Manual del juego

BREAKOUT

MANUAL Y JUEGO
ELABORADO POR:
DEREK REVILLA
DISEÑO Y
MAQUETACION:
MAURICIO CASTILLO

MANUAL DETALLADO DEL JUEGO “BREAKOUT”

Descripción general, archivos necesarios y explicación del código

1. DESCRIPCIÓN DEL JUEGO

Breakout es un videojuego clásico del género arcade que fue creado originalmente por Atari en la década de 1970.

Su objetivo es simple pero adictivo: el jugador controla una barra (también llamada paleta) ubicada en la parte inferior de la pantalla, la cual debe utilizar para golpear una pelota y destruir una serie de ladrillos situados en la parte superior.
Cada vez que la pelota toca un ladrillo, este desaparece y el jugador obtiene puntos. Si la pelota cae al fondo de la pantalla sin ser interceptada por la paleta, el jugador pierde una vida. El juego termina cuando se pierden todas las vidas o se destruyen todos los ladrillos, logrando la victoria.

Este proyecto recrea el clásico Breakout usando únicamente HTML, CSS y JavaScript, sin necesidad de librerías o motores de juego externos.
Su propósito es didáctico: permite comprender conceptos fundamentales de la programación de videojuegos en 2D, como el manejo del lienzo (canvas), detección de colisiones, animaciones con requestAnimationFrame, y control de movimiento mediante el teclado.

2. DOCUMENTOS NECESARIOS PARA SU FUNCIONAMIENTO

- Archivo principal del juego:
- El documento más importante es un archivo con extensión .html, por ejemplo index.html.
- Dentro de este archivo se encuentra todo el código necesario del juego: la estructura HTML, los estilos CSS y la programación en JavaScript.

3. ESTRUCTURA BASE HTML

```
<!DOCTYPE html>
```

Indica que el documento está en formato HTML5, la versión más actual y estándar.
Esto garantiza compatibilidad con todos los navegadores modernos.

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

Define que el idioma principal del contenido es español, lo cual ayuda a motores de búsqueda y lectores de pantalla.

4. ENCABEZADO DEL DOCUMENTO

```
<head>
  <meta charset="UTF-8" />
```

Permite usar caracteres especiales como “ñ”, “á”, “ó”, etc.
Sin esto, algunos textos podrían aparecer con símbolos raros.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

Hace que el sitio se vea bien en dispositivos móviles, adaptando el tamaño del contenido a la pantalla.

```
<title>Breakout Game</title>
```

Título del juego. Es lo que aparece en la pestaña del navegador.

3. ESTILOS (CSS)

```
<style>
```

Aquí se define el aspecto visual del juego.
No afecta la lógica, solo la presentación.

Cuerpo principal

```css
body {
  display: flex;
  flex-direction: column;
  align-items: center;
  background: #222;
  color: #fff;
  font-family: sans-serif;
  margin: 0;
  padding: 20px;
}
```

- display: flex; → coloca los elementos de forma flexible.
- flex-direction: column; → los pone uno debajo del otro.
- align-items: center; → centra los elementos horizontalmente.
- background: #222; → color de fondo gris oscuro.
- color: #fff; → texto blanco para contraste.
- font-family: sans-serif; → tipografía simple y legible.
- padding: 20px; → espacio interno alrededor del contenido.
- margin: 0; → elimina los márgenes predeterminados del navegador.

Título

```css
h1 {
  margin-bottom: 10px;
}
```

Agrega espacio entre el título y los elementos siguientes para separar visualmente.

Información (puntuación y vidas)

```css
.info {
  margin-bottom: 10px;
}
```

Crea separación entre la información y el área del juego (canvas).

Lienzo del juego

```css
canvas {
  border: 2px solid #fff;
  background-color: #000;
}
```

- border dibuja un marco blanco alrededor del juego.
- background-color: #000 da un fondo negro que simula una pantalla retro.

Botón

```css
button {
  margin-top: 10px;
  padding: 8px 16px;
  cursor: pointer;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 4px;
}
```

Da forma moderna al botón:

- Espacio interior (padding) y superior (margin-top).
- Cursor tipo “mano” (cursor: pointer).
- Fondo verde y texto blanco.
- Sin borde visible y con esquinas redondeadas.

```css
button:hover {
  background-color: #45a049;
}
```

Cambia ligeramente el color al pasar el ratón, dando un efecto interactivo.

4. CUERPO DEL DOCUMENTO (BODY)

```
<body>
```

Aquí está todo lo visible e interactivo del juego.

