Documentación del proyecto Snake (Juego de la serpiente)

El juego de la serpiente es una versión clásica del juego de la serpiente, donde el jugador controla una serpiente que se mueve por la pantalla y debe comer alimentos para crecer, el objetivo de la creación de este juego es lograr obtener la mayor puntuación posible sin chocar con las paredes ni con el propio cuerpo de la serpiente, así mismo el motivo por el cual se decidió hacer dicho proyecto es ya que la mayoría de personas jugaron este famoso juego en algún teléfono de teclas, lo cual fue emotivo para nosotros volver a recordar un juego el cual marco la infancia de muchas personas incluyéndonos.

Lo que utilizamos para crear el juego:

HTML: Para la estructura del documento y el contenido.

CSS: Para el estilo y el diseño del juego.

JavaScript: Lógica del juego y manipulación del DOM.

Explicación del código de HTML5:

```
1   <!DOCTYPE html>
2   <html lang="es">
```

Define el tipo de documento

Indica que el idioma es español.

```
3   <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Juego de la Serpiente</title>
7   <link rel="stylesheet" href="css/styles.css">
8   </head>
```

<meta charset=”UTF-8”> Establece la codificación de caracteres del documento como UTF-8, lo cual permite utilizar una amplia gama de caracteres.

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”> configura el viewport para que el diseño sea responsivo en dispositivos móviles.

<title> Define el título, en este caso define el titulo del juego, el cual se mostrará en la pestaña del navegador.

<link rel=”stylesheet” href=”css/styles.css”> vincula el archivo CSS que contiene los estilos.

```
<body>
<div id="contenedor">
    <!-- Puntuación del juego -->
    <div class="puntuacion">Puntuación: <span class="valor-puntuacion">0</span></div>
    <!-- Canvas donde se dibuja el juego -->
    <canvas id="gameCanvas" width="400" height="400"></canvas>
    <!-- Ranking de jugadores -->
    <div class="ranking">
        <h2>Ranking</h2>
        <div id="rankingList"></div>
    </div>
</div>
<script src="js/game.js"></script>
</body>
</html>
```

: Un contenedor principal que agrupa todo el contenido del juego de la serpiente.

Puntuación: 0 : Muestra la puntuación del juego. El valor de la puntuación se actualiza dinámicamente mediante JavaScript.

<canvas id="gameCanvas" width="400" height="400"></canvas>: Un elemento <canvas> de HTML que se usa para dibujar el juego de la serpiente. Tiene un ancho y alto de 400 píxeles.

: Su función es mostrar el ranking de jugadores.

<h2>Ranking</h2>: Un encabezado que indica la sección de ranking.

: Una lista de los jugadores y sus puntuaciones, generado dinámicamente con JavaScript.

<script src="js/game.js"></script>: Vincula el archivo JavaScript que contiene la lógica del juego.

Explicación del Código CSS:

```css
/* Estilo general del cuerpo de la página */
body {
    font-family: Arial, sans-serif;
    background-color: #333;
    color: #fff;
    text-align: center;
    margin: 0;
    padding: 0;
    overflow: hidden; /* Bloquea el scroll */
}
```

font-family: Establece la fuente a Arial o una fuente sans-serif por defecto.

background-color: Establece el fondo de la página el cual es color gris oscuro.

color: El texto es de color blanco (#fff).

text-align: Centra el texto.

margin y padding: Establece márgenes y rellenos a 0 para eliminar espacios alrededor del cuerpo.

overflow: hidden; para evitar el desplazamiento (scroll) de la página, esto permite que no puedan mover la pantalla hacia arriba, abajo, izquierda o derecha.

```css

/* Estilo del contenedor principal del juego */
#contenedor {
    position: relative;
    margin: 50px auto;
    width: 400px; /* Ajusta el tamaño para que el ranking quede afuera */
    background-color: #555;
    padding: 20px;
    border-radius: 10px;
    border: 2px solid #fff;
}
```

position: relative permite posicionar elementos de forma absoluta en relación al contenedor.

