ORBITAL DODGE

Manual y juego elaborado por:
Derek Revilla
Diseño y maquetación:
Derek Revilla

🎮 MANUAL DE USO DEL JUEGO "ORBITAL DODGE 🌀

Introducción

El juego “Orbital Dodge 🌀” es un mini videojuego hecho completamente con HTML, CSS y JavaScript.
El objetivo es controlar una esfera que orbita alrededor de un planeta y evitar ser golpeado por los meteoritos que se acercan desde el exterior.
El jugador puede cambiar la dirección de su órbita presionando las teclas ⬅️ (izquierda) o ➡️ (derecha).
Cada meteorito que pasa el planeta sin chocar con el jugador suma un punto.
El proyecto está compuesto por un solo archivo HTML, dentro del cual se integran los estilos CSS y el código JavaScript.
Sin embargo, este documento explicará cada parte por separado.

1. Estructura del documento HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Orbital Dodge 🌀</title>
```

- <!DOCTYPE html>: indica que el documento está escrito en HTML5.
- <html lang="es">: define el idioma principal de la página (español).
- <head>: contiene la información técnica y los recursos del documento.
- <meta charset="UTF-8">: permite mostrar correctamente caracteres especiales (como tildes o eñes).
- <meta name="viewport">: adapta el juego a pantallas pequeñas o móviles.
- <title>: define el nombre que aparecerá en la pestaña del navegador (“Orbital Dodge 🌀”).

2. Estilos CSS

Los estilos están escritos dentro de la etiqueta <style> en el <head> del documento.
a) Importar fuente tipográfica

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@700&display=swap');
```

- Importa la fuente Montserrat, que se usa para los textos del juego.

- b) Estilos generales del cuerpo

```
body {
  margin: 0;
  background: radial-gradient(circle at center, #0a1128 0%, #001f54 80%);
  overflow: hidden;
  color: #e0e0e0;
  font-family: 'Montserrat', sans-serif;
  user-select: none;
  display: flex;
```

- Elimina márgenes del navegador (margin: 0).
- Crea un fondo espacial con un gradiente circular azul oscuro.
- Evita que el jugador seleccione texto con el mouse.
- Centra todos los elementos verticalmente.
- Aplica la fuente Montserrat y color de texto gris claro.

c) Título y marcador

```css
h1 {
  margin: 20px 0 5px 0;
  color: #00ffc8;
  text-shadow: 0 0 8px #00ffc8aa;
}

#score {
  font-size: 1.3rem;
  margin-bottom: 15px;
  font-weight: 700;
  color: #00ffc8cc;
}
```

h1: muestra el título "Orbital Dodge 🌀" con efecto de brillo verde-azulado.
#score: muestra los puntos del jugador en color cian.

d) Lienzo del juego (<canvas>)

```css
canvas {
  background: transparent;
  border-radius: 15px;
  box-shadow: 0 0 20px #00ffbdcc;
  display: block;
  max-width: 100vw;
}
```

El canvas es donde se dibuja el juego.
Tiene esquinas redondeadas y brillo alrededor.
Ocupa todo el ancho disponible del navegador.

e) Instrucciones y mensaje de reinicio

```css
#instructions {
  font-size: 1rem;
  margin-bottom: 10px;
  color: #008066cc;
}

#restartMsg {
  margin-top: 15px;
  font-size: 1.1rem;
  color: #00ffbdcc;
  opacity: 0.85;
}
```

#instructions: muestra cómo jugar (teclas izquierda y derecha).
#restartMsg: aparece al final del juego, indicando que se presione R para reiniciar.

3. Estructura del cuerpo (<body>)

```html
<h1>Orbital Dodge 🌀</h1>
<div id="score">Puntos: 0</div>
<div id="instructions">Presiona ⬅️ o ➡️ para cambiar direccion</div>
<canvas id="gameCanvas" width="500" height="500"></canvas>
<div id="restartMsg"></div>
```

- **Título: nombre del juego.**
- **Marcador de puntos: inicia en 0.**
- **Instrucciones: muestra las teclas de control.**
- **Canvas: espacio donde se dibujan el planeta, jugador y meteoritos.**
- **Mensaje de reinicio: se mostrará cuando el jugador pierda.**