Título del juego

```
<h1>Breakout</h1>
```

Muestra el nombre principal del juego en la parte superior.

Panel de información

```
<div class="info">
  <span id="score">Puntuación: 0</span>
  <span id="lives">Vidas: 3</span>
</div>
```

Muestra la puntuación y las vidas restantes.
El contenido se actualiza desde JavaScript con textContent.

Área de juego

```
<canvas id="canvas" width="800" height="400"></canvas>
```

- canvas crea el espacio donde se dibuja todo el juego (pelota, bloques, barra).
- width y height definen su tamaño.
- Es un lienzo interactivo 2D controlado por JavaScript.

Botón de reinicio

```
<button id="playBtn">Reiniciar</button>
```

Permite empezar el juego de nuevo al perder o querer reiniciar.

5. SCRIPT DEL JUEGO (JavaScript)

```
<script>
```

Aquí está toda la lógica del juego: movimiento, colisiones, reinicio y puntuación.

Obtener elementos del HTML

```
canvas {
  border: 2px solid #fff;
  background-color: #000;
}
```

- canvas obtiene el lienzo del HTML.
- ctx es el contexto gráfico 2D necesario para dibujar.

```
const scoreEl = document.getElementById('score');
const livesEl = document.getElementById('lives');
const playBtn = document.getElementById('playBtn');
```

Conecta el código con el texto de puntuación, vidas y el botón de reinicio.

Pelota

```
const ball = {
  x: canvas.width / 2,
  y: canvas.height - 30,
  size: 10,
  speed: 4,
  dx: 4,
  dy: -4,
};
```

- x, y: posición inicial.
- size: radio de la pelota.
- speed: velocidad base.
- dx y dy: dirección de movimiento (en X y Y).
- Se mueve diagonalmente desde el inicio.

Barra (paddle)

```
const paddle = {
  width: 100,
  height: 10,
  x: (canvas.width - 100) / 2,
  y: canvas.height - 20,
  speed: 7,
  dx: 0
};
```

- Define la barra que el jugador mueve.
- Comienza centrada horizontalmente.
- dx cambia con las teclas izquierda/derecha.
- speed define la rapidez del movimiento.

Ladrillos

```
const brick = {
  rowCount: 5,
  columnCount: 8,
  width: 75,
  height: 20,
  padding: 10,
  offsetX: 35,
  offsetY: 30
};
```

- Se crean 5 filas y 8 columnas de ladrillos.
- Cada ladrillo mide 75x20 px.
- padding da espacio entre ellos.
- offsetX y offsetY establecen su posición inicial.

6. CREACIÓN DE LOS LADRILLOS

```
function createBricks() {
  bricks = [];
  for (let r = 0; r < brick.rowCount; r++) {
    bricks[r] = [];
    for (let c = 0; c < brick.columnCount; c++) {
      const x = brick.offsetX + c * (brick.width + brick.padding);
      const y = brick.offsetY + r * (brick.height + brick.padding);
      bricks[r][c] = { x, y, status: 1 };
    }
  }
}
```

- Crea una matriz bidimensional (filas y columnas).
- Cada ladrillo tiene coordenadas y un status (1 = activo, 0 = destruido).

7. DIBUJAR OBJETOS

Ladrillos

```
function drawBricks() {
  for (let r = 0; r < brick.rowCount; r++) {
    for (let c = 0; c < brick.columnCount; c++) {
      let b = bricks[r][c];
      if (b.status === 1) {
        ctx.fillStyle = `rgb(${50 + r * 40}, ${100 + c * 15}, 200)`;
        ctx.fillRect(b.x, b.y, brick.width, brick.height);
      }
    }
  }
}
```

- Recorre todos los ladrillos.
- Si el ladrillo está activo (status === 1), lo dibuja con color variable.

Pelota

```
function drawBall() {
  ctx.beginPath();
  ctx.arc(ball.x, ball.y, ball.size, 0, Math.PI * 2);
  ctx.fillStyle = '#ffcc00';
  ctx.fill();
  ctx.closePath();
}
```

- Crea un círculo (la pelota) de color amarillo.

Barra

```
function drawPaddle() {
  ctx.fillStyle = '#00aaff';
  ctx.fillRect(paddle.x, paddle.y, paddle.width, paddle.height);
}
```

Dibuja la barra azul que controla el jugador.

