Scripting, 2023 01

Avance 01, proyecto de curso

Grupo: David Marin Yepes, Rey Valentin Arias & Ana María Díaz.

---

1. GITHUB
    - Diagrama de clases
    - Implementación inicial

2. Descripción de clases:

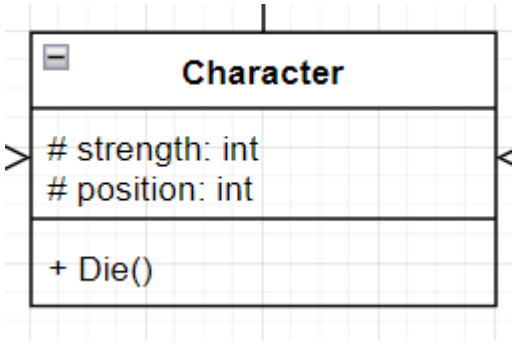
- Descripción de las clases, interfaces, etc, que participarán de su solución.
- En dicha descripción estén definidos los atributos y métodos de cada clase, las firmas de interfaces y demás elementos necesarios

Texto, jugabilidad:

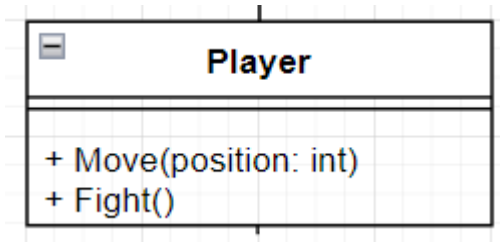
El héroe debe salvar a la doncella del último piso de la torre, para esto debe vencer cada obstáculo por piso, esto es, otros personajes con una puntuación en específico. Para derrotarlos el héroe debe ir a su piso y garantizar que tiene un puntaje mayor al de dicho personaje (Si va a su piso con un menor puntaje, el héroe pierde). Sobre el puntaje: los obstáculos lo tienen por preestablecido mientras que el héroe lo adquiere por cada nueva victoria (Y es acumulativo). Más aún, por cada movimiento del héroe algo más sucede simultáneamente en su entorno (Ejemplo, otra torre emerge o aparecen más collectibles). El propósito de este videojuego de plataforma es que el jugador debe ser atento con los puntajes y estratégico en sus movimientos, y por nivel deberá salvar múltiples doncellas en distintas torres. *// Palabra clave: Merge Puzzle. //Key visual: click.*

Clases: *Atributos: (Valores que almacenan) & Métodos: (Operaciones posibles)*

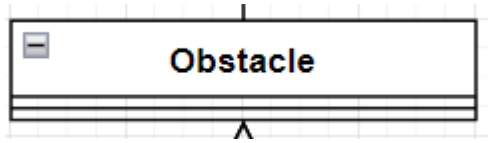
- Personaje: Sus atributos serán su posición actual (protected int position) y su número de fuerza (protected int strength), según esto se determinará si sigue en la partida o si muere, derrotado por algún enemigo (public void Die()).



- Jugador: Éste tiene dos posibles operaciones, la primera es moverse, cambiando así la posición actual del héroe (public void Move(int position)). Y la segunda es atacar. Para atacar el jugador debe desplazar al héroe hasta algún piso donde esté un obstáculo y aquí el juego observará si vence (Tiene mayor puntaje que la del obstáculo) o si pierde (El obstáculo tiene más puntos que el héroe) (public void Fight()).

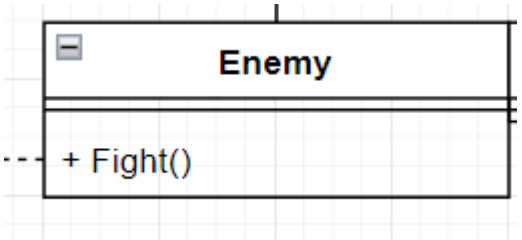


- Obstáculos:

