Con los conocimientos adquiridos durante la unidad vas a implementar un videojuego con una mecánica muy sencilla pero entretenida a la vez.

Partimos de un escenario del tamaño de la ventana. Éstas son las reglas del juego:

1. **El jugador aparece en el centro y hay un grupo robots que lo rodean.**
2. **Los robots explotan cuando colisionan con algo.**
3. **El objetivo del jugador es eliminar a los robots haciéndolos chocar entre sí.**
4. **Los robots se dirigen en línea recta hacia la posición del jugador. Se recomienda (no es obligatorio) que inicialmente avancen muy lentos y vayan acelerando el paso conforme avanza el tiempo.**
5. **El juego acabará cuando algún robot toque al jugador (pierde) o bien todos los robots sean eliminados (gana).**

Tienes que implementar los elementos que le faltan al juego y escribir un breve documento explicativo donde comentes los problemas que te has encontrado y cómo los has solucionado así como cualquier idea que consideres interesante (innovaciones, optimizaciones al código, soluciones originales, etc.) para que tu profesor/a las evalúe. Salvo contraorden del docente, tienes libertad para elegir los assets del juego.

Para la realización de esta tarea se ha utilizado el código ejemplo proporcionado en el ejercicio y se han implantado las modificaciones oportunas para implementar los elementos faltantes:

1- Movimiento de los robots:

Para implementar la velocidad de los robots se ha incluido el siguiente código en el método update:

*float velocidadRobot = 0.01f + (System.currentTimeMillis() - tiempo) * 0.000015f;*

La velocidad de los robots aumenta con el tiempo, lo que hace que el juego sea progresivamente más difícil. Se calcula de la siguiente manera:

- ***0.01f***: Es la velocidad base inicial.
- ***(System.currentTimeMillis() - tiempo) * 0.000015f***: Agrega un factor de aceleración basado en el tiempo transcurrido desde el inicio del juego.

Para crear el movimiento de los robots hacia el jugador se ha incluido el código:

```
for (int i = 0; i < 4; i++) {
    if (robotVivo[i]) {
        float dx = jugadorX - robotX[i];
        float dy = jugadorY - robotY[i];
        float distancia = (float) Math.sqrt(dx * dx + dy * dy);

        // Normaliza el vector de dirección y mueve el robot
        if (distancia > 0) {
            robotX[i] += (dx / distancia) * velocidadRobot * delta;
            robotY[i] += (dy / distancia) * velocidadRobot * delta;
        }
```

- Se calcula la diferencia de posición (dx, dy) entre el robot y el jugador (catetos de un triángulo rectángulo).
- Se obtiene la **distancia** entre ellos usando el teorema de Pitágoras. Calculando la hipotenusa.
- **Normalización del vector de dirección**:
    - Se divide dx y dy por la distancia para obtener un vector de movimiento con longitud 1.
    - Se multiplica por la velocidad del robot y el valor delta (tiempo transcurrido entre fotogramas) para actualizar la posición.

Esto hace que los robots se muevan directamente hacia el jugador.

2- Colisiones entre los robots:

Para configurar las colisiones de los robots entre sí se ha implementado el siguiente código en el método update():

```
// Verificar colisiones con otros robots
for (int j = i + 1; j < 4; j++) {
    if (robotVivo[i] && robotVivo[j]) {
        float distX = robotX[i] - robotX[j];
        float distY = robotY[i] - robotY[j];
        float distanciaEntreRobots = (float) Math.sqrt(distX * distX + distY * distY);
        if (distanciaEntreRobots < 30) { // Ajustar el umbral de colisión
            robotVivo[i] = false;
            robotVivo[j] = false;

            numeroRobotsVivos -= 2;

            // Calculamos la posición de la explosión en el punto medio
            float explosionPosX = (robotX[i] + robotX[j]) / 2;
            float explosionPosY = (robotY[i] + robotY[j]) / 2;

            // Activar la explosión
            mostrarExplosion(x: explosionPosX, y: explosionPosY);
            sonidoExplosion.play();

        }
    }
}
```

- Se comparan todos los robots entre sí para ver si están lo suficientemente cerca (distanciaEntreRobots < 30).
- Si colisionan, ambos robots son eliminados (robotVivo[i] = false; robotVivo[j] = false;).
- Se reduce el contador numeroRobotsVivos en 2.
- Se calcula el punto medio entre los dos robots y se activa la animación de explosión (mostrarExplosion()).
- Se reproduce el sonido de explosión (sonidoExplosion.play();).

3- Colisión del Robot con el Jugador

Para implementar las colisiones de los robots con el jugador se añadido el siguiente código al método update():

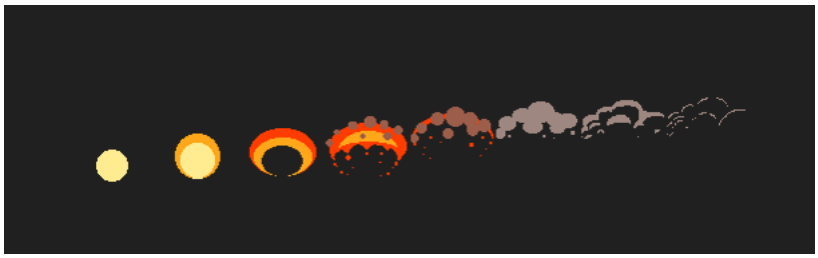
```
// Verificar si un robot toca al jugador
if (robotVivo[i]) {
    float distX = jugadorX - robotX[i];
    float distY = jugadorY - robotY[i];
    if (Math.sqrt(distX * distX + distY * distY) < 32) { // Verifica si el robot colisiona con el jugador
        jugadorVivo = false;
    }

}
```

- Se calcula la distancia entre el robot y el jugador.
- Si es menor a 32, significa que el robot ha alcanzado al jugador.
- Se marca jugadorVivo = false;, lo que indica que el jugador ha perdido.

