballero.getX() -
caballero.getAncho()/2 && this.x <= caballero.getX() +
caballero.getAncho()/2;
        boolean tocaY = this.y >= caballero.getY() -
caballero.getAlto()/2 && this.y <= caballero.getY() +
caballero.getAlto()/2;
        return tocaX && tocaY;
    }

    public void dibujarEsqueleto(Entorno entorno){
        //entorno.dibujarRectangulo(x, y, ancho, alto, 0, Color.WHITE);
        entorno.dibujarImagen(imagenEsqueleto, this.x, this.y, 0, 0.1);
    }

    //Setter y Getters
    //Getters
    public int getX(){
        return this.x;
    }
    public int getY(){
        return this.y;
    }
    public int getVelocidad(){
        return this.velocidad;
    }
    public int getAncho(){
        return this.ancho;
    }
    public int getAlto(){
        return this.alto;
    }


    //Setters
    public void setX(int x){
        this.x = x;
    }
    public void setY(int y){
        this.y = y;
    }
    public void setVelocidad(int velocidad){
        this.velocidad = velocidad;
    }
}
```

Moneda

```
package juego;
```

```java
import java.awt.Color;
import java.awt.Image;
import java.util.Random;

import entorno.Entorno;
import entorno.Herramientas;

public class Moneda {

    public int x,y;
    public int ancho,alto;
    private int velocidad;
    public Image imagen;
    public Random random = new Random();
    public int direccion = 0;
    public Moneda(int x, int y, int ancho, int alto, int velocidad){
        this.x = x;
        this.y = y;
        this.ancho = ancho;
        this.alto = alto;
        this.velocidad = velocidad;
        this.imagen = Herramientas.cargarImagen("images/moneda4.gif");
    }
    public void caer(){

        this.y = this.y + velocidad;
    }
    public void mover(){
        this.x = this.x + this.velocidad;
    }
    public void moverDerecha(){
        this.x = this.x + this.velocidad;
    }
    public void moverIzquierda(){
        this.x = this.x - this.velocidad;
    }
    public int direccionAleatoria(){
        direccion = random.nextInt(2);
        return direccion;
    }
    public void cambiarDireccion(){
        this.velocidad = this.velocidad *(-1);
    }
    public boolean tocaAbajo(Isla isla){
        boolean tocaX = this.x >= isla.getX() - isla.getAncho()/2 &&
this.x <= isla.getX() + isla.getAncho()/2;
        boolean tocaY = this.y + this.alto/2 == isla.getY() -
isla.getAlto()/2;
        return tocaY && tocaX;
```

Fernandez Facundo

```
    }
    public void dibujarMoneda(Entorno entorno){

        entorno.dibujarImagen(imagen, x, y, 0, 0.15);
    }
}
```

Isla

```
package juego;

import java.awt.Color;
import java.awt.Image;

import entorno.Entorno;
import entorno.Herramientas;

public class Isla {

    private Image imagenIsla;
    private int x,y;
    private int ancho, alto;
    Image isla;
    public Isla(int x, int y, int ancho, int alto){

        this.x = x;
        this.y = y;
        this.ancho = ancho;
        this.alto = alto;
        this.imagenIsla = Herramientas.cargarImagen("images/isla-
redimensionada.jpg");
    }

    public void dibujarIsla(Entorno entorno){
        //entorno.dibujarRectangulo(x, y, ancho, alto, 0, Color.CYAN);
        entorno.dibujarImagen(this.imagenIsla, this.x, this.y, 0, 1);
    }


    //Metodos getter

    public int getX(){
        int x = this.x;
        return x;
    }
    public int getY(){
        int y = this.y;
        return y;
    }
```

```
    public Image getImagen(){
        return this.imagenIsla;
    }
    public int getAncho(){
        return this.ancho;
    }
    public int getAlto(){
        return this.alto;
    }
}
```

Musica

