Entorno Cliente

Salvador Jimenez Fernandez
2ºDAW

Entorno Cliente

ÍNDICE

Entorno Cliente

Descripción

Mi elección de juego para el proyecto es un clásico árcade **SPACE INVADERS.**

Se usan **Funciones , Arrays y Objetos** como se pide en los requisitos.

También se usan métodos de **Canvas** necesarios para crear el espacio y efectos visuales del juego.

El código de **Javascript** esta completamente documentado en caso de necesitar
saber que realiza cada función u objeto.

También tiene un **CSS** para darle forma visual al HTML.

A continuación describiré como funciona, los objetivos y algunas capturas del juego en funcionamiento.

Entorno Cliente

Instrucciones

Controles

A : Moverse a la izquierda
D : Moverse a la derecha
↑ : Disparar

Objetivos

Elimina a todos los aliens disparandoles.

Cada alíen eliminado te otorga 100 puntos.

Los aliens también pueden disparar y si te alcanza se acaba la partida.

Se empiezan con dos grupos de aliens y se generaran mas hasta llegar a un limite.

Si todos los aliens son eliminados tras alcanzar el limite de generación de este se gana el juego.

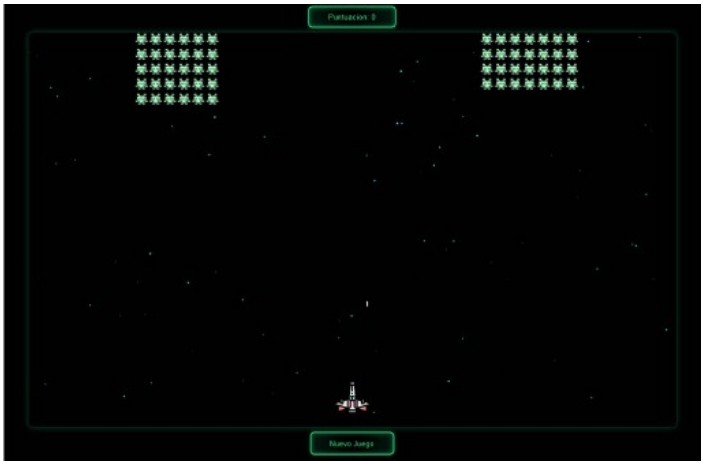
Si tanto como se gana o se pierde se puede iniciar una nueva partida pulsando el botón de NUEVA PARTIDA

Entorno Cliente

Visuales

Pantalla de Inicio

Juego



Derrota

Victoria

Salvador Jimenez Fernandez
2ºDAW
5

Entorno Cliente

Código

Clase Jugador

```
class Jugador {
  constructor() {
    // Propiedades de velocidad y posición del jugador
    this.velocidad = { x: 0, y: 0 };
    this.posicion = { x: 0, y: 0 };

    // Propiedades de rotación, opacidad y carga de la imagen del jugador
    this.rotacion = 0;
    this.opacidad = 1;
    const image = new Image();
    image.src = "./img/nave.png";
    image.onload = () => {
      const escala = 0.15;
      this.image = image;
      this.width = this.image.width * escala;
      this.height = this.image.height * escala;
      this.posicion = {
        x: canvas.width / 2 - this.width / 2,
        y: canvas.height - this.height - 20,
      };
    };

    // Partículas y frames para animación
    this.particulas = [];
    this.frames = 0;
  }

  // Método para dibujar al jugador en el canvas
  draw() {
    // Guarda el estado actual del contexto
    c.save();

    // Aplica la opacidad al dibujar al jugador
    c.globalAlpha = this.opacidad;

    // Realiza las transformaciones de traslación y rotación
    c.translate(
      this.posicion.x + this.width / 2,
      this.posicion.y + this.height / 2
    );
    c.rotate(this.rotacion);
    c.translate(
      -this.posicion.x - this.width / 2,
      -this.posicion.y - this.height / 2
    );

    // Dibuja la imagen del jugador
    c.drawImage(
      this.image,
      this.posicion.x,
      this.posicion.y,
      this.width,
      this.height
    );

    // Restaura el estado del contexto
    c.restore();
  }
```

