Blob.io - Juego Multijugador

Un juego estilo agar.io desarrollado como proyecto para el curso CI-0137 Desarrollo Web de la Universidad de Costa Rica.

Descripción del Proyecto

Blob.io es un juego multijugador en tiempo real donde los jugadores controlan células (blobs) que deben crecer comiendo alimento y otros jugadores más pequeños. El objetivo es alcanzar el mayor tamaño posible y dominar la tabla de posiciones.

Iteración #2 - Frontend

Esta iteración se enfoca en la implementación del frontend del juego con las siguientes características:

Características Implementadas

- **Mecánicas de Juego Básicas**
    - Movimiento fluido del jugador siguiendo el cursor
    - Sistema de crecimiento al comer alimento
    - Colisiones entre jugadores y alimento
    - Física realista con masa y velocidad
- **Interfaz de Usuario**
    - Menú principal con entrada de nombre
    - HUD del juego con puntuación y controles
    - Tabla de posiciones en tiempo real
    - Minimapa para navegación
    - Menús de pausa y game over
- **Inteligencia Artificial**
    - Bots AI para demostración del juego
    - Comportamiento inteligente (buscar comida, evitar peligros)
    - Respawn automático de bots
- **Diseño Responsivo**
    - Adaptable a diferentes tamaños de pantalla
    - Controles táctiles para dispositivos móviles
    - Optimizaciones para rendimiento
- **Herramientas de Desarrollo**
    - Modo debug con información detallada
    - Modo god para testing
    - Monitor de rendimiento

- Manejo de errores robusto

Tecnologías Utilizadas

- **Frontend**: HTML5, CSS3, JavaScript (ES6+)
- **Backend**: Node.js, Express.js, WebSocket
- **Gráficos**: Canvas API
- **Arquitectura**: Modular con clases ES6

Estructura del Proyecto

```
proyecto-blob.io/
├── public/
│   ├── index.html          # Página principal
│   ├── css/
│   │   └── styles.css      # Estilos del juego
│   └── js/
│       ├── main.js         # Punto de entrada
│       ├── game.js         # Motor del juego
│       ├── player.js       # Lógica del jugador
│       ├── food.js         # Sistema de alimentos
│       └── ui.js           # Interfaz de usuario
├── server.js               # Servidor Node.js
├── package.json            # Dependencias del proyecto
└── README.md              # Este archivo
```

Instalación y Ejecución

Requisitos Previos

- Node.js (versión 16 o superior)
- npm (incluido con Node.js)

Pasos de Instalación

1. **Clonar el repositorio**

```
git clone https://github.com/sebasbose/proyecto-blob.io.git
cd proyecto-blob.io
```

2. **Instalar dependencias**

```
npm install
```

3. **Ejecutar el servidor**

```
npm start
```

Para desarrollo con auto-reload:

```
npm run dev
```

4. **Abrir el juego**
    - Navega a `http://localhost:3000` en tu navegador
    - ¡Disfruta jugando!

Cómo Jugar

1. **Inicio del Juego**
    - Iniciar sesión en la plataforma del juego
    - Haz clic en "JUGAR" para comenzar
2. **Controles**
    - **PC**: Mueve el mouse para dirigir tu blob
    - **Móvil**: Toca la pantalla donde quieres moverte
    - **Pausa**: Presiona ESPACIO o el botón de pausa
    - **Salir**: Presiona ESC o el botón de salida
3. **Objetivos**
    - Come puntos de colores para crecer
    - Evita blobs más grandes que puedan comerte
    - Intenta alcanzar el primer lugar en la tabla de posiciones

Comandos de Desarrollo

Durante el desarrollo, puedes usar estos atajos de teclado:

- `Ctrl+Shift+D`: Activar/desactivar modo debug
- `Ctrl+Shift+G`: Activar/desactivar modo god
- `Ctrl+Shift+F`: Generar comida adicional
- `Ctrl+Shift+P`: Mostrar información de rendimiento
- `Ctrl+Shift+R`: Recargar página con confirmación

Próximas Iteraciones

Iteración #3 - Multijugador (backend)

- Implementación completa del backend WebSocket
- Sincronización en tiempo real entre jugadores
- Salas de juego privadas

- Sistema de chat

Arquitectura del Código

Clases Principales

- `Game`: Motor principal del juego, maneja el bucle de juego y la lógica general
- `Player`: Representa a un jugador, maneja movimiento, crecimiento y renderizado
- `FoodManager`: Gestiona la generación y actualización de alimentos
- `UI`: Maneja toda la interfaz de usuario y eventos

Patrones de Diseño Utilizados

- **Module Pattern**: Cada archivo JS es un módulo independiente
- **Observer Pattern**: Sistema de eventos para comunicación entre componentes
- **Component Pattern**: Separación clara de responsabilidades
- **Factory Pattern**: Generación de objetos de juego

Contribución

Este es un proyecto académico desarrollado para CI-0137 Desarrollo Web.

Autor: Sebastian Bolaños Serrano **Institución**: Universidad de Costa Rica
Curso: CI-0137 Desarrollo Web