```
package juego;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.AudioInputStream;
import java.io.File;

public class Musica {
    private Clip clip;

    public Musica(String path) {
        try {
            File file = new File(path);
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(file);
            clip = AudioSystem.getClip();
            clip.open(audioStream);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void reproducir() {
        if (clip != null) {
            clip.start();
            clip.loop(Clip.LOOP_CONTINUOUSLY); // Esto hace que la música
se repita
        }
    }

    public void detener() {
        if (clip != null) {
            clip.stop();
        }
    }
}
```

BolaDeFuego

```
package juego;

import java.awt.Color;
import java.awt.Image;
import entorno.Entorno;
import entorno.Herramientas;

public class BolaDeFuego {

    Image imagenboladefuego =
Herramientas.cargarImagen("images/boladefuego4.gif");
    int x, y;
    int ancho, alto;
    int velocidad;
    int direccion;  // 1 para derecha, -1 para izquierda

    public BolaDeFuego(int x, int y, int ancho, int alto, int velocidad,
int direccion) {
        this.x = x;
        this.y = y;
        this.ancho = ancho;
        this.alto = alto;
        this.velocidad = velocidad;
        this.direccion = direccion; // Dirección determinada por el
caballero
    }

    public void dibujarBolaDeFuego(Entorno entorno) {
        entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto,
0, Color.ORANGE);
        entorno.dibujarImagen(imagenboladefuego, x, y, 0, 0.2);
    }

    public void mover() {
        this.x = this.x + (velocidad * direccion);  // La dirección
afecta el movimiento
    }

    public void comprobarReposicion(int pantallaAncho) {
        // Si la bola de fuego se mueve a la izquierda y sale de la
pantalla
        if (this.direccion == -1 && this.x < 0) {
            this.x = pantallaAncho;  // Reposicionarla al lado derecho de
la pantalla
            this.y = 200; // Ajustar la posición vertical si es necesario
        }
```

```
        // Si la bola de fuego se mueve a la derecha y sale de la
pantalla
        else if (this.direccion == 1 && this.x > pantallaAncho) {
            this.x = 0;  // Reposicionarla al lado izquierdo de la
pantalla
            this.y = 200; // Ajustar la posición vertical si es necesario
        }
    }

    public boolean tocaEsqueleto(Esqueleto esqueleto) {
        boolean tocaX = this.x >= esqueleto.getX() - esqueleto.getAncho()
/ 2 && this.x <= esqueleto.getX() + esqueleto.getAncho() / 2;
        boolean tocaY = this.y >= esqueleto.getY() - esqueleto.getAlto()
/ 2 && this.y <= esqueleto.getY() + esqueleto.getAlto() / 2;
        return tocaX && tocaY;
    }
}
```

Juego

```
package juego;

import entorno.Entorno;
import entorno.Herramientas;
import entorno.InterfaceJuego;
import java.util.Random;

import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

import java.awt.Color;
import java.awt.Image;
import java.io.File;

public class Juego extends InterfaceJuego
{
    // El objeto Entorno que controla el tiempo y otros
    private Entorno entorno;

    // Variables y métodos propios de cada grupo
    public Random random = new Random();



    boolean estaCayendo = true;
    private Musica musicaFondo;
    Image imagenFondo;
```

```
    Cofre cofre;
    BolaDeFuego boladefuego;
    Moneda[] monedas = new Moneda[3];
    Isla[] islas = new Isla[15];
    Caballero caballero;
    Esqueleto esqueleto;
    Esqueleto[] esqueletos = new Esqueleto[3];
    boolean confirmar = false;
    int monedasObtenidasPorCaballero;
    int esqueletosEliminados;
    int monedasPerdidas;
    int maxAlturaAlcanzada = 0;
    Image gameover = Herramientas.cargarImagen("images/game-over-
glitch.gif");
    Juego()
    {
        // Inicializa el objeto entorno
        this.entorno = new Entorno(this, "Al Rescate de los Gnomos", 800,
600);
         // Cargar la música de fondo (indica la ruta al archivo .wav)
         musicaFondo = new Musica("src/sounds/Alrescate.wav");
         musicaFondo.reproducir(); // Comienza la música al inicio del
juego

