ATRAPA A LOS PILOTOS DE F1

Manual y juego elaborado por:Yazu Bautista
Diseño y maquetación: Evelyn Santiago Paez

Manual del juego Atrapa a los pilotos de F1

1. Introducción

Este juego está desarrollado usando HTML, CSS y JavaScript. El jugador controla un auto de Fórmula 1 para atrapar pilotos que caen del cielo. El objetivo es capturar la mayor cantidad de pilotos en 30 segundos, sumando puntos por cada piloto atrapado.

El juego combina estructura (HTML), estilo (CSS) y lógica (JS), lo que permite comprender cómo se construye una página interactiva paso a paso.

2. Archivos del juego

El juego utiliza tres tipos de archivos:

1. HTML (f1-juego.html) → Define la estructura de la página y los elementos visibles.
2. CSS (f1-juego.css) → Da estilo, colores, posiciones y animaciones a los elementos.
3. JavaScript (f1-juego.js) → Controla la interactividad, movimiento, puntaje y temporizador.

Imágenes necesarias:

car.png (auto), max.jpg, carlos.jpg, kimi.jpg, oscar.jpg (pilotos), flecha.png (botón de regreso).

3. HTML Paso a Paso

3.1 Declaración inicial

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

- <!DOCTYPE html> indica que se está usando HTML5.
- <html lang="es"> define el idioma de la página (español).

3.2 Encabezado (<head>)

```
<head>
  <meta charset="UTF-8" />
  <title>Atrapa Pilotos F1</title>
  <link rel="stylesheet" href="f1-juego.css" />
</head>
```

- <meta charset="UTF-8"> → Permite caracteres especiales y emojis.
- <title> → Título visible en la pestaña del navegador.
- <link rel="stylesheet" href="f1-juego.css"> → Vincula el CSS externo.

3.3 Cuerpo (<body>)

```
<a href="f1.html"><img src="flecha.png" class="imagen-derecha"></a>
```

- <a> crea un enlace al menú anterior.
- <img> es el botón gráfico de regreso.

```
<h1>🏎️ Atrapa a los Pilotos de F1 🏁</h1>
```

- <h1> es el título principal del juego.

```
<div class="info">
  <p>Puntos: <span id="score">0</span></p>
  <p>Tiempo: <span id="timer">30</span>s</p>
</div>
```

- contiene el panel de información.
- <p> → Párrafos que muestran puntaje y tiempo.
- y permiten que JS actualice dinámicamente los valores.

```
<div class="game-area">
  <div id="carro"></div>
</div>
```

- → Área de juego donde se mueve el auto y caen los pilotos.
- → Representa el auto, identificado por su id para JS.

```
<div id="game-over" class="hidden">
  <h2>💣 ¡Tiempo agotado!</h2>
  <p>Puntaje final: <span id="final-score">0</span></p>
  <button onclick="location.reload()">Reiniciar</button>
</div>
```

- Pantalla de fin de juego, oculta por defecto (.hidden).
- <h2> → Título del mensaje.
- <p> → Muestra el puntaje final usando dinámico.

<button onclick="location.reload()"> → Permite reiniciar el juego.

```
<script src="f1-juego.js"></script>
```

- Vincula el archivo JavaScript que controla movimiento, generación de pilotos, colisiones y temporizador.

4.CSS Paso a Paso

4.1 Estilo general del cuerpo

```
body {
  text-align: center;
  font-family: Arial, sans-serif;
  background: #f0f0f0;
  margin: 0;
}
```

- text-align: center; → Centra todo el contenido horizontalmente para que los elementos se vean alineados.
- font-family: Arial, sans-serif; → Fuente clara y legible para el texto.
- background: #f0f0f0; → Fondo gris claro, hace contraste con el área de juego oscuro.
- margin: 0; → Elimina el espacio por defecto del navegador, dejando la página ajustada al borde.

4.2 Botón de regreso

```css
.imagen-derecha {
  position: absolute;
  top: 40px;
  left: 60px;
  width: 90px;
}
```

- position: absolute; → Permite colocar la imagen en un lugar específico de la pantalla.
- top: 40px; → Distancia desde la parte superior de la ventana.
- left: 60px; → Distancia desde la izquierda de la ventana.
- width: 90px; → Define tamaño del botón para que no sea demasiado grande ni pequeño.
- Nota: Se usa absolute porque el botón debe estar fijo sin importar el tamaño del área de juego.