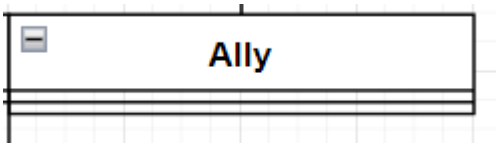


-

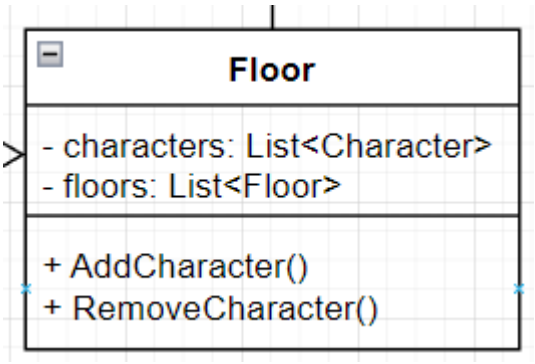
• Enemigos: El enemigo responde al ataque del héroe y proporcionado por el jugador. Si el enemigo gana en puntaje, el héroe muere; pero si el pierde en puntaje, el enemigo y su piso desaparece (Haciendo que el héroe esté un paso más cerca de salvar a la doncella y con esto, de ganar todo el juego) (public void Fight()).



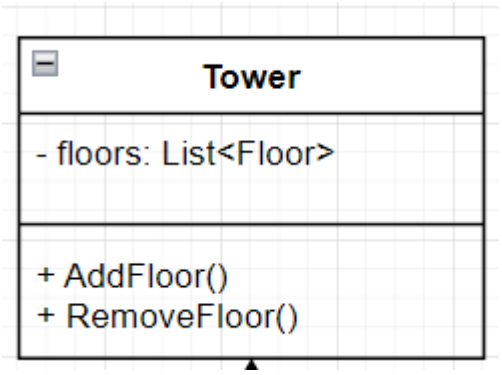
•Aliados (coleccionables): Conforme avanza el nivel, en el juego podrán aparecer power ups u otros peligros para ayudar o impedir, respectivamente, que el héroe logre la gran hazaña.



- Piso: Su atributo responde a si tiene un personaje en aquel piso o no (private List<Floor> floors) y (private List<Floor> floors), y su método responde a sí sigue en la torre y desaparece al héroe (public void RemoveCharacter()) o si desaparece el piso y el enemigo, pero no el héroe, el asciende (public void AddCharacter()).



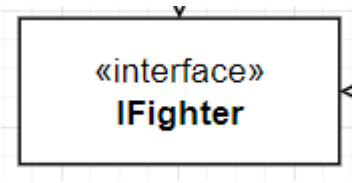
- Torre: El número de pisos cambiará conforme el jugador juegue (Valga la redundancia). Por ende, su atributo será cuántos pisos tiene la torre en un principio (private List<Floor> floors), y su operación será removerlos secuencialmente (public void RemoveFloor()).



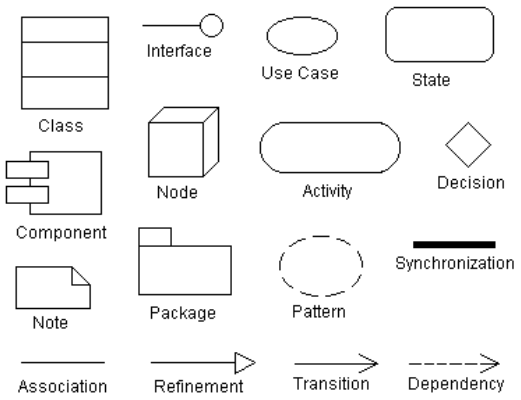
-

Interfaces:

IFighter: Define un contrato hacia las clases que implementan dicho contrato, por lo que estos deben implementar los atributos o métodos que allí se encuentren (public void Fight()).



Relaciones: *Guia:*



- Dependencias: Un objeto usa a otro (En una sola dirección)
- El enemigo y el jugador dependen de la interface
- Asociaciones: Objetos de una clase están conectados a objetos de otra (Bidireccional)
- (Tipo: agregación) El personaje y el piso
- (Tipo: composición) El personaje y la torre
- Organizaciones: Objetos diferentes tienen responsabilidades o propiedades en común
- Generalización: Relación entre una clase más general y una más específica o especializada
- El enemigo y el aliado a obstáculos
- Obstáculos a personaje
- Jugador a personaje