        // Inicializar lo que haga falta para el juego
        imagenFondo = Herramientas.cargarImagen("images/fondo.gif");


        esqueletos[0] = new Esqueleto(70,100,20,30,1);
        esqueletos[1] = new Esqueleto(300,100,20,30,1);
        esqueletos[2] = new Esqueleto(680,100,20,30,1);
        //monedaAncho 12, alto 12;
        monedas[0] = new Moneda(150, 100, 20, 20, 1);
        monedas[1] = new Moneda(300, 100, 20, 20, 1);
        monedas[2] = new Moneda(500, 100, 20, 20, 1);

        generarIslas();
        cofre = new Cofre(400,65,30,30);
        caballero = new Caballero(400, 400, 3,20,30);


        // Inicia el juego!
        this.entorno.iniciar();
    }

    /**
     * Durante el juego, el método tick() será ejecutado en cada instante
y
```

```
     * por lo tanto es el método más importante de esta clase. Aquí se
debe
     * actualizar el estado interno del juego para simular el paso del
tiempo
     * (ver el enunciado del TP para mayor detalle).
     */
    public void tick()
    {
        // Procesamiento de un instante de tiempo

        //El juego se mantendra funcionando mientras el caballero no sea
elminiado
        if(caballero != null && monedasObtenidasPorCaballero < 20 &
monedasPerdidas < 10){
            entorno.dibujarImagen(imagenFondo, 400, 300, 0, 0.8);
            dibujarMonedas(monedas);
            dibujarEsqueletos(esqueletos);
            dibujarIslas(islas);
            cofre.dibujarCofre(entorno);
            //esqueleto.dibujarEsqueleto(entorno);
            caballero.dibujarCaballero(entorno);

            entorno.cambiarFont("Impact", 18, Color.WHITE);
            entorno.escribirTexto("Monedas recogidas: " +
monedasObtenidasPorCaballero, 50, 50);

            entorno.cambiarFont("Impact", 18, Color.WHITE);
            entorno.escribirTexto("Monedas perdidas: " + monedasPerdidas,
50, 75);

            entorno.cambiarFont("Impact", 18, Color.WHITE);
            entorno.escribirTexto("Esqueletos eliminados: " +
esqueletosEliminados , 50, 100);

            entorno.cambiarFont("Impact", 18, Color.WHITE);
            entorno.escribirTexto("Tiempo transcurrido: " +
entorno.tiempo()/1000, 550, 100);



            //SECCION DE CABALLERO Y ESQUELETO

            boolean estaPresionadaDerecha =
entorno.estaPresionada(entorno.TECLA_DERECHA);
            boolean estaPresionadaIzquierda =
entorno.estaPresionada(entorno.TECLA_IZQUIERDA);



            //MOVIMIENTO MONEDA (Gnomo)
```

```
            for (int i = 0; i < monedas.length; i++) {

                Moneda moneda = monedas[i];


                if(monedas[i] != null){

                    //Si se esta tocando alguna isla..
                    if (monedaEstaTocandoAlgunaIsla(islas, moneda) !=
null) {

                        //La direccion a la que se mueve la moneda, se
indica mediante numeros. (1 == izquierda) (0 == derecha)
                        if(moneda.getDireccion() == 1){
                            moneda.moverIzquierda();
                        }else{
                            moneda.moverDerecha();
                        }
                    /*
                     * La direccion a la que se mueve la moneda es
aleatoria, y se decide mientras la misma cae.
                     *  Esto es devido a que si la moneda decidiese la
direccion a la que se deve mover mientras esta
                     *      tocando alguna isla, la misma estaria
cambiando todo le tiempo.
                     */
                    }else{

