MANUAL DEL JUEGO

FLAPPY FROG

MANUAL Y JUEGO
ELABORADO POR: YAZU BAUTISTA
DISEÑO Y MAQUETACION: MAURICIO CASTILLO

MANUAL DE USO DEL JUEGO “FLAPPY FROG”

Introducción

El juego Flappy Frog es una versión inspirada en Flappy Bird, pero protagonizada por una rana.
El objetivo es que el jugador mantenga a la rana volando entre tubos verdes, sin chocar ni caer al suelo.
Cada tubo que se supera aumenta la puntuación.
El juego está desarrollado utilizando HTML, CSS y JavaScript, tres lenguajes esenciales para la creación de videojuegos y páginas web.

Documentos necesarios

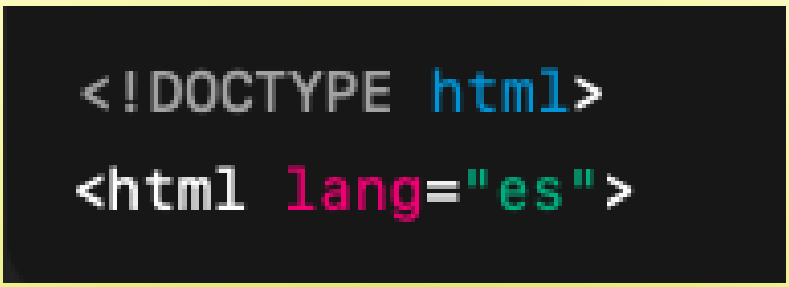
Para que el juego funcione correctamente, es necesario tener los siguientes tres archivos y dos imágenes en la misma carpeta:

1. index.html → Contiene la estructura principal del juego.
2. style.css → Define los colores, tamaños y posiciones (el estilo visual).
3. game.js → Contiene la programación y las funciones del juego.
4. 1.png → Imagen de la rana (jugador).
   2.png → Imagen del fondo del escenario.

Explicación detallada del código HTML (index.html)

El archivo index.html es el que define toda la estructura visual del juego. Dentro de él se colocan los elementos que se mostrarán en pantalla (canvas, botones, puntuación) y las conexiones con el CSS y JavaScript.

A continuación se explica línea por línea:



Estructura general

<!DOCTYPE html> → Indica que el documento está escrito en HTML5, la versión más moderna.

<html lang="es"> → Indica que el idioma principal de la página es el español.

Sección <head>

Esta sección contiene la configuración interna del documento, que no se ve en pantalla, pero es muy importante.

<meta charset="UTF-8"> → Permite usar caracteres especiales (como “ñ” o tildes).

<meta name="viewport"...> → Hace que el juego se vea correctamente en teléfonos y computadoras (ajuste responsivo).

<title> → Muestra el nombre “Flappy Frog” en la pestaña del navegador.

<link rel="stylesheet" href="style.css"> → Conecta el archivo de estilos CSS con el HTML, para aplicar colores y posiciones.

Cuerpo principal <body>

Dentro del <body> se coloca todo lo que el jugador verá en pantalla:

```
dy>
 <div id="game-container">
     <canvas id="gameCanvas"></canvas>
     <div id="score-container">
         Puntuación: <span id="score">0</span>
     </div>
     <button id="start-btn" onclick="startGame()">Iniciar</button>
     <button id="restart-btn" style="display:none" onclick="restartGame()">Reiniciar</but
 </div>

 <script src="game.js"></script>
ody>
tml>
```

- Es el contenedor principal del juego.
- Dentro de él están todos los elementos que lo componen:
- el área de juego (canvas), la puntuación y los botones.

<canvas id="gameCanvas"></canvas>

- Es el lienzo donde se dibuja la rana, los tubos y el fondo.
- Todo el movimiento del juego ocurre aquí gracias al código JavaScript.
- No tiene contenido propio porque todo se dibuja dinámicamente.

```
<div id="score-container">
    Puntuación: <span id="score">0</span>
</div>
```

- Muestra la puntuación actual del jugador.
- El texto “Puntuación:” es fijo, mientras que el número dentro de va aumentando con cada tubo superado.
- El valor inicial es 0.