4- Animación de explosión

Se ha implementado una animación de explosión que se activa cuando dos robots colisionan. La explosión se representa visualmente y se acompaña de un efecto sonoro.

Para gestionar la animación, se ha utilizado la clase Animation de la librería Slick2D, cargando una serie de imágenes secuenciales que simulan la explosión.



Antes de poder mostrar la animación en pantalla, es necesario cargar las imágenes correspondientes y definir la animación en el método init(), que inicializa los recursos del juego. Para ello primero definimos las siguientes variables fuera de la clase init()

```
private Animation explosion;
private float explosionX, explosionY;
private boolean explosionActiva;

private Sound sonidoExplosion;
```

Y en la clase init() creamos el spritesheet (explosion.png), donde cada fotograma tiene un tamaño de 64x64 píxeles.

```
//Incluimos el spritesheet de la explosión:
explosionSheet = new SpriteSheet(ref:"data/explosion.png", tw: 64, th: 64);
```

Para crear la animación se ha creado el método mostrarExplosion():

```
//Método que incluye la animación de la explosión:
private void mostrarExplosion(float x, float y) {
    explosionX = x;
    explosionY = y;
    explosion = new Animation(frames: explosionSheet, duration:100); // 100 ms por frame
    explosion.setLooping(loop:false); //evita que realice la animación en bucle.
    explosion.restart();  // Reinicia la animación desde el primer frame
    explosionActiva = true;
}
```

También incluimos un sonido que reproducirá a la par que la animación. Para ello creamos la variable en la clase principal:

```
private Sound sonidoExplosion;
```

Y instanciamos en el método init():

```
try {
    musicaFondo = new Music(ref:"data/musica de fondo.wav");
    musicaVictoria = new Music(ref:"data/musica victoria.wav");
    sonidoExplosion = new Sound(ref:"data/efecto explosion.wav");

    // Iniciar la música de fondo en bucle
    musicaFondo.loop();
} catch (SlickException e) {
    e.printStackTrace();
}
```

En el método update() se incluye la animación al producirse la colisión entre dos robots y se reproduce el sonido de la explosión:

```
// Verificar colisiones con otros robots
for (int j = i + 1; j < 4; j++) {
    if (robotVivo[i] && robotVivo[j]) {
        float distX = robotX[i] - robotX[j];
        float distY = robotY[i] - robotY[j];
        float distanciaEntreRobots = (float) Math.sqrt(distX * distX + distY * distY);
        if (distanciaEntreRobots < 30) { // Ajustar el umbral de colisión
            robotVivo[i] = false;
            robotVivo[j] = false;

            numeroRobotsVivos -= 2;

            // Calculamos la posición de la explosión en el punto medio
            float explosionPosX = (robotX[i] + robotX[j]) / 2;
            float explosionPosY = (robotY[i] + robotY[j]) / 2;

            // Activar la explosión
            mostrarExplosion(x: explosionPosX, y: explosionPosY);
            sonidoExplosion.play();

        }
    }
}
```

Para que la animación sea visible en pantalla, se dibuja en el método render(), verificando si la explosión está activa.

```
// Dibujar la explosión si está activa
if (explosionActiva) {
    explosion.draw(x: explosionX, y: explosionY);

    // Desactivar la explosión cuando termine la animación
    if (explosion.isStopped()) {
        explosionActiva = false;
    }
}
```

- Si explosionActiva es true, se dibuja la animación en la posición de la colisión.
- Se verifica si la animación ha terminado (isStopped()), y en ese caso, se desactiva la explosión (explosionActiva = false;).

5- Música de fondo y de victoria

El juego implementado utiliza música de fondo y de victoria para mejorar la experiencia del usuario. Se han añadido dos pistas de audio principales:

- **Música de fondo**, que se reproduce en bucle mientras el juego está en ejecución.
- **Música de victoria**, que se activa cuando el jugador elimina a todos los robots enemigos.

En la clase principal declaramos las variables:

```
private Music musicaFondo;
private Music musicaVictoria;
```

La música se gestiona utilizando las clases Music. En el método init(GameContainer container), se cargan los archivos de audio y se inicia en modo loop (bucle) la música de fondo que sonará durante la partida:

```
try {
    musicaFondo = new Music(ref:"data/musica de fondo.wav");
    musicaVictoria = new Music(ref:"data/musica victoria.wav");
    sonidoExplosion = new Sound(ref:"data/efecto explosion.wav");

    // Iniciar la música de fondo en bucle
    musicaFondo.loop();
} catch (SlickException e) {
    e.printStackTrace();
}
```

- Se instancian los objetos Music y Sound con los archivos de audio en la carpeta data.
- musicaFondo.loop(); inicia la reproducción de la música de fondo en bucle desde el inicio del juego.
- try-catch se utiliza para capturar cualquier posible error al cargar los archivos de sonido.

Cuando el jugador ha eliminado a todos los robots, se detiene la música de fondo y se inicia la música de victoria. Para ello incluimos en el método update esta condición, dentro del bucle que gestiona el movimiento y las colisiones de los robots:

```
if (numeroRobotsVivos == 0 && musicaVictoria.playing() == false) {
    musicaFondo.stop();
    musicaVictoria.play();
}
```

- Se verifica si el número de robots vivos es 0 (numeroRobotsVivos == 0).
- Se comprueba que la música de victoria no esté ya reproduciéndose (musicaVictoria.playing() == false).
- Se detiene la música de fondo (musicaFondo.stop();).
- Se reproduce la música de victoria (musicaVictoria.play();).