Clase Alien

```
class Alien {
  constructor({ posicion }) {
    // Propiedades de velocidad y posición del alien
    this.velocidad = {
      x: 2,
      y: 0,
    };
    this.posicion = {
      x: 0,
      y: 0,
    };

    // Carga de la imagen del alien
    this.image = new Image();
    this.image.src = "./img/alien.png";
    this.image.onload = () => {
      const escala = 0.05;
      this.width = this.image.width * escala;
      this.height = this.image.height * escala;
      this.posicion = {
        x: posicion.x,
        y: posicion.y,
      };
    };
  }

  // Método para dibujar al alien en el canvas
  draw() {
    c.drawImage(
      this.image,
      this.posicion.x,
      this.posicion.y,
      this.width,
      this.height
    );
  }

  // Método para actualizar la posición del alien
  actualizar() {
    if (this.image) {
      this.draw();
      this.posicion.x += this.velocidad.x;
      this.posicion.y += this.velocidad.y;
    }
  }

  // Método para que el alien dispare un proyectil
  disparar(alienProyectiles) {
    alienProyectiles.push(
      new AlienProyectil({
        posicion: {
          x: this.posicion.x + this.width / 2,
          y: this.posicion.y + this.height / 2,
        },
        velocidad: {
          x: 0,
          y: 5,
        },
      })
    );
  }
}
```

Entorno Cliente

Clase Proyectil y AlienProyectil

```
// Clase Proyectil
You, hace 3 días | 1 author (You)
class Proyectil {
  constructor({ posicion, velocidad }) {
    this.posicion = posicion;
    this.velocidad = velocidad;
    this.radio = 4;
  }

  // Método para dibujar el proyectil en el canvas
  draw() {
    c.beginPath();
    c.arc(this.posicion.x, this.posicion.y, this.radio, 0, Math.PI * 2);

    c.fillStyle = "red";
    c.fill();
    c.closePath();
  }

  // Método para actualizar la posición del proyectil
  actualizar() {
    this.draw();
    this.posicion.x += this.velocidad.x;
    this.posicion.y += this.velocidad.y;
  }
}

// Clase AlienProyectil
You, hace 3 días | 1 author (You)
class AlienProyectil {
  constructor({ posicion, velocidad }) {
    this.posicion = posicion;
    this.velocidad = velocidad;

    this.width = 3;
    this.height = 10;
  }

  // Método para dibujar el proyectil del alien en el canvas
  draw() {
    c.fillStyle = "white";
    c.fillRect(this.posicion.x, this.posicion.y, this.width, this.height);
  }

  // Método para actualizar la posición del proyectil del alien
  actualizar() {
    this.draw();
    this.posicion.x += this.velocidad.x;
    this.posicion.y += this.velocidad.y;
  }
}
```

Clase Partícula

```
// Clase Particula
You, hace 3 días | 1 author (You)
class Particula {
  constructor({ posicion, velocidad, radio, color, difuminacion }) {
    this.posicion = posicion;
    this.velocidad = velocidad;

    this.radio = radio;
    this.color = color;
    this.opacidad = 1;
    this.difuminacion = difuminacion;
  }

  // Método para dibujar la partícula en el canvas
  draw() {
    c.save();
    c.globalAlpha = this.opacidad;
    c.beginPath();
    c.arc(this.posicion.x, this.posicion.y, this.radio, 0, Math.PI * 2);
    c.fillStyle = this.color;
    c.fill();
    c.closePath();
    c.restore();
  }

  // Método para actualizar la posición de la partícula
  actualizar() {
    this.draw();
    this.posicion.x += this.velocidad.x;
    this.posicion.y += this.velocidad.y;

    // Reduce la opacidad de la partícula si tiene difuminación
    if (this.difuminacion) this.opacidad -= 0.01;
  }
}
```

