UNIVERSIDAD NACIONAL GENERAL SARMIENTO

TRABAJO PRÁCTICO GRUPAL

PROGRAMACIÓN 1

roman.andres.pereyra2020@gmail.com
nicolasbarreto0213@gmail.com
ivanpaez2005@gmail.com

ALUMNOS

BARRETO,
NICOLAS EZEQUIEL
46.431.058
PEREYRA,
ROMÁN ANDRÉS
47.308.457
PAEZ,
IVAN FRANCISCO
46908452

COM-1

Introducción:

En este trabajo, se va a realizar un proyecto en base al lenguaje de programación Java, el cuál constará de implementar funciones, métodos y variables, para cumplir con las expectativas y quede presentado de la mejor manera posible. Este videojuego constará de un mago, que dispondrá de movilidad manual, es decir, que podemos controlarlo nosotros, y será atacado por enemigos (en este caso murciélagos) que se irán acercando a donde se encuentre. Dentro del entorno del juego, dispondremos de poderes, los cuales servirán para aniquilar a estos enemigos. Estos poderes podrán seleccionarse desde el menú. También encontraremos obstáculos, en este caso serán rocas, que impedirán que el jugador atraviese o avance en dicha zona. La partida se da por ganada cuando el jugador consigue matar a todos los enemigos, y se da por perdida, si el jugador no posee más vida.

Desarrollo:

Durante el transcurso de la resolución de este trabajo, se han ido implementando clases, las cuales servirán para que este videojuego funcione de manera adecuada. La primera de ellas, es la clase “Personaje.java”

```
}
public void limitarMovimiento() {
    if (this.x >= 555.0) {
        this.x -= 2;
    }else {
        if(this.y >= 550.0) {
            this.y -= 2;
        }else {
            if (this.x <= 50) {
                this.x += 2;
            }else {
                if (this.y <= 50) {
                    this.y += 2;
                }
```

Allí, encontraremos todo respecto al mago, sus variables que indican su posición, la imagen que lo representa, la vida, la energía. Luego, implementamos tres métodos, uno para dibujarlo en el entorno, otro para desplazarlo vertical y horizontalmente, y por último, uno que limita su movimiento, e impide que este pueda irse por fuera de la pantalla. Después, creamos una clase llamada “Enemigo.java” donde encontraremos todo respecto a los enemigos (también poseen vida, posiciones como el mago) y creamos dos métodos, uno que desplaza al murciélago hacia la posición donde el jugador se encuentre, que esto se consigue, calculando la diferencia entre sus posiciones, luego se calcula la distancia y se le suma un numero aleatorio entre 0 y 1, multiplicado por 0,5, y 0,6, y a su vez también lo multiplicamos al resultado por una desviación que hará que el movimiento sea mucho menos uniforme, y que tarde en desplazarse. Luego un método que determina si un enemigo está muerto, en base a si el poder le quitó la vida suficiente para que este se vuelva null.

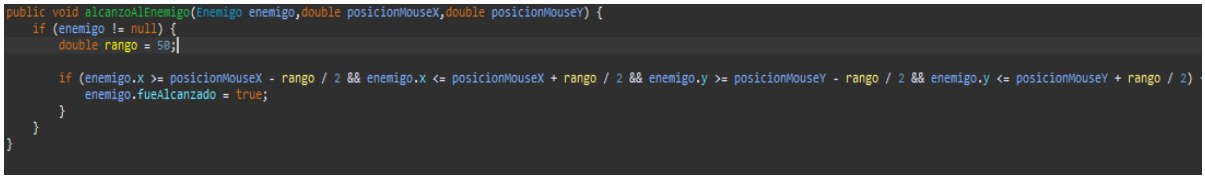
```
}
public void Muerto(Poderes poderes) {
    if (poderes.alcanzoAlEnemigo(null) == true) {
        murcielago = null;
    }
```