4.3 Título y panel de información

```css
h1 {
  margin-top: 20px;
}
.info {
  margin: 10px;
  font-size: 18px;
}
```

- margin-top: 20px; → Separación entre el borde superior y el título.
- .info → Margen pequeño para separar el panel de la zona superior.
- font-size: 18px; → Tamaño de fuente visible y cómodo para ver puntaje y tiempo.

4.4 Área de juego

```css
.game-area {
  position: relative;
  width: 100%;
  height: 500px;
  background: #222;
  overflow: hidden;
  border: 4px solid #000;
}
```

- position: relative; → Permite que los elementos hijos (auto y pilotos) se posicionen relativamente a este contenedor.
- width: 100%; → Ocupa todo el ancho disponible del navegador.
- height: 500px; → Tamaño fijo para definir la zona de juego.
- background: #222; → Fondo oscuro que resalta el auto y pilotos.
- overflow: hidden; → Los pilotos que caen fuera de esta zona no se muestran, evita que salgan del área.
- border: 4px solid #000; → Borde negro visible para delimitar la zona de juego.

4.5 Auto

```css
#carro {
  width: 80px;
  height: 40px;
  background-image: url('car.png');
  background-size: cover;
  position: absolute;
  bottom: 10px;
  left: 50%;
  transform: translateX(-50%);
}
```

- width y height → Tamaño del auto proporcional al área de juego.
- background-image → Imagen del auto.
- background-size: cover; → Asegura que la imagen llene el contenedor sin deformarse.
- position: absolute; → Permite mover el auto dinámicamente con JS.
- bottom: 10px; → Coloca el auto cerca del borde inferior.
- left: 50%; → Posición horizontal inicial centrada.
- transform: translateX(-50%); → Ajusta el centro exacto del auto para que quede centrado con respecto a left: 50%.

4.6 Pilotos

```css
.piloto {
  position: absolute;
  width: 40px;
  height: 40px;
  border-radius: 50%;
  top: -40px;
  animation: caer 4s linear forwards;
}
```

- position: absolute; → Permite mover cada piloto individualmente desde JS.
- width y height → Tamaño uniforme para todos los pilotos.
- border-radius: 50%; → Hace que la imagen sea circular.
- top: -40px; → Comienza fuera del área de juego, para que caiga visualmente desde arriba.
- animation: caer 4s linear forwards; → Aplica la animación de caída definida abajo.

```css
@keyframes caer {
  0% { top: -40px; }
  100% { top: 500px; }
}
```

- @keyframes caer → Define cómo se mueve el piloto desde arriba (-40px) hasta el borde inferior (500px).
- 4s → Tiempo total de caída.
- linear → Movimiento constante, sin aceleración.
- forwards → Mantiene al piloto en la posición final al terminar la animación.

4.7 Pantalla de fin

```css
#game-over { margin-top: 20px; }
.hidden { display: none; }
```

- #game-over → Separa del área de juego para mostrar mensaje y puntaje final.
- .hidden → Oculta elementos hasta que JS los muestre.

5. JavaScript Paso a Paso

5.1 Selección de elementos

```
const carro = document.getElementById("carro");
const gameArea = document.querySelector(".game-area");
const scoreDisplay = document.getElementById("score");
const timerDisplay = document.getElementById("timer");
const finalScore = document.getElementById("final-score");
const gameOverBox = document.getElementById("game-over");
```

- getElementById y querySelector → Permiten manipular elementos específicos de HTML desde JS.
- Cada variable hace referencia a un elemento para actualizar posición, puntaje o mostrar pantallas.

5.2 Variables de control

```
const pilotos = ["max.jpg", "carlos.jpg", "kimi.jpg", "oscar.jpg"];
let score = 0;
let tiempo = 30;
let gameOver = false;
let movingLeft = false;
let movingRight = false;
```

- pilotos → Array con imágenes de los pilotos para generar aleatoriamente.
- score → Puntaje acumulado por piloto atrapado.
- tiempo → Duración del juego en segundos.
- gameOver → Controla si el juego terminó.
- movingLeft y movingRight → Detectan si el auto se está moviendo en una dirección.