Botón “Iniciar”

```
<button id="start-btn" onclick="startGame()">Iniciar</button>
```

- Es el botón verde que aparece al comienzo.
- Al hacer clic, ejecuta la función startGame() (definida en game.js) para empezar la partida.

Botón “Reiniciar”

```
<button id="restart-btn" style="display:none" onclick="restartGame()">Reiniciar</button
```

- Es el botón rojo que solo aparece cuando el jugador pierde.
- Tiene un estilo inicial display:none, lo que significa que está oculto al principio.
- Cuando se muestra, al hacer clic llama la función restartGame() para reiniciar el juego y volver a empezar.

Enlace con JavaScript

```
<script src="game.js"></script>
```

- Conecta el archivo de programación (game.js) con el HTML.
- Este archivo contiene toda la lógica del juego (movimientos, colisiones, puntuación, etc.).
- Es lo último del documento, para que se cargue después de que todo el HTML esté listo.

Cierre del documento

```
</body>
</html>
```

- </body> → Cierra el cuerpo del documento.
- </html> → Marca el fin completo del archivo HTML.

2. Documento CSS – style.css

Este documento controla la apariencia visual del juego: colores, posición de elementos, tamaños y fuentes.

Configuración general

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

- Quita márgenes y rellenos predeterminados de todos los elementos.
- Asegura que el diseño sea uniforme en cualquier navegador.

Diseño del cuerpo del sitio

```css
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #87CEFA; /* Cielo celeste */
    font-family: Arial, sans-serif;
}
```

- Usa flexbox para centrar el contenido.
- background-color aplica un fondo celeste que simula el cielo.
- height: 100vh ocupa toda la pantalla.

Contenedor del jueg

```css
#game-container {
    position: relative;
    width: 100%;
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

- Mantiene todo centrado en pantalla.
- position: relative permite colocar los botones y la puntuación encima del lienzo.

Área de juego (canvas)

```
#gameCanvas {
    border: 2px solid #000;
    display: block;
    background-color: #87CEFA;
    position: absolute;
}
```

- border: le da un marco al área de juego.
- position: absolute: permite que otros elementos (como los botones) se ubiquen encima.

Puntuación

```
#score-container {
    font-size: 24px;
    color: #FFFFFF;
    position: absolute;
    top: 10px;
}
```

- Muestra el texto “Puntuación: 0” en la parte superior.
- position: absolute lo coloca fijo sin moverse con el lienzo.

Botones

```
#start-btn, #restart-btn {
    padding: 10px 20px;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
    position: absolute;
    bottom: 50px;
}
```

- Ambos botones se posicionan en la parte inferior.
- cursor: pointer cambia el puntero al pasar el ratón.
- border-radius les da bordes redondeados.

Botón “Iniciar”

```
#start-btn {
    background-color: #28a745;
    color: white;
}
#start-btn:hover {
    background-color: #218838;
}
#start-btn:active {
    background-color: #1e7e34;
}
```

- Verde = iniciar partida.
- Usa distintos tonos al pasar o presionar.

Botón “Reiniciar”

```css
#restart-btn {
    background-color: #dc3545;
    color: white;
    display: none;
}
#restart-btn:hover {
    background-color: #c82333;
}
#restart-btn:active {
    background-color: #bd2130;
}
```

- Rojo = reiniciar la partida cuando pierdes.
- Al inicio está oculto (display: none).

3. Documento JavaScript – game.js

Aquí se encuentra la programación del juego: control de movimiento, gravedad, colisiones, tubos, puntuación y reinicio.

Configuración inicial

```js
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
```

- Obtiene el lienzo (canvas) donde se dibuja el juego.
- ctx es el contexto 2D para dibujar imágenes y figuras.
- Ajusta el tamaño del juego a toda la pantalla.

Carga de imágenes

