🎮 Episodio 10: Implementación de UI (Interfaz de Usuario)

> **Final de la primera etapa: “Treasure Hunting”**
> Video de referencia: Episodio 10 - UI

📋 Resumen

Este episodio marca el final de la primera etapa del juego. El objetivo es dar **feedback visual al jugador** mostrando: - Contador de llaves recolectadas - Mensajes temporales (ej: “¡Conseguiste una llave!”) - Tiempo de juego - Pantalla de “Juego Terminado”

🗂️ Archivos que se modifican

Archivo	Acción	Descripción
`UI.java`	**NUEVO**	Clase que maneja todo el HUD (dibuja sobre el juego)
`PanelJuego.java`	Modificar	Instanciar UI y llamar a `ui.draw()`
`Jugador.java`	Modificar	Llamar a métodos de UI para mostrar mensajes
`Sound.java`	Ya corregido	Tu proyecto ya tiene el fix de dos instancias

---

🔧 Estado Actual de tu Proyecto

✅ Ya implementado correctamente:

Tu proyecto **ya tiene la corrección del sistema de sonido**. En PanelJuego.java tienes:

```
// Sistema de sonido
Sound musica = new Sound();         // Para música de fondo (loop)
Sound efectoSonido = new Sound();   // Para efectos cortos (SE)
```

¿Por qué dos instancias?
Imagina que tienes un reproductor de música. Si intentas pausar una canción y al mismo tiempo reproducir un sonido de "coin", se genera un conflicto. Con dos reproductores separados, cada uno hace su trabajo sin interferir.

🎯 Implementación Paso a Paso

Paso 1: Crear la Clase `UI.java`

Crea el archivo `src/main/UI.java`:

```
package main;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.text.DecimalFormat;

import objetos.OBJ_llave;

/**
 * Clase que maneja la Interfaz de Usuario (HUD).
 * Dibuja información sobre el juego: llaves, tiempo, mensajes.
 */
public class UI {

    PanelJuego pj;

    // ⚠️ IMPORTANTE: Las fuentes se declaran AQUÍ, no dentro de
        draw()
```

```java
// Esto es por rendimiento - draw() se ejecuta 60 veces por
    segundo
Font arial_40;
Font arial_80B;  // B = Bold

// Imagen de la llave para el HUD
BufferedImage imagenLlave;

// Sistema de mensajes temporales
public boolean mensajeActivo = false;
public String mensaje = "";
int contadorMensaje = 0;

// Estado del juego
public boolean juegoTerminado = false;

// Tiempo de juego
double tiempoJuego;
DecimalFormat formatoDecimal = new DecimalFormat("#0.00");

/**
 * Constructor: Inicializa fuentes y carga recursos.
 */
public UI(PanelJuego pj) {
    this.pj = pj;

    // Crear fuentes UNA sola vez (optimización)
    arial_40 = new Font("Arial", Font.PLAIN, 40);
    arial_80B = new Font("Arial", Font.BOLD, 80);

    // Cargar imagen de llave para mostrar en el HUD
    OBJ_llave llave = new OBJ_llave();
    imagenLlave = llave.imagen;
}

/**
 * Muestra un mensaje temporal en pantalla.
 * El mensaje desaparece después de ~2 segundos.
 */
public void mostrarMensaje(String texto) {
    mensaje = texto;
    mensajeActivo = true;
}
```

```java
/**
 * Método principal de dibujado del HUD.
 * Se llama desde paintComponent() de PanelJuego.
 */
public void draw(Graphics2D g2) {

    if (juegoTerminado == true) {
        // Pantalla de victoria
        dibujarPantallaFin(g2);
    } else {
        // HUD normal durante el juego
        dibujarHUD(g2);
    }
}

/**
 * Dibuja el HUD normal: llaves, tiempo, mensajes.
 */
private void dibujarHUD(Graphics2D g2) {

    g2.setFont(arial_40);
    g2.setColor(Color.WHITE);

    // === CONTADOR DE LLAVES ===
    // Dibujar icono de llave
    g2.drawImage(imagenLlave, pj.tamanioTile / 2,
    pj.tamanioTile / 2,
                pj.tamanioTile, pj.tamanioTile, null);

    // Dibujar número de llaves (x2, x3, etc.)
    g2.drawString("x " + pj.jugador.numeroLlaves, 74, 65);

    // === TIEMPO DE JUEGO ===
    tiempoJuego += (double) 1 / 60; // Incrementar cada frame
    g2.drawString("Tiempo: " +
    formatoDecimal.format(tiempoJuego),
                 pj.tamanioTile * 11, 65);

    // === MENSAJES TEMPORALES ===
    if (mensajeActivo == true) {
        g2.setFont(g2.getFont().deriveFont(30F));
        g2.drawString(mensaje, pj.tamanioTile / 2,
    pj.tamanioTile * 5);

        contadorMensaje++;
```

```
            // Después de 120 frames (~2 segundos), ocultar
        mensaje
            if (contadorMensaje > 120) {
                contadorMensaje = 0;
                mensajeActivo = false;
            }
        }
    }

    /**
     * Dibuja la pantalla de fin del juego.
     */
    private void dibujarPantallaFin(Graphics2D g2) {

        g2.setFont(arial_40);
        g2.setColor(Color.WHITE);

        String texto;
        int x, y;
        int longitudTexto;

        // === MENSAJE DE FELICITACIONES ===
        texto = "¡Encontraste el tesoro!";
        g2.setFont(arial_80B);

        // Centrar texto horizontalmente
        longitudTexto = (int)
        g2.getFontMetrics().getStringBounds(texto, g2).getWidth();
        x = (pj.anchoPantalla / 2) - (longitudTexto / 2);
        y = pj.altoPantalla / 2 - (pj.tamanioTile * 2);

        // Sombra del texto (efecto visual)
        g2.setColor(Color.BLACK);
        g2.drawString(texto, x + 5, y + 5);

        // Texto principal
        g2.setColor(Color.YELLOW);
        g2.drawString(texto, x, y);

        // === TIEMPO FINAL ===
        g2.setFont(arial_40);
        g2.setColor(Color.WHITE);
        texto = "Tu tiempo fue: " +
        formatoDecimal.format(tiempoJuego) + " segundos";
```

```
            longitudTexto = (int)
            g2.getFontMetrics().getStringBounds(texto, g2).getWidth();
            x = (pj.anchoPantalla / 2) - (longitudTexto / 2);
            y = pj.altoPantalla / 2 + (pj.tamanioTile * 2);
            g2.drawString(texto, x, y);

            // === DETENER EL JUEGO ===
            pj.threadJuego = null;
        }

        /**
         * Método utilitario para obtener la
            posición X centrada de un texto.
         */
        public int obtenerXCentrado(String texto, Graphics2D g2) {
            int longitudTexto = (int)
            g2.getFontMetrics().getStringBounds(texto, g2).getWidth();
            return (pj.anchoPantalla / 2) - (longitudTexto / 2);
        }
    }
```

