Logotipo

Descripción generada automáticamente con confianza media

**LeagueTEC**

Joseph Daniel Salas Rivas

Fabian Ricardo Vargas Araya

María Laura Alpízar Rodríguez

Instituto Tecnológico de Costa Rica, Unidad de Computación

Programación orientada a objetos

Jonathan Andres Solis Parajeles

9 de octubre de 2024

**Tabla de contenido**

Descripción del problema…………………………………………………………………………3

Análisis del problema……………………..………………………………………………………6

Solución del problema…………………………………………………………………………….7

**Descripción del problema**

La empresa TEC-Games ha solicitado el desarrollo de un juego llamado *LeagueTEC*, que estará basado en un sistema de batalla entre personajes y será implementado en el lenguaje de programación Java. Este proyecto tiene como objetivo emular en un ambiente virtual una batalla entre personajes con diferentes habilidades y atributos, aplicando principios fundamentales de la Programación Orientada a Objetos (POO). El sistema debe permitir que dos jugadores se enfrenten en arenas predefinidas, controlando personajes que pueden moverse, atacar, y utilizar habilidades especiales, mientras buscan destruir las torres del oponente para ganar la partida.

**Elementos del juego:**

1. **Arenas de juego**:

* Cada partida se desarrollará en una arena, que puede ser diseñada por los usuarios o seleccionada de una lista predeterminada de cuatro tipos: fuego, agua, tierra y aire.
* Las arenas tienen dimensiones mínimas de 10x10 (10 columnas y 10 filas), con la mitad del terreno asignado a cada jugador.
* En cada arena, ambos equipos deben contar con un mínimo de una torre y tres personajes por equipo.

1. **Personajes**:

* Cada jugador tiene la capacidad de crear, editar o eliminar personajes. Un personaje tiene varios atributos como puntos de vida, maná, daño, y defensa.
* Los personajes se desplazan por la arena y atacan al oponente con una cantidad limitada de movimientos por turno.
* Los ataques de los personajes están basados en su poder ofensivo y defensivo, y pueden verse incrementados según el tipo de arena (elemento vital) en el que se encuentren.
* Al derrotar a otro personaje o destruir una torre enemiga, el personaje aumenta de nivel, incrementando sus atributos en un 25%.
* Si un personaje es derrotado, debe esperar un turno completo para renacer en una de las torres de su equipo.

1. **Habilidades**:

* Los personajes pueden usar habilidades especiales que consumen maná.
* Las habilidades están asociadas a un solo personaje y pueden variar según el tipo de personaje y su elemento vital.
* El maná se regenera a un ritmo del 25% por cada turno.

1. **Torres**:

* Las torres juegan un papel crucial en la defensa de cada equipo. Cada equipo debe tener al menos una torre en su lado del campo.
* Las torres tienen una resistencia predeterminada de 10,000 puntos de vida.
* El objetivo final de cada equipo es destruir todas las torres del oponente, ya que esto determinará el ganador de la partida.

1. **Sistema de turnos**:

* El turno se define de manera aleatoria y finaliza una vez que el jugador ha utilizado las habilidades de su personaje o ha atacado a otro personaje o torre.
* Si un jugador logra derrotar a un personaje o destruir una torre durante su turno, puede continuar actuando hasta que no le queden más acciones disponibles.

1. **Movimiento y ataque**:

* El desplazamiento de los personajes es limitado a las direcciones de avance, arriba o abajo en la arena.
* Los personajes pueden moverse, pasar o usar una habilidad como parte de sus acciones durante el turno.

1. **Final de la partida**:

* La partida termina cuando uno de los jugadores logra destruir todas las torres del oponente.
* Aunque los personajes pueden morir múltiples veces, las torres solo pueden ser destruidas una vez, y su destrucción es irreversible.

1. **Estadísticas**:

* El sistema debe registrar estadísticas detalladas como el número de partidas jugadas, ganadas y perdidas por cada jugador, además de contar el número de muertes de personajes y torres durante las partidas.
* También se debe llevar un registro del rendimiento de los jugadores con respecto a las partidas jugadas.