margin: 50px auto: centra horizontalmente y añade un margen superior e inferior de 50px.

width: Fija el ancho en 400px.

background-color: Color de fondo gris medio (#555).

padding: Añade un relleno interno de 20px.

border-radius: Redondea las esquinas con un radio de 10px.

border: Añade un borde blanco de 2px alrededor.

```
22
23  /* Estilo del canvas del juego */
24  canvas {
25      display: block;
26      margin: 0 auto;
27      border: 2px solid #fff;
28      background-color: #000;
29  }
```

display: block convierte el canvas en un elemento de bloque.

margin: 0 auto centra el canvas horizontalmente.

border: Añade un borde blanco de 2px alrededor del canvas.

background-color: El fondo del canvas es color negro (#000)

```
30
31  /* Estilo de la puntuación */
32  .puntuacion {
33      margin-bottom: 20px;
34      font-size: 24px;
35  }
36
```

margin-bottom: Añade un margen inferior de 20px.

font-size: Establece el tamaño de la fuente a 24px.

```
37  /* Estilo del ranking */
38  .ranking {
39      position: absolute;
40      top: 50%;
41      right: -200px; /* Ajusta la posición a la derecha del contenedor */
42      transform: translateY(-50%);
43      width: 180px; /* Ancho del ranking */
44      background-color: #444;
45      padding: 10px;
46      border-radius: 10px;
47      border: 2px solid #fff;
48      text-align: left;
49  }
```

position: absolute permite posicionar el ranking en relación al contenedor principal.

top: 50% coloca el ranking a la mitad del contenedor principal.

right: -200px posiciona el ranking fuera del borde derecho del contenedor.

transform: translateY(-50%) centra verticalmente el ranking.

width: Establece el ancho del ranking a 180px.

background-color: Color de fondo gris oscuro (#444).

padding: Añade un relleno interno de 10px.

border-radius: Redondea las esquinas del ranking con un radio de 10px.

border: Añade un borde blanco de 2px alrededor del ranking.

text-align: Alinea el texto a la izquierda

```
50
51  /* Estilo del título del ranking */
52  .ranking h2 {
53      font-size: 20px;
54      margin-bottom: 10px;
55  }
56
```

font-size: Establece el tamaño de la fuente a 20px.

margin-bottom: Añade un margen inferior de 10px

```
56
57  /* Estilo de los elementos individuales del ranking */
58  .ranking-item {
59      margin-bottom: 5px;
60  }
61
```

margin-bottom: Añade un margen inferior de 5px a cada elemento del ranking.

Explicación del código JavaScript

```
// Variables globales y configuración inicial
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
const escala = 20;
const filas = canvas.height / escala;
const columnas = canvas.width / escala;

let serpiente, fruta, puntuacion, nombreJugador, intervaloJuego;
const velocidadSerpiente = 150; // Velocidad inicial de la serpiente en milisegundos

```

canvas y ctx: Referencias al elemento <canvas> y su contexto de renderizado en 2D.

escala: Tamaño de cada celda en el juego.

filas y columnas: Número de filas y columnas en el canvas, basado en la escala.

serpiente, fruta, puntuacion, nombreJugador, intervaloJuego: Variables para gestionar el estado del juego.

velocidadSerpiente: Intervalo de tiempo en milisegundos para la actualización del juego.

```
// Clase Serpiente
class Serpiente {
    constructor() {
        this.reset(); // Inicializa la serpiente
    }

    reset() {
        this.x = canvas.width / 2;
        this.y = canvas.height / 2;
        this.velocidadX = escala;
        this.velocidadY = 0;
        this.total = 2;
        this.cola = [];
    }

    // Dibuja la serpiente en el canvas
    dibujar() {
        ctx.fillStyle = '#FFFF00'; // Color amarillo para la cabeza
        ctx.fillRect(this.x, this.y, escala, escala);
        ctx.fillStyle = '#00FF00'; // Color verde para el cuerpo
        this.cola.forEach(segmento => ctx.fillRect(segmento.x, segmento.y, escala, escala));
    }