Paso 2: Modificar `PanelJuego.java`

2.1 Agregar la instancia de UI

Después de las variables de sonido, agrega:

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

// Interfaz de Usuario (HUD)
public UI ui = new UI(this);
```

2.2 Llamar a `ui.draw()` en paintComponent

El orden es **MUY IMPORTANTE**. La UI debe dibujarse **al final** para que aparezca encima de todo:

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;

    // 1. Tiles (fondo)
    tileManager.draw(g2);
```

```
        // 2. Objetos (llaves, puertas, cofres)
        for (int i = 0; i < objs.length; i++) {
            if (objs[i] != null) {
                objs[i].draw(g2, this);
            }
        }

        // 3. Jugador
        jugador.draw(g2);

        // 4. UI (HUD) - ¡SIEMPRE AL FINAL!
        ui.draw(g2);

        g2.dispose();
    }
```

Paso 3: Modificar `Jugador.java`

En el método `recogerObjeto()`, agrega llamadas a `ui.mostrarMensaje()`:

```
public void recogerObjeto(int index) {
    if (index != 999) {
        String nombreObjeto = pj.objs[index].nombre;

        switch (nombreObjeto) {
        case "llave":
            pj.playSE(1);
            numeroLlaves++;
            pj.objs[index] = null;
            pj.ui.mostrarMensaje("¡Conseguiste una
        llave!");  // ← AGREGAR
            break;

        case "puerta":
            if (numeroLlaves > 0) {
                pj.playSE(3);
                pj.objs[index] = null;
                numeroLlaves--;
            } else {
                pj.ui.mostrarMensaje("¡Necesitas una
        llave!");  // ← AGREGAR
            }
            break;
```

```
            case "cofre":
                pj.playSE(4);
                pj.objs[index] = null;
                pj.ui.juegoTerminado = true;  // ← AGREGAR
                pj.stopMusic();                // ← AGREGAR
                break;

            case "botas":
                pj.playSE(2);
                vel += 4;
                pj.objs[index] = null;
                pj.ui.mostrarMensaje("¡Velocidad aumentada!");  // ← AGREGAR
                break;
            }
        }
}
```

🧠 Cómo Funciona la Lógica (Explicación Simple)

🖼️ El Orden de Dibujo (Capas)

Imagina que estás pintando un cuadro en capas:

```
┌─────────────────────────────────┐
│  4. UI (HUD) - Encima de todo   │  ← Llaves: x3, Tiempo: 45.32s
├─────────────────────────────────┤
│  3. Jugador                     │  ← Tu personaje
├─────────────────────────────────┤
│  2. Objetos                     │  ← Llaves, puertas, cofres
├─────────────────────────────────┤
│  1. Tiles (Mapa)                │  ← Pasto, agua, árboles
└─────────────────────────────────┘
```

Si dibujas el HUD **antes** que el jugador, el jugador se pintaría encima y no verías la UI.