```
const backgroundImage = new Image();
backgroundImage.src = '2.png'; // Fondo del juego

const frogImage = new Image();
frogImage.src = '1.png'; // Imagen de la rana
```

Estas imágenes deben guardarse en la misma carpeta que los otros archivos.

Objeto “rana”

```
let frog = {
    x: 50,
    y: canvas.height / 2,
    width: 100,
    height: 100,
    velocity: 0,
    flap() {
        this.velocity = -10; // Salto
    },
    update() {
        this.velocity += 0.5; // Gravedad
        this.y += this.velocity;
    },
    draw() {
        ctx.drawImage(frogImage, this.x, this.y, this.width, this.height);
    }
};
```

- La rana tiene posición (x, y), tamaño y movimiento.
- flap(): simula el salto.
- update(): aplica la gravedad.
- draw(): dibuja la rana en pantalla.

Creación y dibujo de tubos

```
let pipes = [];

function createPipe() {
    const gap = 250;
    const height = Math.random() * (canvas.height - gap - 100) + 50;
    const pipeTop = { x: canvas.width, y: 0, width: 60, height: height };
    const pipeBottom = { x: canvas.width, y: height + gap, width: 60, height: canvas.he
    pipes.push(pipeTop, pipeBottom);
}
```

- Crea tubos con un espacio (gap) entre ellos.
- Math.random() hace que cada tubo tenga una altura diferente.

```
function drawPipes() {
    ctx.fillStyle = '#00FF00';
    pipes.forEach(pipe => {
        ctx.fillRect(pipe.x, pipe.y, pipe.width, pipe.height);
    });
}
```

- Pinta los tubos de color verde.
- Se actualizan en cada fotograma para simular movimiento.

Colisiones

```
function detectCollisions() {
    pipes.forEach(pipe => {
        if (
            frog.x < pipe.x + pipe.width &&
            frog.x + frog.width > pipe.x &&
            frog.y < pipe.y + pipe.height &&
            frog.y + frog.height > pipe.y
        ) {
            gameOver = true;
            showGameOver();
        }
    });
}
```

```
let score = 0;

function updateScore() {
    pipes.forEach(pipe => {
        if (pipe.x + pipe.width < frog.x && !pipe.passed) {
            score++;
            pipe.passed = true;
        }
    });
    document.getElementById('score').textContent = score;
}
```

- Detecta si la rana toca algún tubo.
- Si choca, muestra el mensaje de Game Over.

Marcador y reinicio

- Aumenta el puntaje cada vez que se pasa un tubo.
- Muestra el valor actualizado en pantalla.

Funciones principales del juego

```
function startGame() {
    gameStarted = true;
    document.getElementById('start-btn').style.display = 'none';
    gameLoop();
}

function restartGame() {
    pipes = [];
    frog.y = canvas.height / 2;
    frog.velocity = 0;
    score = 0;
    gameOver = false;
    frame = 0;
    document.getElementById('score').textContent = score;
    document.getElementById('restart-btn').style.display = 'none';
    gameLoop();
}
```

- startGame() → comienza la partida.

restartGame() → reinicia todo cuando el jugador pierde.

Bucle principal

```
function gameLoop() {
    frame++;
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);
    if (frame % 90 === 0 && !gameOver) createPipe();
    frog.update(); frog.draw();
    drawPipes(); detectCollisions(); updateScore();
    if (!gameOver) requestAnimationFrame(gameLoop);
}
```

- Este bucle actualiza y redibuja todo constantemente.
- Controla el ritmo del juego y genera movimiento fluido.

Conclusión

El juego “Flappy Frog” muestra cómo combinar HTML, CSS y JavaScript para crear un videojuego simple, divertido y visualmente atractivo.

Cada documento cumple un papel esencial:

- HTML → estructura del juego
- CSS → apariencia y diseño
- JS → movimiento, colisiones y lógica

El resultado es una experiencia interactiva completa desarrollada con las tecnologías básicas de la web.