                        moneda.direccionAleatoria();
                        moneda.caer();
                    }
                    //Si se supera el margen de la pantalla se elimina la
moneda
                    if(moneda.getY() >= 600){
                        this.monedasPerdidas ++;
                        monedas[i] = null;
                    }
                    /*
                     * Las monedas podran ser recogidas por el caballero,
unicamente si se encuentran por la mitad inferior del entorno.
                     *  Es decir, se podran recoger unicamente en las
islas inferiores. Se evito utilizar metodos que comprueben si
                     *  la moneda o el caballero se encuentran tocando
las islas inferiores, debido a que de hacerlo, no se podrian
                     *  recoger las monedas que estan cayendo en el aire.
                     */
                    if(caballero.tocaMoneda(moneda) && moneda.getY() >
300){
                        monedas[i] = null;
```

```
                        monedasObtenidasPorCaballero++;
                        reproducirSonidoMoneda();
                    }

                }else{
                    generarMonedas(monedas);
                    dibujarMonedas(monedas);
                }

            }
            //MOVIMIENTO ESQUELETO
            for (int i = 0; i < esqueletos.length; i++) {

                Esqueleto esqueleto = esqueletos[i];

                if(esqueletos[i] != null){

                    if(esqueletos[i].getY() >= 600){
                        esqueletos[i] = null;
                    }
                    if(esqueletoEstaTocandoAlgunaIsla(islas, esqueleto)
!= null){
                        Isla isla =
esqueletoEstaTocandoAlgunaIsla(islas,esqueleto); //Guarda la isla que el
esqueleto toco

                        //Compruebban si el esqueleto llego al borde
derecho o izquierdo de la isla
                        boolean tocaDerecha = esqueleto.getX() +
esqueleto.getAncho()/2 == isla.getX() + isla.getAncho()/2;
                        boolean tocaIzquierda = esqueleto.getX() -
esqueleto.getAncho()/2 == isla.getX() - isla.getAncho()/2;


                        if(tocaDerecha || tocaIzquierda){
                            esqueleto.cambiarDireccion();
                            esqueleto.mover(); //Es importante para que
el esqueleto reactive su movimiento al llegar a un borde, y no se quede
quieto en el mismo.
                        }else{
                            esqueleto.mover();
                        }
                    }else{
                        esqueleto.caer();
                    }
                    //Comprueba si el esqueleto toca alguna moneda
                    for (int j = 0; j < monedas.length; j++) {

                        Moneda moneda = monedas[i];
```

```
                    if(esqueleto.tocaMoneda(moneda)){

                        this.monedasPerdidas ++;
                        monedas[i] = null;

                    }
                }
                //Comprueba si el esqueleto toca al caballero
                if(esqueleto.tocaCaballero(caballero)){
                    caballero = null;
                }
            }else{
                generarEsqueleto(esqueletos);
                dibujarEsqueletos(esqueletos);
            }

        }


        //MOVIMIENTO CABALLERO

        if(caballeroEstaTocandoAlgunaIsla(islas)==null &&
!entorno.estaPresionada(entorno.TECLA_ESPACIO)){
            caballero.caer();
            estaCayendo = true;
        }
        if(caballeroEstaTocandoAlgunaIsla(islas)!= null){
            maxAlturaAlcanzada = 0;
            estaCayendo = false;
        }
        if(maxAlturaAlcanzada > 15){
            caballero.caer();
            estaCayendo = true;
        }
        if(estaCayendo == false){
            if(entorno.estaPresionada(entorno.TECLA_ESPACIO) &&
maxAlturaAlcanzada <= 15){
                caballero.saltar();
                reproducirSonidoSalto();
                maxAlturaAlcanzada++;

            }
        }

        if(caballero.getY() > 600){
            caballero = null;
        }
        if(estaPresionadaDerecha){
```

```
                caballero.moverDerecha();