Entorno Cliente

Clase Cuadricula

```
// Clase Cuadricula
You, hace 3 días | 1 author (You)
class Cuadricula {
  constructor() {
    // Posición y velocidad de la cuadrícula
    this.posicion = { x: canvas.width / 2, y: 0 };
    this.velocidad = { x: 5, y: 0 };

    // Array de aliens en la cuadrícula
    this.aliens = [];

    // Genera aleatoriamente el número de columnas y líneas de aliens
    const columnas = Math.floor(Math.random() * (10 - 3) + 5);
    const lineas = Math.floor(Math.random() * (10 - 5) + 2);

    // Calcula el ancho total de la cuadrícula
    this.width = columnas * 30;

    // Crea los aliens y los añade al array
    for (let i = 0; i < columnas; i++) {
      for (let j = 0; j < lineas; j++) {
        this.aliens.push(
          new Alien({
            posicion: {
              x: this.posicion.x + i * 30 - this.width / 2,
              y: j * 30,
            },
          })
        );
      }
    }
  }

  // Método para actualizar la posición de los aliens en la cuadrícula
  actualizar() {
    this.aliens.forEach((alien) => {
      // Cambia la dirección de los aliens al llegar a los bordes del canvas
      if (
        alien.posicion.x + alien.width >= canvas.width ||
        alien.posicion.x <= 0
      ) {
        alien.velocidad.x = -alien.velocidad.x;
        alien.posicion.y += 20;
      }
      alien.actualizar();
    });
  }
}
```

Instancias,variables,fondo

```
// Creación de instancias de las clases
const jugador = new Jugador();
const cuadriculas = [new Cuadricula()];
const proyectiles = [];
const alienProyectiles = [];
const particulas = [];

// Objeto que almacena el estado de las teclas
const teclas = {
  a: { presionada: false },
  d: { presionada: false },
  ArrowUp: { presionada: false },
};

// Variables de control del juego
let frames = 0;
let framesRandom = Math.floor(Math.random() * 500) + 500;
let game = { over: false, active: true };
let puntuacion = 0;
let cuadriculasGeneradas = 0;
      You, hace 4 días • Juego terminado
// Creación de partículas para el fondo de estrellas
for (let i = 0; i < 100; i++) {
  particulas.push(
    new Particula({
      posicion: {
        x: Math.random() * canvas.width,
        y: Math.random() * canvas.height,
      },
      velocidad: { x: 0, y: 0.3 },
      radio: Math.random() * 2,
      color: "cyan",
    })
  );
}

// Función para crear partículas
function crearParticulas({ objeto, color, difuminacion }) {
  for (let i = 0; i < 15; i++) {
    particulas.push(
      new Particula({
        posicion: {
          x: objeto.posicion.x + objeto.width / 2,
          y: objeto.posicion.y + objeto.height / 2,
        },
        velocidad: {
          x: (Math.random() - 0.5) * 2,
          y: (Math.random() - 0.5) * 2,
        },
        radio: Math.random() * 3,
        color: color || "yellow",
        difuminacion: true,
      })
    );
  }
}
```

Entorno Cliente

Función Animar

```
// Función principal que anima el juego
function animar() {

  // Verifica si el jugador ha ganado la partida
  if (cuadriculas.length === 0 && game.active) {
    victoria();
    return
  }

  // Verifica si el juego está activo
  if (!game.active) {
    finDelJuego();
    return;
  }

  // Solicita la ejecución de la función animar en el próximo cuadro de animac
  requestAnimationFrame(animar);

  // Limpia el lienzo con un fondo negro
  c.fillStyle = "black";
  c.fillRect(0, 0, canvas.width, canvas.height);
      You, hace 3 días • Juego terminado y Documentado
  // Actualiza y dibuja al jugador
  jugador.actualizar();

  // Actualiza y dibuja las partículas en el fondo del juego
  particulas.forEach((particula, i) => {
    // Reposiciona las partículas que salen de la pantalla
    if (particula.posicion.y - particula.radio >= canvas.height) {
      particula.posicion.x = Math.random() * canvas.width;
      particula.posicion.y = -particula.radio;
    }

    // Elimina las partículas que han desvanecido
    if (particula.opacidad <= 0) {
      setTimeout(() => {
        particulas.splice(i, 1);
      }, 0);
    } else {
      particula.actualizar();
    }
  });

  // Actualiza y dibuja los proyectiles de los aliens
  alienProyectiles.forEach((alienProyectil, i) => {
    // Elimina los proyectiles que salen de la pantalla
    if (alienProyectil.posicion.y + alienProyectil.height >= canvas.height) {
      setTimeout(() => {
        alienProyectiles.splice(i, 1);
      }, 0);
    }
    alienProyectil.actualizar();

    // Verifica colisiones con el jugador con los proyectiles
    if (
      alienProyectil.posicion.y + alienProyectil.height >= jugador.posicion.y &&
      alienProyectil.posicion.x + alienProyectil.width >= jugador.posicion.x &&
      alienProyectil.posicion.x <= jugador.posicion.x + jugador.width
    ) {
      // Reduce la opacidad del jugador y finaliza el juego
      jugador.opacidad = 0;
      game.over = true;
      // Crea partículas de explosión alrededor del jugador
      crearParticulas({
        objeto: jugador,
        color: "white",
        difuminacion: true,
      });
      setTimeout(() => {
        game.active = false;
      }, 2000);
    }
  });

  // Actualiza y dibuja los proyectiles del jugador
  proyectiles.forEach((proyectil, indice) => {
    // Elimina los proyectiles que salen de la pantalla
    if (proyectil.posicion.y - proyectil.radio <= 0) {
      setTimeout(() => {
        proyectiles.splice(indice, 1);
      }, 0);
    } else {
      proyectil.actualizar();
    }
  });

  // Actualiza y dibuja las cuadrículas de aliens
  cuadriculas.forEach((cuadricula, cuadriculaI) => {
    cuadricula.actualizar();

    // Disparo de Aliens cada 100 frames
    if (frames % 100 === 0 && cuadricula.aliens.length > 0) {
      cuadricula.aliens[
        Math.floor(Math.random() * cuadricula.aliens.length)
      ].disparar(alienProyectiles);
    }
```