5.3 Movimiento del auto

```
function moveCar(time) {
  if (!lastTime) lastTime = time;
  const delta = time - lastTime;
  lastTime = time;

  const maxLeft = gameArea.clientWidth - carro.offsetWidth;
  let current = carro.offsetLeft;

  const distancia = (velocidadPxPorSeg * delta) / 1000;

  if (movingLeft) {
    carro.style.left = Math.max(current - distancia, 0) + "px";
  }
  if (movingRight) {
    carro.style.left = Math.min(current + distancia, maxLeft) + "px";
  }

  if (!gameOver && (movingLeft || movingRight)) {
    requestAnimationFrame(moveCar);
  } else {
    lastTime = null;
  }
}
```

- delta → Calcula tiempo transcurrido entre frames para que el movimiento sea suave y proporcional al tiempo real.
- maxLeft → Evita que el auto salga del área de juego.
- distancia → Cuánto debe moverse el auto según velocidad y tiempo.
- Math.max y Math.min → Limita la posición del auto dentro del área de juego.
- requestAnimationFrame → Ejecuta la animación de forma fluida según la frecuencia de refresco del navegador.

```
document.addEventListener("keydown", (e) => {
  if (e.key === "ArrowLeft") movingLeft = true;
  if (e.key === "ArrowRight") movingRight = true;
});
document.addEventListener("keyup", (e) => {
  if (e.key === "ArrowLeft") movingLeft = false;
  if (e.key === "ArrowRight") movingRight = false;
});
```

- Detecta presión y liberación de teclas para iniciar y detener movimiento.

5.4 Generación de pilotos

```
function crearPiloto() {
  if (gameOver) return;

  const piloto = document.createElement("img");
  piloto.classList.add("piloto");
  piloto.src = pilotos[Math.floor(Math.random() * pilotos.length)];
  piloto.style.left = Math.random() * (gameArea.clientWidth - 40) + "px";
  gameArea.appendChild(piloto);

  const fallInterval = setInterval(() => {
    const top = piloto.offsetTop;
    if (top > 460) {
      const pilotoLeft = piloto.offsetLeft;
      const carroLeft = carro.offsetLeft;
      const carroRight = carroLeft + carro.offsetWidth;
      if (pilotoLeft + 30 > carroLeft && pilotoLeft < carroRight) score++;
      piloto.remove();
      clearInterval(fallInterval);
    }
  }, 50);
}
```

- document.createElement("img") → Crea elemento piloto dinámicamente.
- Math.random() → Posición horizontal aleatoria.
- setInterval → Verifica posición del piloto cada 50ms para detectar colisión.
- score++ → Incrementa puntaje si el piloto colisiona con el auto.

5.5 Temporizador y fin del juego

```
const pilotoInterval = setInterval(crearPiloto, 800);

const tiempoInterval = setInterval(() => {
  tiempo--;
  timerDisplay.textContent = tiempo;
  if (tiempo <= 0) {
    clearInterval(pilotoInterval);
    clearInterval(tiempoInterval);
    gameOver = true;
    gameOverBox.classList.remove("hidden");
    finalScore.textContent = score;
  }
}, 1000);
```

- pilotoInterval → Genera un nuevo piloto cada 0.8 segundos.
- tiempoInterval → Disminuye tiempo cada segundo.
- clearInterval → Detiene intervalos cuando el juego termina.
- gameOverBox.classList.remove("hidden") → Muestra la pantalla de fin del juego.

6. Instrucciones de uso

1. Coloca todos los archivos y las imágenes en la misma carpeta.
2. Abre f1-juego.html en un navegador moderno.
3. Usa flechas izquierda y derecha para mover el auto.
4. Atrapa los pilotos para sumar puntos.
5. Juego dura 30 segundos.
6. Pulsa “Reiniciar” para volver a jugar.

7. Recomendaciones

- Usar navegador actualizado.
- Mantener rutas de imágenes correctas.
- Ajustar velocidad o tiempo desde f1-juego.js si deseas personalizar.

8. Conclusión

Este manual explica cada línea de código y su función, permitiendo entender cómo HTML estructura, CSS estiliza y JS controla un juego interactivo.

El juego es totalmente personalizable, permitiendo agregar nuevos pilotos o animaciones para mejorar la experiencia.

9.Imagenes necesarias