```

reset(): Inicializa o reinicia la serpiente en el centro del canvas con una longitud de 2.

dibujar(): Dibuja la serpiente en el canvas, con la cabeza en amarillo y el cuerpo en verde.

```
34      // Actualiza la posición de la serpiente
35      actualizar() {
36          for (let i = 0; i < this.cola.length - 1; i++) {
37              this.cola[i] = this.cola[i + 1];
38          }
39          this.cola[this.total - 1] = { x: this.x, y: this.y };
40          this.x += this.velocidadX;
41          this.y += this.velocidadY;
42          this.verificarColisionBordes();
43      }
44
45      // Cambia la dirección de la serpiente según las teclas de flecha presionadas
46      cambiarDireccion(direccion) {
47          const movimientos = {
48              'Up': () => { if (this.velocidadY === 0) { this.velocidadX = 0; this.velocidadY = -escala; }},
49              'Down': () => { if (this.velocidadY === 0) { this.velocidadX = 0; this.velocidadY = escala; }},
50              'Left': () => { if (this.velocidadX === 0) { this.velocidadX = -escala; this.velocidadY = 0; }},
51              'Right': () => { if (this.velocidadX === 0) { this.velocidadX = escala; this.velocidadY = 0; }}
52          };
53          movimientos[direccion]?.(); // Si la dirección es válida, la cambia
54      }
```

actualizar(): Actualiza la posición de la serpiente y verifica colisiones con los bordes.

cambiarDireccion(): Cambia la dirección de la serpiente basada en las teclas de flecha presionadas.

```
55
56      // Verifica si la serpiente ha comido la fruta
57      comer(fruta) {
58          if (this.x === fruta.x && this.y === fruta.y) {
59              this.total++;
60              return true;
61          }
62          return false;
63      }
64
65      // Verifica si la serpiente ha colisionado consigo misma
66      verificarColision() {
67          this.cola.forEach(segmento => {
68              if (this.x === segmento.x && this.y === segmento.y) {
69                  this.perder();
70              }
71          });
72      }
73
```

comer(): Verifica si la serpiente ha comido la fruta y aumenta su tamaño.

verificarColision(): Verifica si la serpiente ha colisionado consigo misma.

```
73
74      // Verifica si la serpiente ha colisionado con los bordes del canvas
75      verificarColisionBordes() {
76          if (this.x >= canvas.width || this.x < 0 || this.y >= canvas.height || this.y < 0) {
77              this.perder();
78          }
79      }
80
81      // Maneja la pérdida del juego
82      perder() {
83          clearInterval(intervaloJuego); // Detiene el juego
84          mostrarRanking();
85          alert(`¡Perdiste! Tu puntuación fue: ${puntuacion}. Presiona "Aceptar" para volver a intentarlo.`);
86          solicitarNombre(); // Reinicia el juego
87      }
88  }
89
```

verificarColisionBordes(): Verifica si la serpiente ha colisionado con los bordes del canvas.

perder(): Maneja el evento de perder el juego, mostrando un mensaje y reiniciando el juego.

```
89
90  // Clase Fruta
91  class Fruta {
92      // Coloca la fruta en una ubicación aleatoria dentro del canvas
93      elegirUbicacion() {
94          this.x = Math.floor(Math.random() * columnas) * escala;
95          this.y = Math.floor(Math.random() * filas) * escala;
96      }
97
98      // Dibuja la fruta en el canvas
99      dibujar() {
100         ctx.fillStyle = '#FF0000'; // Color rojo para la fruta
101         ctx.fillRect(this.x, this.y, escala, escala);
102     }
103 }
104
```

elegirUbicacion(): Coloca la fruta en una ubicación aleatoria dentro del canvas.