Entorno Cliente

```
// Actualiza y dibuja cada alien
cuadricula.aliens.forEach((alien, i) => {
  alien.actualizar({ velocidad: cuadricula.velocidad });

  // Verifica colisiones con los proyectiles del jugador
  proyectiles.forEach((proyectil, j) => {
    if (
      proyectil.posicion.y - proyectil.radio <=
        alien.posicion.y + alien.height &&
      proyectil.posicion.x + proyectil.radio >= alien.posicion.x &&
      proyectil.posicion.x - proyectil.radio <=
        alien.posicion.x + alien.width &&
      proyectil.posicion.y + proyectil.radio >= alien.posicion.y
    ) {
      // Elimina el alien y el proyectil, suma puntos y crea partículas de explosión
      setTimeout(() => {
        const indexAlien = cuadricula.aliens.findIndex(
          (alien2) => alien2 === alien
        );
        const indexProyectil = proyectiles.findIndex(
          (proyectil2) => proyectil2 === proyectil
        );

        if (indexAlien !== -1 && indexProyectil !== -1) {
          puntuacion += 100;
          puntuacionJugador.innerHTML = puntuacion;
          crearParticulas({
            objeto: alien,
            difuminacion: true,
          });
          cuadricula.aliens.splice(indexAlien, 1);
          proyectiles.splice(indexProyectil, 1);

          // Ajusta el tamaño de la cuadrícula si quedan aliens
          if (cuadricula.aliens.length > 0) {
            const primerAlien = cuadricula.aliens[0];
            const ultimoAlien =
              cuadricula.aliens[cuadricula.aliens.length - 1];

            cuadricula.width =
              ultimoAlien.posicion.x -
              primerAlien.posicion.x +
              ultimoAlien.width;
            cuadricula.posicion.x = primerAlien.posicion.x;
          } else {
            cuadriculas.splice(cuadriculaI, 1);
          }
        }
      }, 0);
    }
  });
});
});
```

```
// Control de movimiento del jugador según las teclas presionadas
if (teclas.a.presionada && jugador.posicion.x >= 0) {
  jugador.velocidad.x = -7;
  jugador.rotacion = -0.15;
} else if (
  teclas.d.presionada &&
  jugador.posicion.x + jugador.width <= canvas.width
) {
  jugador.velocidad.x = 7;
  jugador.rotacion = 0.15;
} else {
  jugador.velocidad.x = 0;
  jugador.rotacion = 0;
}



// Generación de nuevas cuadrículas de aliens cada ciertos cuadros
if (frames % framesRandom === 0 && cuadriculasGeneradas < 5) {
  cuadriculas.push(new Cuadricula());
  framesRandom = Math.floor(Math.random() * 500) + 500;
  frames = 0;
  cuadriculasGeneradas++;
}

frames++;
}
```

