La Batalla de Gritos

Demuestra tu maestría en programación con este desafío final. Gestiona estados complejos para crear un 'gritómetro' que mida la duración de tus gritos.

Cursos

- Grados 9-12

Materiales

- Celular y computador
- Conexión a Internet
- Cartón, tijeras y un par de elásticos para sujetar el celular al cartón

Descripción

Este es el proyecto final y más complejo, diseñado para que los estudiantes sinteticen todo lo aprendido. Deberán construir una "máquina de estados" funcional para un juego, gestionando las transiciones entre los estados "esperando", "contando" y "terminado". Esto requiere una comprensión sólida de cómo los eventos, los bucles, las variables y los condicionales anidados trabajan juntos para crear una aplicación interactiva robusta.

Objetivos Educativos

- Sintetizar todos los conceptos de programación aprendidos en un único proyecto complejo.
- Diseñar e implementar un programa con múltiples estados y transiciones lógicas (una máquina de estados).
- Gestionar el flujo del programa utilizando una combinación sinérgica de lógica dirigida por eventos y por bucles.
- Depurar y resolver problemas en un programa con una lógica de estado compleja.

Inicio (10 minutos) - La Máquina de Estados

1. Da la bienvenida a la clase: **"Han llegado a la cima de la montaña. Hoy construirán su proyecto más inteligente hasta la fecha: un juego que mide la duración de sus gritos."**

2. Introduce el concepto final: **"Piensen en un videojuego. Un personaje puede estar 'explorando', 'en batalla' o 'en un menú'. Cada uno es un 'estado' diferente. Nuestro juego de hoy también tiene estados: 'listo para empezar', 'contando el tiempo' y 'mostrando el resultado'. Aprender a gestionar estos estados es la clave de la programación avanzada."**
3. Plantea el desafío arquitectónico: **"¿Cómo lo haremos? Necesitaremos una combinación de todo: un evento para empezar, un bucle para contar y variables para recordar en qué estado estamos."**

Máquinas de Estado: Los "Modos" de un Programa

Casi todos los programas complejos pueden describirse como una **máquina de estado**. Esto significa que el programa puede estar en diferentes "modos" o "estados", y existen reglas claras para pasar de un estado a otro. Nuestro gritómetro tiene tres estados principales: 1. **Estado de Espera:** El juego está listo para empezar, solo necesita la señal de partida. La variable `partidaEnCurso` es falsa. 2. **Estado de Conteo:** La barra de LEDs va avanzando. La variable `partidaEnCurso` es verdadera. 3. **Estado Terminado (Cooldown):** El juego ha terminado y muestra el resultado con música durante 5 segundos. Durante este tiempo, `partidaEnCurso` permanece verdadero, por lo que no se puede iniciar una nueva partida. Después de los 5 segundos, `partidaEnCurso` vuelve a ser falso y el juego queda listo para comenzar nuevamente. Ojo: un grito dado durante este enfriamiento se pierde sin aviso — hay que esperar a que la pantalla se apague y volver a gritar.

La Sinergia de Eventos y Bucles

En proyectos complejos, a menudo no se usa *solo* un evento o *solo* un bucle. Se utiliza una combinación inteligente de ambos, donde cada uno hace lo que mejor sabe hacer: * **Eventos:** Son perfectos para **iniciar** acciones o cambiar de estado de forma instantánea. En nuestro caso, un evento de ruido fuerte es el pistoletazo de salida que inicia la partida. * **Bucles:** Son perfectos para **gestionar** procesos continuos que dependen de un estado. Nuestro bucle principal no inicia el juego. Cuando partidaEnCurso es verdadero, se encarga de mantener el juego activo mientras el nivel de ruido sea mayor o igual a 5.

Cuando el ruido baja de ese valor, el bucle termina y el programa pasa a la etapa final de la partida.

La Lógica de Nuestra Batalla

Nuestra arquitectura combina todo lo aprendido: * Usamos una variable booleana, `partidaEnCurso`, para controlar si se puede empezar una nueva partida (gestiona el estado de **Espera** y **Cooldown**). * Comprobamos si inició el juego, y mientras el nivel de ruido sea mayor o igual a 5, la barra sigue avanzando **un LED cada medio segundo** (gestiona el estado de **Conteo** para encender la pantalla LED). * Un **evento** de ruido fuerte pone en marcha el juego activando `partidaEnCurso`. El evento se dispara siempre; es el **bucle principal** el que solo le hace caso cuando está en **Espera**. * El **bucle principal** se encarga del resto: actualiza la pantalla si estamos **Contando** y cuando el ruido se detiene pasa al estado **Terminado**.

Desarrollo (20-30 minutos) - Construyendo el Gritómetro

1. Este es el proyecto más complejo. Anima a los estudiantes a construirlo paso a paso y a probarlo con frecuencia.
2. Guíalos a través de **la lógica de gestión de estado, explicando el rol de cada variable y cómo el evento y el bucle colaboran**, como se detalla a continuación.

Cierre (5-10 minutos) - ¡Te has Graduado!

1. Una vez que los gritómetros estén funcionando y los estudiantes estén compitiendo, celebra su logro.
2. Haz una reflexión final: **"Miren el código que han construido. Tiene eventos, bucles, variables, condicionales... lo tiene todo. Describan cómo funciona, cómo pasa de un estado a otro. ¡Felicidades, han programado una máquina de estados completa y han completado este viaje de introducción al pensamiento computacional!"**

Reflexiona

Este programa gestiona varios "estados". ¿Puedes nombrar los diferentes estados y qué variables los controlan?

Describe el propósito del evento y el propósito del bucle principal. ¿Por qué se necesitan ambos para que este juego funcione correctamente?

Si quisieras que el juego fuera más difícil (que haya que gritar más tiempo para que se enciendan los LEDS), ¿qué valor en el código cambiarías?