dibujar(): Dibuja la fruta en el canvas de color rojo.

```
104
105 // Configuración inicial del juego
106 function setup() {
107     solicitarNombre(); // Pide el nombre del jugador
108 }
```

setup(): Función inicial que solicita el nombre del jugador y luego inicia el juego

```
109
110 // Solicita el nombre del jugador y luego inicia el juego
111 function solicitarNombre() {
112     nombreJugador = prompt('Ingresa tu nombre:')?.trim() || 'Invitado';
113     iniciarJuego();
114 }
115
116 // Inicializa o reinicia el juego
117 function iniciarJuego() {
118     puntuacion = 0;
119     actualizarPuntuacion(); // Resetea la puntuación
120     clearInterval(intervaloJuego); // Limpia cualquier intervalo de juego previo
121     serpiente = new Serpiente(); // Crea una nueva serpiente
122     fruta = new Fruta(); // Crea una nueva fruta
123     fruta.elegirUbicacion(); // Coloca la fruta en una ubicación aleatoria
124     intervaloJuego = setInterval(juegoLoop, velocidadSerpiente); // Inicia el bucle del juego
125 }
```

solicitarNombre(): Solicita el nombre del jugador y llama a iniciarJuego().

iniciarJuego(): Inicializa o reinicia el juego, creando una nueva serpiente y fruta, y empezando el bucle del juego.

```
126
127 // Bucle principal del juego
128 function juegoLoop() {
129     ctx.clearRect(0, 0, canvas.width, canvas.height); // Limpia el canvas
130     fruta.dibujar(); // Dibuja la fruta
131     serpiente.actualizar(); // Actualiza la posición de la serpiente
132     serpiente.dibujar(); // Dibuja la serpiente
133
134     if (serpiente.comer(fruta)) { // Si la serpiente come la fruta
135         puntuacion += 10; // Aumenta la puntuación
136         fruta.elegirUbicacion(); // Coloca una nueva fruta
137         actualizarPuntuacion(); // Actualiza la puntuación en pantalla
138     }
139
140     serpiente.verificarColision(); // Verifica colisiones de la serpiente
141 }
142
```

juegoLoop(): Función que se llama repetidamente para actualizar el estado del juego y dibujar los elementos.

```
143 // Actualiza la visualización de la puntuación en la página
144 function actualizarPuntuacion() {
145     document.querySelector('.valor-puntuacion').innerText = puntuacion;
146 }
147
```

actualizarPuntuacion(): Actualiza la puntuación mostrada en la página.

```
147
148 // Muestra el ranking de jugadores
149 function mostrarRanking() {
150     const rankingList = document.getElementById('rankingList');
151     const rankingItem = document.createElement('div');
152     rankingItem.classList.add('ranking-item');
153     rankingItem.innerText = `${nombreJugador}: ${puntuacion}`;
154     rankingList.appendChild(rankingItem); // Agrega el nuevo elemento al ranking
155
156     // Ordena el ranking
157     const items = Array.from(rankingList.children);
158     items.sort((a, b) => parseInt(b.innerText.split(': ')[1]) - parseInt(a.innerText.split(': ')[1]));
159
160     rankingList.innerHTML = ''; // Limpia el ranking
161     items.slice(0, 5).forEach(item => rankingList.appendChild(item)); // Muestra los 5 mejores
162 }
```

mostrarRanking(): Muestra y ordena el ranking de jugadores en la página, manteniendo los 5 mejores resultados.

```
163
164 // Detecta las pulsaciones de teclas y cambia la dirección de la serpiente
165 window.addEventListener('keydown', evt => serpiente.cambiarDireccion(evt.key.replace('Arrow', '')));
166
167 setup(); // Inicia la configuración del juego
168
```

Esta sección del código se encarga de detectar cuando el usuario presiona una tecla y, en respuesta, cambia la dirección de la serpiente en el juego.