Información

```
function drawInfo() {
  scoreEl.textContent = `Puntuación: ${score}`;
  livesEl.textContent = `Vidas: ${lives}`;
}
```

Actualiza los textos visibles según el progreso del juego

8. MOVIMIENTO Y COLISIONES

Movimiento de la barra

```
function movePaddle() {
  paddle.x += paddle.dx;
  if (paddle.x < 0) paddle.x = 0;
  if (paddle.x + paddle.width > canvas.width) {
    paddle.x = canvas.width - paddle.width;
  }
}
```

- Mueve la barra según dx.
- Evita que salga del lienzo a los lados.

Movimiento de la pelota

```
function moveBall() {
  ball.x += ball.dx;
  ball.y += ball.dy;
```

Cambia la posición de la pelota cada cuadro.

Rebote con paredes

```
if (ball.x + ball.size > canvas.width || ball.x - ball.size < 0) {
  ball.dx *= -1;
}
if (ball.y - ball.size < 0) {
  ball.dy *= -1;
}
```

La pelota rebota si toca los bordes laterales o el techo.

Rebote con la barra

```
if (ball.x > paddle.x && ball.x < paddle.x + paddle.width &&
    ball.y + ball.size > paddle.y) {
  ball.dy = -ball.speed;
}
```

Detecta si la pelota toca la barra y la devuelve hacia arriba.

Choque con ladrillos

```
for (let r = 0; r < brick.rowCount; r++) {
  for (let c = 0; c < brick.columnCount; c++) {
    let b = bricks[r][c];
    if (b.status === 1 &&
        ball.x > b.x && ball.x < b.x + brick.width &&
        ball.y - ball.size < b.y + brick.height &&
        ball.y + ball.size > b.y) {
      ball.dy *= -1;
      b.status = 0;
      score++;
    }
  }
}
```

- Si la pelota toca un ladrillo, cambia la dirección (dy) y lo destruye (status = 0).
- Incrementa la puntuación.

Pérdida de vidas

```
  if (ball.y + ball.size > canvas.height) {
    lives--;
    if (lives === 0) {
      alert('¡Juego terminado! Puntuación: ' + score);
      document.location.reload();
    } else {
      ball.x = canvas.width / 2;
      ball.y = canvas.height - 30;
      ball.dx = 4;
      ball.dy = -4;
      paddle.x = (canvas.width - paddle.width) / 2;
    }
  }
}
```

- Si la pelota toca el fondo, se resta una vida.
- Si las vidas llegan a 0, se muestra un mensaje y se recarga el juego.

9. FUNCIÓN PRINCIPAL (update)

```
function update() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  drawBricks();
  drawBall();
  drawPaddle();
  drawInfo();
  movePaddle();
  moveBall();

  if (score === brick.rowCount * brick.columnCount) {
    alert('¡Ganaste! 🎉');
    document.location.reload();
  }

  animationId = requestAnimationFrame(update);
}
```

- Limpia la pantalla.
- Dibuja todos los objetos actualizados.
- Detecta si ya se destruyeron todos los ladrillos (condición de victoria).
- requestAnimationFrame crea el bucle de animación continuo.

10. CONTROL DEL TECLADO

```
function keyDown(e) {
  if (e.key === 'ArrowRight' || e.key === 'Right') {
    paddle.dx = paddle.speed;
  } else if (e.key === 'ArrowLeft' || e.key === 'Left') {
    paddle.dx = -paddle.speed;
  }
}
```

Detecta cuando se presiona una flecha y mueve la barra.

```
function keyUp(e) {
  if (e.key === 'ArrowRight' || e.key === 'Right' ||
      e.key === 'ArrowLeft' || e.key === 'Left') {
    paddle.dx = 0;
  }
}
```

Cuando se suelta la tecla, la barra se detiene.

11. REINICIAR EL JUEGO

```
function resetGame() {
  score = 0;
  lives = 3;
  createBricks();
  ball.x = canvas.width / 2;
  ball.y = canvas.height - 30;
  ball.dx = 4;
  ball.dy = -4;
  paddle.x = (canvas.width - paddle.width) / 2;
  cancelAnimationFrame(animationId);
  update();
}
```

- Reinicia puntuación, vidas, ladrillos y posiciones.
- Detiene la animación anterior y vuelve a iniciar el juego.

12. EVENTOS PRINCIPALES

```
document.addEventListener('keydown', keyDown);
document.addEventListener('keyup', keyUp);
playBtn.addEventListener('click', resetGame);

createBricks();
update();
```

- keydown y keyup: detectan el movimiento del jugador.
- playBtn: reinicia el juego al hacer clic.
- createBricks() crea los ladrillos al iniciar.
- update() comienza la animación general.