⏱️ El Sistema de Mensajes Temporales

Funciona como un temporizador de cocina:

```
[Jugador toca llave]
     ↓
mostrarMensaje("¡Conseguiste una llave!")
     ↓
mensajeActivo = TRUE
mensaje = "¡Conseguiste una llave!"
     ↓
[Cada frame en draw()]
     │
     ├→ ¿mensajeActivo? SÍ → Dibujar mensaje
     │                     → contadorMensaje++ (0, 1, 2... 119,
120)
     │
     └→ ¿contadorMensaje > 120? SÍ → mensajeActivo = FALSE
                                   → contadorMensaje = 0
```

¿Por qué 120? Porque el juego corre a 60 FPS. 120 frames = 2 segundos.

🎯 Centrar Texto en Pantalla

El problema con `g2.drawString()` es que dibuja desde la esquina **izquierda** del texto:

```
Pantalla: [──────────────────────────]
                    ↓
Texto normal:     [Hola]──────────── (X = pantalla/2)
                    ↑
             Texto empieza aquí, NO está centrado

Texto centrado:      [Hola]
                       ↑
                Centro real del texto
```

La fórmula mágica:

```
// 1. Obtener el ancho del texto en píxeles
int anchoTexto = g2.getFontMetrics().getStringBounds(texto,
        g2).getWidth();
```

```java
// 2. Calcular X para centrar
int x = (anchoPantalla / 2) - (anchoTexto / 2);
```

Es como colgar un cuadro: mides el cuadro, mides la pared, y pones el clavo en el punto medio.

⏰ El Contador de Tiempo

```java
double tiempoJuego;  // Empieza en 0.0

// Cada frame (60 veces por segundo):
tiempoJuego += (double) 1 / 60;  // Suma ~0.0167 segundos

// Frame 1:  tiempoJuego = 0.0167
// Frame 60: tiempoJuego = 1.0 (1 segundo)
// Frame 120: tiempoJuego = 2.0 (2 segundos)
```

DecimalFormat evita mostrar números feos como `45.3333333`:

```java
DecimalFormat df = new DecimalFormat("#0.00");
df.format(45.3333333);  // Resultado: "45.33"
```

📦 Clases Java Utilizadas

Clase	Uso
`Graphics2D`	Dibujar texto, imágenes, formas
`Font`	Definir tipografía (Arial, 40px, Bold, etc.)
`FontMetrics`	Medir el ancho/alto del texto
`DecimalFormat`	Formatear números decimales
`Color`	Definir colores (WHITE, YELLOW, BLACK)

⚠️ Errores Comunes

❌ Crear Font dentro de draw()

```java
// MAL - Se crea 60 veces por segundo
public void draw(Graphics2D g2) {
    Font fuente = new Font("Arial", Font.PLAIN, 40);  // ❌
```

```
    g2.setFont(fuente);
}

// BIEN - Se crea UNA vez en el constructor
Font fuente;  // Variable de clase

public UI(PanelJuego pj) {
    fuente = new Font("Arial", Font.PLAIN, 40);  // ✅
}

public void draw(Graphics2D g2) {
    g2.setFont(fuente);  // Solo usa la referencia
}
```

❌ Dibujar UI antes que el jugador

```
// MAL - El jugador se pinta encima del HUD
ui.draw(g2);
jugador.draw(g2);

// BIEN - El HUD se ve encima del jugador
jugador.draw(g2);
ui.draw(g2);
```

🎬 Resultado Final

Después de implementar todo:

1. **Esquina superior izquierda**: Icono de llave + contador (x0, x1, x2...)
2. **Esquina superior derecha**: Tiempo transcurrido (Tiempo: 45.32)
3. **Centro de pantalla**: Mensajes temporales al recoger objetos
4. **Al tocar el cofre**: Pantalla de victoria + tiempo final + juego se detiene

---

🔗 Timestamps del Video

- 00:33 - Corrección del sistema de sonido
- 01:53 - Creación de la clase UI
- 04:08 - Optimización de rendimiento (Font)
- 11:33 - Sistema de mensajes temporales

- 18:32 - Pantalla de fin del juego
- 22:45 - Centrado de texto
- 27:48 - Contador de tiempo
- 29:58 - DecimalFormat

> **Siguiente paso**: Episodio 11 - Estados del Juego (Menú, Pausa, etc.)