Los minijuegos en JS

Los minijuegos en JS

La tarea evaluable de este tema consistirá en la creación y ejecución de varios mini-juegos en JavaScript, que desarrollaremos de manera progresiva durante las clases.

Estos proyectos podrán formar parte de vuestro portafolio personal, por lo que es importante que el código esté bien organizado, sea claro y estructurado. Además, os animo a aplicar estilos visuales atractivos para que vuestro trabajo destaque y sea valorado en futuros contextos profesionales.

Primer Minijuego (Momory)

Parte 1: Tu tarea es generar una cuadrícula de cartas en el contenedor   que está presente en el HTML.

- En el body del HTML debe aparecer un div con una id=”juego”. Y todo el desarrollo del juego se generará desde el script.
- En el js generaremos una cuadrícula de cartas (por ejemplo, 4x4 o 5x5) de forma dinámica usando JavaScript.
- Cada carta debe tener un diseño inicial uniforme, un color uniforme con la palabra carta, o incluso la trasera de una carta de juego.
- Se trata de un juego de parejas por lo que no podremos identificarlas con un id (puesto que los id deben de ser únicos en cada html). De forma oculta cada carta deberá tener la información de una imagen y un atributo (name por ejemplo) ambos elementos pertenecerán a dos cartas.
- Usa arrays para representar las cartas y sus valores.
- Mezcla los valores aleatoriamente antes de asignarlos a las cartas para que cada partida sea diferente.

Resultado:

- Como resultado de este paso deberemos tener una cuadrícula generada en js con un numero par de cartas que visiblemente tengan la misma apariencia pero que ocultamente tengan valores diferentes.
- Crea una apariencia bonita en en el html para invitar al usuario a participar.

Primer Minijuego (Momory)

Parte 2: Ahora que tienes la cuadrícula de cartas, tu tarea es darles interactividad. Sigue estas instrucciones:

- **Eventos de clic:** Haz que al hacer clic en una carta, esta se voltee y muestre su contenido oculto, y oculta su contenido visible.
- **Restricciones:** Los jugadores solo pueden voltear dos cartas a la vez. Si se voltea una tercera, esta no debe reaccionar hasta que se resuelvan las dos primeras.
- **Lógica de coincidencia:** Si las dos cartas volteadas coinciden, deben permanecer descubiertas. Si no coincide, deben volver a voltearse después de un breve retraso (por ejemplo, 1 segundo).
- **Estilo:**Trabaja algún efecto CSS para voltear las cartas, transiciones, animaciones....

Resultado:

- Las cartas pueden voltearse al hacer clic.
- Si dos cartas coinciden, quedarán descubiertas. Si no es coincidencia, se vuelve a ocultar.

Primer Minijuego (Momory)

Parte 3: Con la cuadrícula y las listas de interactividad, tu tarea final es gestionar el estado del juego y añadir funcionalidades finales. Sigue estas instrucciones:

- **Verificar la victoria:** Agrega una comprobación para determinar si el jugador ha ganado, es decir, si todas las cartas están descubiertas. Al ganar, muestra un mensaje de felicitación (por ejemplo, "¡Has ganado!") y ofrece la posibilidad de reiniciar el juego.
- **Reiniciar el juego:** Implemente un botón o una función que permita reiniciar el juego. Esto debe:
  - Barajar nuevamente las cartas.
  - Ocultar todas las cartas.
  - Reiniciar cualquier contador o estado.
- **Extras:**
  - Añade un contador de movimientos y de tiempo y muéstralo al jugador durante la partida.
  - Permite cambiar el tamaño de la cuadrícula (por ejemplo, 4x4, 6x6).

Resultado esperado:

- Un mensaje que notifique la victoria cuando todas las cartas estén descubiertas.
- Un botón para reiniciar la partida y comenzar desde cero.