4. Código JavaScript – Lógica del Juego

El bloque de <script> contiene toda la programación.
Se divide en secciones:

a) Elementos base y variables iniciales

```
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
const scoreEl = document.getElementById('score');
const restartMsg = document.getElementById('restartMsg');
```

Se obtienen los elementos del HTML y el contexto 2D del canvas (donde se dibuja todo).

```
const centerX = canvas.width / 2;
const centerY = canvas.height / 2;
```

Calcula el centro del lienzo, usado como punto de referencia del planeta.

b) Jugador (la esfera que orbita)

```
const player = {
  radius: 120,
  angle: 0,
  size: 18,
  speed: 0.03,
  direction: 1,
  color: '#00ffc8',
};
```

radius: distancia del jugador al centro (órbita).
angle: posición angular alrededor del planeta.
speed: velocidad de rotación.
direction: 1 = derecha, -1 = izquierda.
color: color del jugador (verde brillante).

c) Meteoritos y puntuación

```
let meteors = [];
let meteorSpeed = 2.2;
let spawnTimer = 0;
let spawnInterval = 1500;

let score = 0;
let gameOver = false;
```

Controlan la creación, velocidad y frecuencia de los meteoritos.
score almacena los puntos del jugador.
gameOver indica si la partida terminó.

d) Dibujo de los elementos

Planeta Central:

```
function drawPlanet() {
  const gradient = ctx.createRadialGradient(centerX, centerY, 20, centerX, center
  gradient.addColorStop(0, '#008066ff');
  gradient.addColorStop(1, '#002b2200');
  ctx.fillStyle = gradient;
  ctx.beginPath();
  ctx.arc(centerX, centerY, 70, 0, Math.PI * 2);
  ctx.fill();
  ctx.strokeStyle = '#00ffbd';
  ctx.lineWidth = 3;
  ctx.stroke();
}
```

Dibuja el planeta con degradado y borde brillante.
Jugador orbitando:

Dibuja el planeta con degradado y borde brillante.
Jugador orbitando:

```
function drawPlayer() { ... }
```

Calcula su posición con coseno y seno según su ángulo.
Meteoritos:

```
function spawnMeteor() { ... }
function drawMeteors() { ... }
```

→ Se generan desde fuera del lienzo y se mueven hacia el centro.
Órbita del jugador:

```
function drawOrbit() { ... }
```

→ Dibuja una línea punteada circular que marca el camino del jugador.

e) Movimiento y colisiones

```
function updatePlayer() { player.angle += player.speed * player.direction; }
function updateMeteors() { ... }
function checkCollisions() { ... }
```

El jugador rota continuamente.
Los meteoritos se acercan al planeta.
Si un meteorito toca al jugador → gameOver = true.

f) Sistema de puntuación

```
function updateScore() {
  meteors.forEach(m => {
    const dist = Math.hypot(m.x - centerX, m.y - centerY);
    if (dist <= 75) score++;
  });
}
```

Cada meteorito que alcanza el planeta sin golpear al jugador suma un punto.
g) Bucle principal del juego

```
function gameLoop(timestamp) { ... requestAnimationFrame(gameLoop); }
```

- Dibuja y actualiza todo el juego continuamente.
- Usa requestAnimationFrame para una animación fluida.
- Aumenta la dificultad con el tiempo (más meteoritos, más rápidos).

h) Controles del teclado

```
window.addEventListener('keydown', e => {
  if (e.code === 'ArrowLeft') player.direction = -1;
  if (e.code === 'ArrowRight') player.direction = 1;
  if (e.code === 'KeyR' && gameOver) resetGame();
});
```

- Permite cambiar la dirección del jugador.
- La tecla R reinicia la partida.

- Permite cambiar la dirección del jugador.
- La tecla R reinicia la partida.

i) Reinicio del juego

```
function resetGame() {
  meteors = [];
  score = 0;
  spawnInterval = 1500;
  meteorSpeed = 2.2;
  player.angle = 0;
  gameOver = false;
  restartMsg.textContent = '';
}
```

🧠 Conclusión

El código del juego “Orbital Dodge 🌀” combina física simple, animación y detección de colisiones en un solo documento HTML.
Gracias al uso del canvas, los elementos se pueden animar fluidamente, creando una experiencia visual atractiva.
El jugador solo necesita usar las flechas para girar en la órbita y esquivar los meteoritos, mientras que el sistema de puntuación y dificultad creciente hacen el juego dinámico y entretenido.