            }
            if (estaPresionadaIzquierda) {
                caballero.moverIzquierda();
            }

            //SECCION BOLA DE FUEGO

            if(entorno.sePresiono(entorno.TECLA_CTRL) && confirmar ==
false){

                //Controla cuando se crea la bola de fuego, para que no
cree nuevas bolas de fuego mientras hay una en pantalla.
                confirmar = true;
                boladefuego = new BolaDeFuego(caballero.getX(),
caballero.getY(),10, 10, 10);
                if(entorno.estaPresionada(entorno.TECLA_IZQUIERDA) ||
entorno.sePresiono(entorno.TECLA_IZQUIERDA)){
                    caballero.setDireccion(1);
                }
                if(entorno.estaPresionada(entorno.TECLA_DERECHA) ||
entorno.sePresiono(entorno.TECLA_DERECHA)){
                    caballero.setDireccion(2);
                }
                reproducirSonidoBolaDeFuego();
            }

            if(confirmar == true){

                if(caballero.getDireccion() == 1){
                    if(boladefuego.x > 0){
                        boladefuego.dibujarBolaDeFuego(entorno);
                        boladefuego.moverIzquierda();

                    }else{
                        confirmar = false; //Cuando la bola de fuego
supera el margen del entorno, se nos permite volver a generar otra bola
de fuego
                    }
                }else{
                    if(boladefuego.x <= entorno.ancho()){
                        boladefuego.dibujarBolaDeFuego(entorno);
                        boladefuego.moverDerecha();

                    }else{
                        confirmar = false; //Cuando la bola de fuego
supera el margen del entorno, se nos permite volver a generar otra bola
de fuego
```

```
                    }
                }

                //Seccion dedicada a comprobar si la bola de fuego
impacta con algun esqueleto
                for (int i = 0; i < esqueletos.length; i++) {

                    Esqueleto esqueleto = esqueletos[i];

                    if(boladefuego.tocaEsqueleto(esqueleto)){

                        this.esqueletosEliminados ++;
                        esqueletos[i] = null;
                        reproducirSonidoEliminarEsqueleto();
                        confirmar = false; //Al eliminar un esuqueleto,
se nos permite volver a generar otra bola de fuego
                    }
                }
            }
        }else{
            //Se activa la pantalla de game over
            entorno.dibujarImagen(gameover, 400, 300, 0, 2);
            reproducirSonidoGameOver();
        }




    }
    //SONIDOS

    private void reproducirSonidoSalto() {
        try {
            File sonido = new File("src/sounds/Salto.wav");
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(sonido));
            clip.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void reproducirSonidoMoneda() {
        try {
            File sonido = new File("src/sounds/moneda.wav");
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(sonido));
            clip.start();
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }


    private void reproducirSonidoBolaDeFuego() {
        try {
            File sonido = new File("src/sounds/Fuegodisparo.wav");
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(sonido));
            clip.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void reproducirSonidoEliminarEsqueleto() {
        try {
            File sonido = new
File("src/sounds/eliminacionEsqueleto.wav");
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(sonido));
            clip.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void reproducirSonidoGameOver() {
        try {
            File sonido = new File("src/sounds/gameover.wav");
            Clip clip = AudioSystem.getClip();
            clip.open(AudioSystem.getAudioInputStream(sonido));
            clip.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //METODOS
    public void generarIslas(){

        //Islas ordenadas de islas inferiores a islas superiores

        islas[0] = new Isla(100, 500, 100, 30);
        islas[1] = new Isla(250, 500, 100, 30);
        islas[2] = new Isla(400, 500, 100, 30);
        islas[3] = new Isla(550, 500, 100, 30);
        islas[4] = new Isla(700, 500, 100, 30);
```