Segundo Minijuego (Tetris)

El objetivo de este proyecto es diseñar e implementar un juego de Tetris desde cero utilizando JavaScript. Todo el contenido dinámico, como la cuadrícula de juego y las piezas, se generará con JavaScript, y el HTML inicial contendrá solo un contenedor vacío para el tablero. Este proyecto se dividirá en **tres partes** para abordar su desarrollo paso a paso.

Parte 1: Tu primera tarea es generar la cuadrícula del tablero y definir las piezas del Tetris. Esto incluye los bloques que caen y sus diferentes formas. Sigue las siguientes instrucciones:

Generar la cuadrícula del tablero: Utilice un contenedor `<div>`en el HTML (por ejemplo, con `id="tablero"`) para representar el área de juego. Genera una cuadrícula dinámica en este contenedor, por ejemplo, de **10 columnas por 20 filas** . Cada celda debe ser una `<div>`con una clase (por ejemplo, `celda`) para poder aplicar estilos y manejar su estado (ocupada o vacía).

Definir las piezas del Tetris: Crea representaciones de las piezas del Tetris en forma de matrices o arrays. Por ejemplo:

- I:`[[1, 1, 1, 1]]`
- □:`[[1, 1], [1, 1]]`
- T:`[[0, 1, 0], [1, 1, 1]]`

Y define una función para rotar estas piezas cambiando su orientación.

Colocar una pieza en el tablero: Implemente una función para agregar una pieza al tablero. La pieza debe aparecer en la parte superior y comenzar a "caer".

Resultado esperado:

- Una cuadrícula visible de celdas de 10x20.
- Piezas definidas y representadas como arrays.
- Una pieza que aparece al inicio y ocupa las celdas correspondientes.

Segundo Minijuego (Tetris)

Parte 2: En esta parte, harás que las piezas se muevan y puedas controlarlas. Sigue estas instrucciones:

Movimiento automático hacia abajo: Haz que la pieza caiga automáticamente, moviéndose una fila hacia abajo a intervalos regulares (usa `setInterval`).

Movimiento controlado por el jugador: Implementa controles de teclado para mover la pieza:

- Izquierda (`ArrowLeft`): Mover una celda a la izquierda.
- Derecha (`ArrowRight`): Mover una celda a la derecha.
- Abajo (`ArrowDown`): Acelerar la caída de la pieza.
- Rotar (`ArrowUp`): Rotar la pieza.

Colisiones: Agregue detección de colisiones para:

- Evite que la pieza se mueva fuera de los bordes del tablero.
- Detectar si la pieza toca otra pieza o el fondo del tablero, lo que detendrá su movimiento.

Fijar la pieza: Si la pieza ya no puede moverse hacia abajo (porque colisiona), "fíjala" en su lugar, marcando las celdas correspondientes como ocupadas. Genera una nueva pieza en la parte superior.

Resultado esperado:

- Las piezas caen automáticamente.
- El jugador puede mover y rotar las piezas con el teclado.
- Las piezas se detuvieron y fijan correctamente cuando colisionan.

Segundo Minijuego (Tetris)

Parte 3: Tu última tarea es implementar la lógica para borrar líneas completas, aumentar la dificultad y verificar el estado del juego.

Borrar líneas completas: Implemente una función para detectar si una fila del tablero está completamente llena (todas las celdas ocupadas). Si una fila está llena, bórrala y desplaza todas las filas superiores hacia abajo.

Puntuación y dificultad: Añade un sistema de puntuación:

- Por ejemplo, 100 puntos por cada línea completada.
- Aumenta la velocidad de caída de las piezas a medida que el jugador obtenga más puntos.

Fin del juego: Si una pieza se fija y no hay espacio para generar una nueva pieza en la parte superior, muestra un mensaje de "Game Over" y detén el juego.

Reiniciar el juego: Agregue un botón o función para reiniciar el juego.

Resultado esperado:

- Líneas completas se detectan y eliminan.
- La puntuación aumenta y la dificultad sube con el progreso.
- El juego finaliza correctamente al llenar el tablero.