MOKEPON WORLD

Informe de Desarrollo y Analisis

Un juego inspirado en Pokemon

Generado el 12 de April, 2025

Informe Detallado: Desarrollo de Mokepon World

Generado: 12/04/2025 06:23

Tabla de Contenidos

Informe Detallado: Desarrollo de Mokepon World

Generado: 12/04/2025 06:23

1. Introduccion

Mokepon World es un juego inspirado en Pokemon que combina elementos de estrategia, combate por turnos y exploracion de mapas. Este informe detalla el analisis tecnico del proyecto, junto con una explicacion de la metodologia en cascada (Waterfall) que se aplica al desarrollo.

El juego permite a los jugadores explorar diferentes mapas, capturar criaturas llamadas "Mokepons" y participar en batallas contra entrenadores controlados por la IA.

Cada Mokepon tiene tipos especificos y ataques unicos, creando un sistema de combate estrategico con ventajas y desventajas de tipo.

Este documento examina la estructura actual del codigo, los patrones de diseno implementados y propone optimizaciones para mejorar tanto el rendimiento como la mantenibilidad del codigo.

2. Metodologia en Cascada

Definicion y Etapas

La metodologia en cascada (Waterfall) es un proceso de desarrollo secuencial que avanza a traves de fases distintas con resultados especificos en cada etapa. Es especialmente adecuada para proyectos con requisitos bien definidos y pocos cambios esperados durante el desarrollo.

Etapas de la Metodologia

- Recopilacion y analisis de requisitos: Se identifican y documentan todas las necesidades del cliente y alcance del proyecto.
- Diseno del sistema: Se crea un diseno detallado incluyendo arquitectura, componentes y flujos de datos.
- Implementacion: Se desarrolla el codigo basado en las especificaciones de diseno.
- Verificacion: Se realizan pruebas exhaustivas para garantizar que el software cumple con los requisitos.
- Despliegue: El sistema se entrega al cliente y se implementa en el entorno de produccion.
- Mantenimiento: Se realizan actualizaciones, correcciones y mejoras segun sea necesario.

Ventajas de la Metodologia en Cascada para Mokepon World

- Estructura clara: Proporciona un marco organizado para el desarrollo del juego.
- Documentacion completa: Facilita la comprension del proyecto para nuevos desarrolladores.
- Planificacion eficiente: Permite estimar tiempos y recursos con mayor precision.
- Definicion temprana: Los elementos del juego (mokepons, ataques, mapas) pueden definirse claramente desde el inicio.

3. Analisis del Proyecto Mokepon World

Vision General

Mokepon World es un proyecto de juego que combina HTML, CSS y JavaScript para crear una experiencia interactiva basada en navegador. El juego implementa una estructura de objetos orientada a clases para representar entidades del juego, sistemas de animacion para los enemigos, y una interfaz de usuario para navegacion entre mapas.

Componentes Principales

- Interfaz de usuario: Pagina principal con seleccion de mapas y sistema de navegacion.
- Sistema de animacion: Mecanismo para mover enemigos en patrones predefinidos.
- Sistema de clases: Definicion de Mokepons y ataques usando programacion orientada a objetos.
- Logica de combate: Sistema para determinar resultados de batallas basado en tipos y estadisticas.

Interfaz de Usuario

La interfaz principal presenta un diseno de cuadricula con seis mapas seleccionables. Cada opcion de mapa muestra una imagen representativa, un titulo y un boton de acceso. La navegacion superior incluye un titulo del juego y espacio para opciones adicionales.

Informe Detallado: Desarrollo de Mokepon World

Generado: 12/04/2025 06:23

4. Estructura del Codigo

Clases Principales

- Clase Mokepon:

  Define las criaturas del juego con propiedades como imagen, ataques, posicion, dimensiones e identificador unico.

JavaScript - Clase Mokepon

```
class Mokepon {
    constructor(image, input, label, name, x, y, width, height, idMk) {
        this.image = image;
        this.input = input;
        this.atacks = [];
        this.label = label;
        this.name = name;
        this.type = [];
        this.x = x;
        this.y = y;
        this.width = 70;
        this.height = 70;
        this.idMk = randomMonster(1, 1000000);
    }
}
```

- Clase Atack:

  Define los ataques disponibles para los Mokepons, incluyendo nombre, ID, clase CSS, color e identificador de ataque.

JavaScript - Clase Atack

```
class Atack {
    constructor(atkName, id, classMk, color, atackId) {
        this.atkName = atkName;
        this.id = id;
        this.classMk = classMk;
        this.color = color;
        this.atackId = atackId;
    }
}
```

Informe Detallado: Desarrollo de Mokepon World

Generado: 12/04/2025 06:23

Sistema de Animacion de Enemigos

El juego implementa un sofisticado sistema de animacion para mover enemigos en patrones lineales predefinidos. Cada enemigo tiene funciones especificas que controlan su movimiento usando setInterval y setTimeout para crear ciclos de animacion continuos.