En tercer lugar, sigue la clase “Fondo.java” que contiene pocos códigos, lo único que hicimos fue asignarle una imagen a un constructor, y lo dibujamos en el entorno, como el fondo de nuestro juego. Por siguiente, tenemos la clase “Botonera.java”, allí se encuentran todos los botones del menú, el fondo del menú, y algunos detalles extras añadidos simplemente por estética, y luego dibujamos todos esos botones, los cuales tienen asignadas sus respectivas imágenes, en el entorno. A continuación, prosigue la clase “Poderes.java”, donde se encuentran los poderes de nuestro jugador, con su coste de energía, su imagen predeterminada y dos métodos, uno para dibujarlo en el entorno, y otro que en base a si este alcanzo al enemigo, devuelve true, y si no, devuelve false. Una de las últimas clases implementada, es la de “BarrasJugador.java”

```
}
public void bajaLaVida(Personaje personaje, Enemigo enemigo,Entorno e) {
        double dx = personaje.x - enemigo.x;
        double dy = personaje.y - enemigo.y;
        double distancia = Math.sqrt(dx * dx + dy * dy);
        if (distancia <= 1) {
            if (personaje.cooldown == 0 && personaje.vida > 0) {
                personaje.vida -= 1;
                personaje.cooldown = 120;
                }
        }
        e.dibujarRectangulo(700,410, 100, 10, 0, Color.RED);
        e.dibujarRectangulo(700 - (100 - personaje.vida) / 2,410, personaje.vida, 10, 0, Color.GREEN);
            if (personaje.cooldown > 0) {
                personaje.cooldown--;
            }
```

donde se dibujan los rectángulos, que son barras tanto de vida, como la barra de poder, y luego un método que determina si baja la vida, en base a si un enemigo está cerca del mago, o en su misma posición, esta empezará disminuir, y luego, un método que consume la barra de energía si el jugador utiliza un poder. Para culminar, la clase “Juego.java”, aquí se encuentra el núcleo de nuestro programa, y allí inicializamos todas las variables que utilizaremos en el tick, para su respectivo funcionamiento. En el método Juego (), agregamos todas las variables y les asignamos valores si estas lo requerían. Luego, implementamos un método for el cual determina la posición en la que van a aparecer los murciélagos, con varias condiciones que hacen que ellos aparezcan, arriba, debajo, a la izquierda o derecha de la pantalla. Dentro del tick, ya agregando todo lo que haga falta para iniciar el juego, comenzando por el método de colisionar las rocas, que este nos trajo ciertos problemas para resolverlo, ya que el jugador pese a tener las condiciones delimitadas por el ancho, y el alto de la roca, mas bordes y demás, este seguía atravesándolas, ya luego, completando y observando los códigos, logramos corregirlo con éxito. Obviamente, dibujamos en pantalla todas las imágenes para que ya cuando el juego esté iniciado se impriman. Después creamos un método que en base a un booleano “estaSeleccionado”, selecciona el poder y si este se suelta en el entorno del juego, luego se deseleccionará, es decir, pasará a valer false. También, el método seleccionarPoderes() hicimos rectángulos clickeables en el fondo para que, al ser tocados con el botón izquierdo, realicen la función de ser un botón que al ser presionado se seleccione un

poder y luego de esto se lance hacia los enemigos que queramos asesinar. Esto se hizo con e.alcanzoEnemigo()



```
public void alcanzoAlEnemigo(Enemigo enemigo,double posicionMouseX,double posicionMouseY) {
    if (enemigo != null) {
        double rango = 50;

        if (enemigo.x >= posicionMouseX - rango / 2 && enemigo.x <= posicionMouseX + rango / 2 && enemigo.y >= posicionMouseY - rango / 2 && enemigo.y <= posicionMouseY + rango / 2) {
            enemigo.fueAlcanzado = true;
        }
    }
}
```

que toma el poder seleccionado anteriormente para lanzarlos a los enemigos y hacer que estos pasen a estar en estado null. Además, agregamos un menú elaborado con una imagen gif, título del juego y un boton de start que debe presionarse si o si para iniciar el juego (además de que, si le pasamos el cursor por arriba sin clickear, sombrea para dar un detalle más elaborado). Por último, agregamos las respectivas pantallas de game winner (si el mago asesino a 50 enemigos, esto echo a través de un contador) y game over (por si perdes al llegar a la barra de vida en 0). Como extra se agregó una barra de mana o energía, esto hace que para lanzar un poder consumas un poco de esa mana y si esta llegase a 0 no te permitiría seguir lanzando poderes, sino que deberías esperar a que esta se recargue un poco (función agregada que se encarga de cargar de a poco la barra de mana cada vez que se consume un poco después de un tiempo).

Conclusión:

Un trabajo con mucho aprendizaje detrás, además del esfuerzo y el tiempo dedicado. Para poder llevar a cabo este juego de “Gondolf”, tuvimos que analizar todas las funciones que traía el entorno y saber cómo utilizarlas estratégicamente. ¡Muy contentos de que todo ese esfuerzo se haya visto reflejado en un divertido minijuego sobre un mago que debe acabar con todos los murciélagos para sobrevivir!