

Final Reality

Proyecto Semestral

Profesores

Ignacio Slater Matías Toro

Auxiliares

Daniel Ramírez Diego Acevedo Martín Rojas Vicente González

Ayudantes

Alina González	Arturo Kullmer	Carlos Ruz	Catalina López
Constanza Pizarro	Juan Molina	Martina Mora	Máximo Flores

Índice

1. Introducción al proyecto	3
2. Descripción del problema	3
2.1. Personajes	3
2.2. Enemigos	3
2.3. Armamento	3
2.4. Hechizos	4
2.5. Efectos	5
2.6. Sistema de combate	5
3. Modelo de la solución	6
4. Evaluación	6
4.1. Puntaje	6
4.2. Modalidad de entrega	7
4.3. Bonificaciones	7
5. Recomendaciones	8

1. Introducción al proyecto

Uno de los principales objetivos del curso es que aprendan a escribir programas de calidad y utilizando buenas metodologías de diseño y programación. Para lograr este objetivo, se les presenta un proyecto semestral dividido en 3 Tareas en las que se espera que aplique los contenidos enseñados en el curso.

El proyecto simula un «caso real» en el que usted deberá desarrollar código de acuerdo a requisitos, como si los hubiera hecho un cliente «real» que serán explicados en el presente documento. Para promover el desarrollo constante de la solución, junto con presentar los requisitos específicos a implementar en cada momento, existirán pequeños entregables, denominados Entregas Parciales.

El proyecto a realizar será crear una versión simplificada del combate de la saga de videojuegos *Final Fantasy* desarrollado por *Square Enix*. A grandes rasgos, el juego consta de un grupo de personajes que puede controlar un usuario, y un grupo de enemigos que son controlados por la computadora. Estos personajes y enemigos poseen cualidades y características, e interactúan con diferentes elementos.

2. Descripción del problema

2.1. Personajes

El jugador tendrá a su disposición una cantidad fija de tres posibles personajes que controlará. Cada personaje tiene un nombre, puntos de vida, defensa y peso. El peso de un personaje determinará la rapidez de este para poder acceder a su turno, es decir, un personaje más pesado tardará más en poder realizar una acción. Llamaremos Party al grupo de personajes que controla el jugador.

Todos los personajes tienen una clase, la cual les otorgarán distintas capacidades, habilidades y atributos extra. Se pueden separar en dos categorías: Personajes comunes y mágicos. Los personajes comunes son aquellos que no pueden utilizar magia, en particular se componen de:

- Paladines
- Guerreros
- Ninjas

Por otro lado, los personajes mágicos son aquellos que poseen maná, un atributo con el que pueden efectuar hechizos. Los tipos son:

- Magos Negros
- Magos Blancos

Adicionalmente, cada personaje tiene la capacidad de equiparse un arma.

2.2. Enemigos

En los combates, además de los personajes que controlará el jugador habrá uno o más enemigos. Cada enemigo tendrá un nombre, puntos de vida, defensa, ataque y peso. Los enemigos **no pueden** equiparse armas ni usar hechizos. A diferencia de los personajes, la cantidad de enemigos no es fija, pero tiene una cantidad máxima de cinco que pueden participar en el combate.

2.3. Armamento

Las armas son objetos que los personajes pueden equiparse para atacar. Tienen un nombre, puntos de ataque, peso y un dueño asociado. Un personaje sin arma no puede realizar un ataque. Tampoco puede

equiparse un arma que ya tenga un dueño. Existen dos categorías: Armas comunes y mágicas. Su única diferencia reside en que las armas mágicas tienen adicionalmente puntos de ataque mágico, que serán utilizados para calcular el daño de un hechizo.

Las armas comunes son:

- Espadas
- Hachas
- Arcos

Las armas mágicas son:

- Varitas
- Bastones

Los tipos de armas que puede equiparse cada clase se muestran en el cuadro siguiente. Tenga en consideración que un personaje debe tener exactamente un arma equipada para atacar, pero el arma con la que ataca puede ser cambiada en su turno por cualquier otra que cumpla los criterios presentados en el Cuadro 1.

	Sword	Axe	Bow	Wand	Staff
Paladin	✓	✓	✗	✗	✗
Warrior	✓	✓	✓	✗	✗
Ninja	✓	✗	✓	✓	✗
Black Mage	✓	✗	✗	✓	✓
White Mage	✗	✗	✓	✓	✓

Cuadro 1: Armas equipables

2.4. Hechizos

Para atacar, además de sus armas, los magos pueden utilizar hechizos con distintos efectos. Existen dos tipos de magias; blanca, y negra, que pueden ser utilizadas únicamente por los magos de ese mismo color. Los hechizos existentes son:

[Magia oscura] **Trueno:** Reduce la vida del enemigo en `magicDamage` y tiene un 30% de probabilidad de paralizarle.

Costo: 20 puntos de maná.