**Restricciones del problema:**

* El terreno de juego está limitado a un mínimo de 10x10 casillas.
* Cada equipo debe tener al menos una torre y tres personajes en la arena.
* Los movimientos de los personajes están limitados a avanzar, subir o bajar en el tablero.
* Las habilidades deben consumir una cantidad específica de maná y regenerarse en un 25% por turno.
* El tipo de arena (fuego, agua, aire, tierra) influye en la efectividad de los ataques de los personajes.

**Desafíos que enfrentar:**

* Definir una estructura de clases adecuada que represente a personajes, habilidades, torres y arenas, aplicando correctamente los principios de encapsulación, herencia y polimorfismo.
* Diseñar un sistema eficiente de turnos y movimiento que permita a los jugadores interactuar en tiempo real con el juego.
* Implementar una interfaz gráfica de usuario que facilite la interacción con el sistema de manera amigable e intuitiva.

El proyecto requiere no solo el diseño y la implementación de las clases necesarias, sino también la integración de una interfaz gráfica que permita a los jugadores interactuar con el sistema de forma visual, así como la validación de la jugabilidad mediante la ejecución de pruebas exhaustivas.

**Análisis del Problema**

Para la correcta implementación del juego, se identifican varios desafíos técnicos que deben abordarse de manera efectiva:

**Definición y Organización de Clases**

El diseño de LeagueTEC debe estructurarse alrededor de un sistema de clases que capture todos los elementos del juego, asegurando la cohesión, encapsulación, y reusabilidad de código mediante la POO. Los elementos principales incluyen:

1. Clase Character: Representa a los personajes jugables, con atributos como puntos de vida, maná, daño, defensa y métodos para atacar, moverse, y utilizar habilidades especiales. Es crucial que esta clase soporte herencia para crear subclases específicas (e.g., personajes de fuego, agua, etc.) con habilidades y bonificaciones únicas.
2. Clase Skill: Define las habilidades especiales que los personajes pueden utilizar. Cada habilidad debe consumir una cantidad específica de maná y tener efectos variables dependiendo del tipo de personaje y el entorno de la arena. Es importante que esta clase gestione los efectos visuales y de impacto en los personajes y torres.
3. Clase Tower: Representa las estructuras defensivas de cada equipo. Las torres tienen atributos de vida y métodos para recibir daño, proteger a los personajes y definir el punto de respawn de los personajes derrotados.
4. Clase Arena: Gestiona el entorno de juego, definiendo el campo de batalla y aplicando efectos según el tipo de arena (fuego, agua, tierra, aire). Debe influir en las estadísticas de los personajes, potenciando o debilitando sus habilidades.
5. Clase Match: Controla el flujo general del juego, incluyendo la inicialización de partidas, la gestión de turnos, y la interacción entre personajes, torres y habilidades.

**Gestión de Turnos y Movimientos**

El sistema de turnos es fundamental para la mecánica del juego. Este sistema debe manejar de manera justa y eficiente las acciones de los jugadores, permitiendo que cada uno pueda realizar movimientos estratégicos. Las acciones disponibles incluyen moverse, atacar, y usar habilidades. El movimiento está restringido a las dimensiones de la arena y debe ser implementado de manera que permita decisiones tácticas, como posicionar personajes de forma ventajosa o evitar ataques enemigos.

* 1. Secuencia de Turnos: La asignación de turnos debe ser aleatoria al inicio del juego y seguir un orden lógico. Cada turno debe permitir un número limitado de movimientos y acciones, que se agotan al atacar, usar habilidades o finalizar el turno.
  2. Resurrección y Respawn: Los personajes derrotados deben esperar un turno completo para reaparecer en una de las torres de su equipo, lo que añade una capa estratégica al juego, obligando a los jugadores a proteger sus torres.

**Interacción con el Entorno y Bonificaciones de Arena**

El tipo de arena (fuego, agua, tierra, aire) debe influir significativamente en la jugabilidad, otorgando bonificaciones o penalizaciones a los personajes dependiendo de su tipo. Por ejemplo, un personaje de fuego puede recibir bonificaciones en una arena de fuego pero sufrir penalizaciones en una de agua. La implementación de estos efectos debe ser dinámica y gestionada por la clase Arena, modificando los atributos y habilidades de los personajes en tiempo real.