Entorno Cliente

Eventos

```
// Evento al presionar una tecla
addEventListener("keydown", ({ key }) => {
  if (game.over) return;

  switch (key) {
    case "a":
      teclas.a.presionada = true;
      break;
    case "d":
      teclas.d.presionada = true;
      break;
    case "ArrowUp":
      // Disparo de proyectil al presionar la tecla de flecha hacia arriba
      teclas.ArrowUp.presionada = true;
      proyectiles.push(
        new Proyectil({
          posicion: {
            x: jugador.posicion.x + jugador.width / 2,
            y: jugador.posicion.y,
          },
          velocidad: {
            x: 0,
            y: -10,
          },
        })
      );
      break;
    default:
      break;
  }
});

// Evento al soltar una tecla
addEventListener("keyup", ({ key }) => {
  switch (key) {
    case "a":
      teclas.a.presionada = false;
      break;
    case "d":
      teclas.d.presionada = false;
      break;
    default:
      break;
  }
});      You, hace 3 semanas • 21 de Noviembre, Entorno Cliente
```

Función Reiniciar

```
// Función para reiniciar el juego al hacer clic en un botón
function reiniciar() {
  // Comprueba si el juego ha terminado
  if (!game.over) return;

  // Restablece las variables del juego
  game.over = false;
  game.active = true;
  puntuacion = 0;
  puntuacionJugador.innerHTML = puntuacion;

  // Restablece la posición y el estado del jugador
  jugador.posicion = {
    x: canvas.width / 2 - jugador.width / 2,
    y: canvas.height - jugador.height - 20,
  };
  jugador.velocidad = {
    x: 0,
    y: 0,
  };
  jugador.rotacion = 0;
  jugador.opacidad = 1;

  // Limpia los arrays de proyectiles, proyectiles de aliens y cuadrículas
  proyectiles.length = 0;
  alienProyectiles.length = 0;
  cuadriculas.length = 0;

  // Restablece los contadores de cuadros y cuadros aleatorios
  frames = 0;
  framesRandom = Math.floor(Math.random() * 500) + 500;

  // Restablece el estado de las teclas
  teclas.a.presionada = false;
  teclas.d.presionada = false;
  teclas.ArrowUp.presionada = false;

  // Restablece la opacidad de las partículas
  particulas.forEach((particula) => {
    particula.opacidad = 1;
  });

  // Restablece el contador de cuadrículas generadas
  cuadriculasGeneradas = 0;

  // Restablece la velocidad de los aliens y genera nuevas cuadrículas
  for (let i = 0; i < 1; i++) {
    cuadriculas.push(new Cuadricula());
  }

  // Elimina el mensaje de fin del juego del lienzo
  c.clearRect(0, 0, canvas.width, canvas.height);

  // Vuelve a iniciar el bucle de animación
  animar();
}
```

Salvador Jimenez Fernandez
2ºDAW
11

Entorno Cliente

Pantalla Inicio, Funciones comenzarJuego, findelJuego y victoria

```
// Obtiene el elemento de la pantalla de inicio del HTML
const pantallaInicio = document.getElementById("pantallaInicio");

// Función para comenzar el juego al hacer clic en la pantalla de inicio
function comenzarJuego() {
  pantallaInicio.style.display = "none";
  animar();
}

// Agrega un evento de clic a la pantalla de inicio para comenzar el juego
pantallaInicio.addEventListener("click", comenzarJuego);

// Función para mostrar el mensaje de fin del juego en el lienzo
function finDelJuego() {
  c.fillStyle = "white";
  c.font = "bold 40px Arial";
  c.fillText("FIN DEL JUEGO", canvas.width / 2 - 150, canvas.height / 2);
}

// Función para mostrar el mensaje de victoria en el lienzo
function victoria() {
  c.fillStyle = "white";
  c.font = "bold 40px Arial";
  c.fillText("VICTORIA", canvas.width / 2 - 150, canvas.height / 2);
  game.active = false;
  game.over = true;
}      You, hace 3 días • Juego terminado y Documentado
```

Salvador Jimenez Fernandez
2ºDAW

Entorno Cliente

Conclusión

La creación de este juego fue mucho mas esfuerzo del que pensaba en un principio.

El manejo de objetos y arrays es muy extenso pese a ser un juego que a simple vista es muy simple.

A tal nivel que cualquier fallo hace que no funcione o se meta en bucles infinitos causando problemas de rendimiento.

También el tener que trabajar con unos métodos como los de canvas es una dificultad añadida pues hasta ahora no lo había visto.

Aun se pueden añadir muchas mas funciones e utilidades pero en este estado se puede considerar un juego.