JavaScript - Sistema de Animacion

```
function moveEnemies1() {
    intervalsIdList.repeaterLists = setInterval(() => {
        mapEnemies[0].x += speedEnemy;
        checkEnemyColisions();
    }, 50);

    setTimeout(() => {
        clearInterval(intervalsIdList.repeaterLists);
        requestAnimationFrame(moveEnemiesReverse1);
    }, 2200);
}

function moveEnemiesReverse1() {
    intervalsIdList.repeaterReverseLists = setInterval(() => {
        mapEnemies[0].x -= speedEnemy;
        checkEnemyColisions();
    }, 50);

    setTimeout(() => {
        clearInterval(intervalsIdList.repeaterReverseLists);
        requestAnimationFrame(moveEnemies1);
    }, 2200);
}
```

Cada enemigo tiene una funcion especifica para renderizarlo en el canvas del mapa, utilizando su imagen y posicion actual:

JavaScript - Renderizado de Enemigos

```
function mapEnemySetter1() {
    enemyImage.src = enemyCrabster.image;
    mapDimension.drawImage(
        enemyImage,
        enemyCrabster.x,
        enemyCrabster.y,
        enemyCrabster.width,
        enemyCrabster.height
    );
}
```

5. Patrones de Diseno Implementados

Patron Constructor

El proyecto utiliza el patron Constructor a traves de clases JavaScript para crear instancias de Mokepons y Ataques. Este patron permite la creacion estandarizada de objetos con propiedades consistentes.

Patron de Modulo

El codigo esta organizado en funciones modulares que encapsulan comportamientos especificos, como la animacion de enemigos y la renderizacion de elementos en el mapa.

Patron de Estado

Las animaciones de los enemigos implementan un patron de estado simple donde cada enemigo alterna entre dos estados (movimiento normal y movimiento inverso) basado en temporizadores.

6. Optimizaciones Recomendadas

Refactorizacion del Sistema de Animacion

El sistema actual de animacion de enemigos contiene codigo repetitivo. Se recomienda implementar una funcion generica que pueda manejar las animaciones para todos los enemigos:

JavaScript - Animacion Refactorizada

```
function createEnemyAnimation(enemy, property, direction, duration) {
    const intervalId = setInterval(() => {
        enemy[property] += speedEnemy * direction;
        checkEnemyColisions();
    }, 50);

    setTimeout(() => {
        clearInterval(intervalId);
        createEnemyAnimation(enemy, property, -direction, duration);
    }, duration);

    return intervalId;
}

// Uso:
function startAnimations() {
    intervalsIdList.enemy1 = createEnemyAnimation(mapEnemies[0], 'x', 1, 2200);
    intervalsIdList.enemy2 = createEnemyAnimation(mapEnemies[1], 'x', -1, 2400);
    // etc.
}
```

Mejora del Sistema de Renderizado

Similar al sistema de animacion, las funciones de renderizado pueden consolidarse en una unica funcion generica:

JavaScript - Renderizado Refactorizado

```
function renderEnemy(enemy) {
    enemyImage.src = enemy.image;
    mapDimension.drawImage(
        enemyImage,
        enemy.x,
        enemy.y,
        enemy.width,
        enemy.height
    );
}

function renderAllEnemies() {
    mapEnemies.forEach(renderEnemy);
    // Render copied enemies too
    renderEnemy(copiedEnemyRaykiou);
    renderEnemy(copiedEnemyJoka);
    // etc.
}
```

Mejoras a la Pagina Principal

La interfaz de usuario principal puede mejorarse con las siguientes adiciones: - Imagenes

reales para las miniaturas de los mapas - Enlaces funcionales para cada mapa - Opciones de navegacion completas en la barra superior - Estilos CSS mejorados para mayor atractivo visual - Implementacion de un diseno responsivo para diferentes dispositivos

Implementacion de un Sistema de Combate Mejorado

El sistema de combate podria mejorarse mediante: - Tabla de ventajas/desventajas de tipo para simplificar la logica de combate - Animaciones y efectos visuales durante las batallas - Mecanicas de debilitamiento y curacion de Mokepons - Sistema de experiencia y nivel para Mokepons capturados - Mayor variedad de ataques con diferentes efectos

7. Conclusion

Mokepon World muestra un prometedor diseno de juego inspirado en Pokemon con mecanicas interesantes de combate y exploracion. El proyecto implementa conceptos importantes de programacion orientada a objetos y animacion basada en tiempo. La aplicacion de la metodologia Waterfall beneficia al proyecto al proporcionar una estructura clara para el desarrollo, con fases bien definidas que permiten una planificacion eficiente y una documentacion completa. Las optimizaciones propuestas pueden mejorar significativamente la mantenibilidad y escalabilidad del codigo:

1. La refactorizacion del sistema de animacion eliminara redundancias

2. La mejora del sistema de renderizado simplificara la adicion de nuevos enemigos

3. Las actualizaciones de la interfaz de usuario mejoraran la experiencia del jugador Implementando estas mejoras, Mokepon World puede evolucionar en un juego mas robusto y atractivo, manteniendo al mismo tiempo una base de codigo limpia y eficiente que facilitara futuras expansiones.