```
        islas[5] = new Isla(175, 400, 100, 30);
        islas[6] = new Isla(325, 400, 100, 30);
        islas[7] = new Isla(475, 400, 100, 30);
        islas[8] = new Isla(625, 400, 100, 30);

        islas[9] = new Isla(250, 300, 100, 30);
        islas[10] = new Isla(400, 300, 100, 30);
        islas[11] = new Isla(550, 300, 100, 30);


        islas[12] = new Isla(325, 200, 100, 30);
        islas[13] = new Isla(475, 200, 100, 30);

        islas[14] = new Isla(400, 100, 100, 30);
    }
    public void dibujarIslas(Isla[] isla){
        for (int i = 0; i < isla.length; i++) {
            islas[i].dibujarIsla(entorno);
        }
    }

    //Hecho con chatGPT// Genera nnumeros aleatorios entre 0 y 800
(excluyendo los valores que corresponden al X de la isla superior)
    public static int generarNumeroAleatorio(Random random) {
        // Generamos un número aleatorio en el rango [0, 700] (350
números en 0-349 y 351 en 450-800)
        int indice = random.nextInt(701); // 701 porque queremos un
número entre 0 y 700 inclusive

        // Mapeamos el índice al rango adecuado
        if (indice < 350) {
            return indice; // Corresponde al rango 0-349
        } else {
            return 450 + (indice - 350); // Corresponde al rango 450-800
        }
    }
    public Isla esqueletoEstaTocandoAlgunaIsla(Isla[] islas, Esqueleto
esqueleto){
        for (int i = 0; i < islas.length; i++) {
            Isla isla = islas[i];
            if(esqueleto.tocaAbajo(isla)){
                return isla;
            }
        }


        return null;
    }
```

```
public Isla caballeroEstaTocandoAlgunaIsla(Isla[] islas){
    for (int i = 0; i < islas.length; i++) {
        Isla isla = islas[i];
        if(caballero.tocaAbajo(isla)){
            return isla;
        }
    }
    return null;
}
public Isla caballeroChocaConAlgunaIsla(Isla[] islas){
    for (int i = 0; i < islas.length; i++) {
        Isla isla = islas[i];
        if(caballero.tocaArriba(isla)){
            return isla;
        }
    }
    return null;
}
public Isla monedaEstaTocandoAlgunaIsla(Isla[] islas,Moneda moneda){
    for (int i = 0; i < islas.length; i++) {
        Isla isla = islas[i];
        if (moneda.tocaAbajo(isla)) {
            return isla;
        }
    }
    return null;
}
public boolean faltaAlgunaMoneda(Moneda[] monedas){
    for (int i = 0; i < monedas.length; i++) {
        if(monedas[i] != null){
            return true;
        }
    }
    return false;
}
public void generarMonedas(Moneda[] monedas){
    for (int i = 0; i < monedas.length; i++) {
        if(monedas[i] == null){
            monedas[i] = new Moneda(cofre.x, cofre.y, 20, 20, 1);
        }
    }
}
public void generarEsqueleto(Esqueleto[] esqueletos){
    //int cordenadaX = 0;
    for (int i = 0; i < esqueletos.length; i++) {
        if(esqueletos[i] == null){
            int cordenadaX = generarNumeroAleatorio(random);
            System.out.println(cordenadaX);
```

```
                esqueletos[i] = new Esqueleto(cordenadaX, cofre.y, 20,
30, 1);
            }
        }
    }
    public void dibujarEsqueletos(Esqueleto[] esqueletos){
        for (int i = 0; i < esqueletos.length; i++) {
            if(esqueletos[i] !=  null){
                esqueletos[i].dibujarEsqueleto(entorno);
            }
        }
    }
    public void dibujarMonedas(Moneda[] monedas){
        for (int i = 0; i < monedas.length; i++) {
            if(monedas[i] != null){
                monedas[i].dibujarMoneda(entorno);
            }
        }
    }




    @SuppressWarnings("unused")
    public static void main(String[] args)
    {
        Juego juego = new Juego();
    }
}
```

Juego

Fernandez Facundo