**Registro y Control de Estadísticas**

El sistema de estadísticas debe ser robusto y detallado, proporcionando a los jugadores información sobre su desempeño en el juego. Las estadísticas deben incluir:

Número de partidas jugadas, ganadas y perdidas.

Rendimiento individual de cada personaje.

Número de muertes y resurrecciones de personajes.

Torres destruidas y defendidas.

Estas estadísticas no solo mejoran la experiencia de usuario al permitir seguimiento del progreso, sino que también son útiles para balancear el juego y ajustar atributos según el rendimiento de los jugadores.

**Diseño de la Interfaz Gráfica de Usuario (GUI)**

Una interfaz gráfica de usuario intuitiva y amigable es esencial para la jugabilidad. Debe incluir:

Tablero de Juego: Representación visual de la arena, personajes y torres, con actualizaciones en tiempo real del estado de cada elemento.

Panel de Control: Herramientas para mover personajes, usar habilidades, y ver estadísticas.

Indicadores de Estado: Visualización clara de puntos de vida, maná, y habilidades disponibles de cada personaje.

**Solución del Problema**

Para resolver el problema, se propone la siguiente estructura y lógica:

1. **Estructuras de Clases**:

* Se crearán clases como Personaje, Habilidad, Torre y Arena.
* La clase Personaje tendrá atributos como vida, maná, daño y defensa, y métodos para atacar y moverse.
* La clase Habilidad definirá las habilidades especiales, su costo de maná y efectos.
* Torre representará las estructuras que los jugadores deben proteger y destruir.
* Arena gestionará el entorno de juego y el impacto en las habilidades.

1. **Lógica de Interacción**:

* **Turnos**: El sistema define de forma aleatoria el turno de cada jugador, y cada jugador puede mover o atacar hasta agotar sus acciones disponibles.
* **Movimiento y Ataque**: Los personajes se mueven en un área limitada (10x10) y pueden atacar torres o personajes enemigos.
* **Habilidades**: Las habilidades dependen del tipo de personaje y se potencian o reducen según la arena en la que se encuentren.

1. **Recorrido de Consultas y Operaciones**:

* **Creación de Personajes y Torres**: Los jugadores pueden crear personajes con atributos personalizados y seleccionar o diseñar la arena de juego.
* **Registro de Estadísticas**: Al finalizar cada partida, se almacenan las estadísticas en una base de datos para futuras consultas y análisis del rendimiento.
* **Simulación de la Partida**: La lógica del juego se desarrolla en bucles de turnos, permitiendo a los jugadores tomar decisiones estratégicas en tiempo real.

1. **Interfaz de Usuario**:

* Se implementa una interfaz gráfica que permite a los jugadores interactuar con los elementos del juego de manera intuitiva, visualizando el estado de cada personaje y las torres en tiempo real.

**Bitácora de la Etapa I:**

**Objetivo de la Etapa I:**

**Comprender el problema y desarrollar el diagrama de clases.**

**Actividades realizadas:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actividad** | **Descripción** | **Responsable** | **Duración** |
| **Análisis del enunciado** | Estudio y análisis detallado de los requisitos del proyecto. | Todos | 1 hora |
| **Diagrama de clases** | Identificación de clases, atributos, y relaciones necesarias. | Fabián y Laura | 3 horas |
| **Revisión y ajustes** | Revisión del diagrama de clases para asegurar coherencia. | Todos | 1 hora |

**Total, de horas invertidas: 5 horas**

**Bitácora de la Etapa II:**

**Objetivo de la Etapa II:**

**Comenzar a escribir las clases en Java, analizar el problema planteado y diseñar una solución**

**Actividades realizadas:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actividad** | **Descripción** | **Responsable** | **Duración** |
| **Desarrollo del documento** | Estudio y análisis detallado de los requisitos del proyecto. | Laura | 1 hora |
| **Clases y relaciones en Java** | Identificación de clases, atributos, y relaciones necesarias. | Fabian | 3 horas |

**Total, de horas invertidas: 4 horas**