Guía de Implementación: Power-Up Botas y Sistema de Audio

Esta guía describe cómo implementar dos características en el proyecto **PrimerJuego2D**:

1. **Power-Up "Botas"**: Un objeto que aumenta la velocidad del jugador.
2. **Sistema de Audio**: Un motor de sonido para reproducir música de fondo y efectos de sonido.

📋 Tabla de Contenidos

Resumen General

Objetivo

- **Mecánica de juego**: Añadir un "power-up" (botas) que modifique los atributos del jugador (velocidad).
- **Motor de Audio**: Implementar una clase utilitaria nativa de Java (`javax.sound.sampled`) para manejar música en bucle y efectos de sonido puntuales.

Estructura Actual del Proyecto

```
PrimerJuego2D/
├── src/
│   ├── entidad/
│   │   ├── Entidad.java          # Clase base de entidades
```

```
│   │   └── Jugador.java           # Lógica del jugador
│   ├── main/
│   │   ├── AssetSetter.java       # Coloca objetos en el mapa
│   │   ├── PanelJuego.java        # Panel principal del juego
│   │   └── ...
│   └── objetos/
│       ├── superObjeto.java       # Clase padre de objetos
│       ├── OBJ_llave.java
│       ├── OBJ_puerta.java
│       └── OBJ_cofre.java
└── res/
    ├── objetos/
    │   ├── llave.png
    │   ├── puerta.png
    │   ├── cofre.png
    │   └── zapato.png             # ¡Ya existe la imagen!
    └── sound/                     # A CREAR: carpeta de audio
```

Parte A: Implementación del Power-Up Botas

Paso A1: Crear la Clase `OBJ_botas.java`

> [!NOTE] La imagen `zapato.png` ya existe en `/res/objetos/`. Solo crea la clase Java.

Crear archivo: `src/objetos/OBJ_botas.java`

```java
package objetos;

import java.io.IOException;
import javax.imageio.ImageIO;

/**
 * Objeto coleccionable que aumenta la velocidad del jugador.
 */
public class OBJ_botas extends superObjeto {

    public OBJ_botas() {
        nombre = "botas";
        try {
```

```
            imagen = ImageIO.read(getClass().getResource("/
        objetos/zapato.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Paso A2: Registrar las Botas en `AssetSetter.java`

Editar archivo: AssetSetter.java

1. Añadir el import al inicio del archivo:

```
 import objetos.OBJ_llave;
 import objetos.OBJ_puerta;
 import objetos.OBJ_cofre;
+import objetos.OBJ_botas;
```

1. Necesitas aumentar el tamaño del array en `PanelJuego.java`:

```
-public superObjeto[] objs = new superObjeto[10];
+public superObjeto[] objs = new superObjeto[15];
```

1. Añadir las botas en el método `setObjetct()` en `AssetSetter.java`:

```
// Power-Up: Botas de velocidad
pj.objs[10] = new OBJ_botas();
pj.objs[10].worldX = 25 * pj.tamanioTile;  // Ajusta las
        coordenadas
pj.objs[10].worldY = 15 * pj.tamanioTile;  // según tu mapa
```

Paso A3: Añadir la Lógica de Recolección en `Jugador.java`

Editar archivo: Jugador.java

En el método `recogerObjeto(int index)`, añadir un nuevo `case` en el `switch`:

```
case "botas":
    vel += 2;  // Aumenta la velocidad del jugador
    pj.objs[index] = null;  // Elimina el objeto del mapa
    System.out.println("¡Botas de velocidad! Velocidad actual: "
        + vel);
    break;
```

Ubicación: Después del `case "cofre":` y antes del cierre del `switch`.

Parte B: Sistema de Audio

Paso B1: Crear Carpeta de Recursos de Sonido

Crear la carpeta `res/sound/` y colocar archivos `.wav`:

```
res/sound/
├── musica_fondo.wav      # Índice 0: Música de fondo (loop)
├── recoger_llave.wav     # Índice 1: Efecto al recoger llave
├── recoger_powerup.wav   # Índice 2: Efecto al recoger power-up
├── abrir_puerta.wav      # Índice 3: Efecto al abrir puerta
└── abrir_cofre.wav       # Índice 4: Efecto al abrir cofre
```

> [!WARNING] **Java estándar solo soporta archivos `.wav`**. Si tienes MP3, debes convertirlos a WAV. Herramienta recomendada: Audacity o usar `ffmpeg`:
>
> ```
> ffmpeg -i musica.mp3 -acodec pcm_s16le -ar 44100
>         musica_fondo.wav
> ```

Paso B2: Crear la Clase `Sound.java`

Crear archivo: `src/main/Sound.java`

```java
package main;

import java.net.URL;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

/**
 * Gestor de audio del juego.
 * Maneja la reproducción de música de fondo y efectos de sonido.
 */
public class Sound {

    Clip clip;
    URL[] soundURL = new URL[30];  // Array para almacenar rutas
        de audio

    /**
```

```java
     * Constructor: Inicializa las rutas de los archivos de
        sonido.
     */
    public Sound() {
        // Índice 0: Música de fondo
        soundURL[0] = getClass().getResource("/sound/
        musica_fondo.wav");

        // Índice 1: Efecto de recoger llave
        soundURL[1] = getClass().getResource("/sound/
        recoger_llave.wav");

        // Índice 2: Efecto de recoger power-up (botas)
        soundURL[2] = getClass().getResource("/sound/
        recoger_powerup.wav");

        // Índice 3: Efecto de abrir puerta
        soundURL[3] = getClass().getResource("/sound/
        abrir_puerta.wav");

        // Índice 4: Efecto de abrir cofre
        soundURL[4] = getClass().getResource("/sound/
        abrir_cofre.wav");
    }

    /**
     * Carga un archivo de audio en memoria.
     * @param i - Índice del sonido en el array soundURL.
     */
    public void setFile(int i) {
        try {
            AudioInputStream ais =
        AudioSystem.getAudioInputStream(soundURL[i]);
            clip = AudioSystem.getClip();
            clip.open(ais);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Inicia la reproducción del clip cargado.
     */
    public void play() {
        clip.start();
    }
```

```
    /**
     * Reproduce el clip en bucle infinito (para música de fondo).
     */
    public void loop() {
        clip.loop(Clip.LOOP_CONTINUOUSLY);
    }

    /**
     * Detiene la reproducción del clip.
     */
    public void stop() {
        clip.stop();
    }
}
```

Paso B3: Integrar el Sistema de Audio en `PanelJuego.java`

Editar archivo: PanelJuego.java

3.1 Añadir atributos de sonido

Después de la declaración del `AssetSetter`:

```
// Sistema de sonido
Sound musica = new Sound();
Sound efectoSonido = new Sound();
```

3.2 Añadir métodos wrapper para el audio

Antes del cierre de la clase, añadir estos métodos:

```
/**
 * Reproduce música de fondo en bucle.
 * @param i - Índice del archivo de música en Sound.soundURL[]
 */
public void playMusic(int i) {
    musica.setFile(i);
    musica.play();
    musica.loop();
}

/**
 * Detiene la música de fondo.
 */
```

```
public void stopMusic() {
    musica.stop();
}

/**
 * Reproduce un efecto de sonido puntual.
 * @param i - Índice del efecto de sonido en Sound.soundURL[]
 */
public void playSE(int i) {
    efectoSonido.setFile(i);
    efectoSonido.play();
}
```

3.3 Iniciar la música en `setupJuego()`

Modificar el método `setupJuego()`:

```
public void setupJuego() {
    aSetter.setObjetct();
    playMusic(0);  // Inicia la música de fondo
}
```

Paso B4: Disparar Efectos de Sonido en `Jugador.java`

Editar archivo: Jugador.java

Modificar el método `recogerObjeto()` para reproducir sonidos:

```
public void recogerObjeto(int index) {
    if (index != 999) {
        String nombreObjeto = pj.objs[index].nombre;
        switch (nombreObjeto) {
        case "llave":
            pj.playSE(1);  // 🔊 Sonido de llave
            numeroLlaves++;
            pj.objs[index] = null;
            System.out.println("Llaves: " + numeroLlaves);
            break;
        case "puerta":
            if (numeroLlaves > 0) {
                pj.playSE(3);  // 🔊 Sonido de puerta
                pj.objs[index] = null;
                numeroLlaves--;
                System.out.println("¡Puerta abierta! Llaves
        restantes: " + numeroLlaves);
```

```
            } else {

            System.out.println("Necesitas una llave para abrir esta
            puerta");
            }
            break;
        case "cofre":
            pj.playSE(4);  // 🔊 Sonido de cofre
            pj.objs[index] = null;
            System.out.println("¡Cofre abierto!");
            break;
        case "botas":
            pj.playSE(2);  // 🔊 Sonido de power-up
            vel += 2;
            pj.objs[index] = null;
            System.out.println("¡Botas de velocidad! Velocidad
            actual: " + vel);
            break;
        }
    }
}
```

Verificación

Checklist de Archivos

- [ ] `src/objetos/OBJ_botas.java` - Nueva clase creada
- [ ] `src/main/Sound.java` - Nueva clase creada
- [ ] `src/main/AssetSetter.java` - Import y posición de botas añadidos
- [ ] `src/main/PanelJuego.java` - Array ampliado + Sistema de audio integrado
- [ ] `src/entidad/Jugador.java` - Lógica de botas y sonidos añadida
- [ ] `res/sound/` - Carpeta con archivos `.wav`

Pruebas

1. **Compilar sin errores**: Ejecutar el proyecto y verificar que compila.

2. **Botas funcionan**: Caminar hacia las botas y verificar:
   - El objeto desaparece al tocarlo
   - La velocidad del jugador aumenta (verificar en consola)
   - Se reproduce el efecto de sonido
3. **Música de fondo**: Al iniciar el juego, debe sonar la música.
4. **Efectos de sonido**: Recoger llaves, abrir puertas y cofres debe producir sonidos.

📝 Notas Adicionales

> [!IMPORTANT] **Limitaciones de `javax.sound.sampled`**: - Solo soporta formatos **WAV**, **AU** y **AIFF**. - Para MP3/OGG, necesitarías librerías externas como **JLayer** o **JavaFX Media**.

> [!TIP] **Consejo de rendimiento**: Si planeas tener muchos efectos de sonido, considera pre-cargar los clips en el constructor de `Sound` para evitar latencia durante el juego.

🗂️ Resumen de Cambios por Archivo

Archivo	Acción	Descripción
`OBJ_botas.java`	**NUEVO**	Clase del power-up de velocidad
`Sound.java`	**NUEVO**	Gestor de audio del juego
`AssetSetter.java`	Modificar	Añadir import y posición de botas
`PanelJuego.java`	Modificar	Ampliar array, añadir sistema de audio
`Jugador.java`	Modificar	Añadir lógica de botas y sonidos
`res/sound/`	**NUEVO**	Carpeta con archivos de audio `.wav`