[Magia oscura] **Fuego:** Reduce la vida del enemigo en `magicDamage` y tiene un 20% de probabilidad de quemarle.

Costo: 15 puntos de maná.

[Magia de luz] **Curación:** Cura a un aliado el 30% de sus puntos de vida máximos.

Costo: 15 puntos de maná.

[Magia de luz] **Veneno:** Envenena a un enemigo.

Costo: 30 puntos de maná.

[Magia de luz] **Parálisis:** Paraliza a un enemigo.

Costo: 25 puntos de maná.

Donde `magicDamage` son los puntos de ataque mágico del arma equipada al personaje. De forma similar a las armas, los magos pueden tener varios hechizos a su disposición para utilizar, en caso de decidir usar alguno debe seleccionarse exactamente uno.

2.5. Efectos

En el juego existirán tres tipos de efectos adversos, los cuales son causados por el resultado de algún hechizo utilizado por un personaje. Los efectos son:

- **Parálisis:** Cuando un enemigo está paralizado no puede realizar acción alguna, por lo que su turno es omitido. El efecto de la parálisis dura un turno.
- **Envenenado:** Los enemigos envenenados pierden $\frac{\text{magicDamage}}{3}$ puntos de vida al comienzo de cada uno de sus turnos, con una duración de cuatro turnos del enemigo que sufrió el efecto.
- **Quemado:** Los enemigos quemados pierden $\frac{\text{magicDamage}}{2}$ puntos de salud al comienzo de cada uno de sus turnos. El efecto dura tres turnos del enemigo que sufrió el efecto.

En este contexto, `magicDamage` es el daño mágico del arma que tenía equipado el personaje al momento de conjurar el hechizo al enemigo.

2.6. Sistema de combate

En el contexto del juego, definiremos como turnos los espacios en que el jugador esté utilizando a uno de sus personajes (seleccionando o realizando una acción), y los momentos en los que los enemigos «decidan» qué hacer.

En el turno de cada personaje, el jugador puede cambiar el arma de este tantas veces como quiera, pero para terminar el turno debe realizar una acción: atacar o utilizar un hechizo. El turno del personaje termina inmediatamente luego de que se realice la acción. El caso de los enemigos es análogo.

El orden de los turnos se maneja individualmente por unidad, esto quiere decir que cada personaje y enemigo tendrá una forma de definir cuándo le toca. Esto implica que el orden de las unidades no necesariamente será el mismo siempre.

Este orden dependerá del peso del personaje `weightchar` y el de su arma equipada `weightweapon`. Cada personaje (y enemigo) tendrá una barra de acción `actionBar`, la cual tendrá un máximo de puntos distinto para cada unidad y estará dado por:

$$\text{actionBar} = \text{weight}_{\text{char}} + 0.5 \cdot \text{weight}_{\text{weap}}$$

En el caso de los enemigos, simplemente será:

$$\text{actionBar} = \text{weight}_{\text{enemy}}$$

El sistema de combate consta del siguiente ciclo:

1. Al inicio de la partida, todas las unidades comienzan con su barra de acción en 0.
2. Se aumenta la barra de acción de todas las unidades **simultáneamente** en una cantidad k fija a definir.
 - Si después de aumentar la acción de todas las unidades, ninguna llega a llenar su barra de acción, el proceso se repite hasta que una la complete.

- Si después de aumentar la acción de todas las unidades, una o más completaron su barra de acción, se debe otorgar los turnos en orden de prioridad a aquellas que hayan logrado un excedente más grande.
3. La unidad que completó su barra de acción, consigue el turno. Al iniciar, se deben aplicar los efectos de estado que tenga, si los tuviese. Luego, debe realizar una acción para finalizar el turno:
 - Todas las unidades, al atacar, inflingirán `attackDamage - defensePoints`, donde `attackDamage` son los puntos de ataque de la unidad atacante, y `defensePoints`, los puntos de defensa de la unidad que recibe el ataque.
 - Si el ataque o hechizo en cuestión derrota a la unidad afectada, esta es eliminada del combate.
 - Si una unidad es derrotada al iniciar su turno debido a un efecto de hechizo, qué pena por ella. El turno finaliza inmediatamente sin acción alguna y es eliminada del combate.
 4. Se vacía la barra de acción de la unidad y esta vuelve a esperar por su turno.
 5. Si ningún personaje de la Party del jugador sigue vivo, o si todos los enemigos fueron derrotados, la partida finaliza. En el caso contrario, se retorna al paso 2.

3. Modelo de la solución

La resolución de este proyecto se hará siguiendo el patrón arquitectónico *Modelo-Vista-Controlador* donde primero se implementará el *Modelo* luego el *Controlador* y finalmente la *Vista*. Este patrón se irá explicando a lo largo del proyecto, y tendrán una guía constante para su correcta resolución. En el contexto del proyecto, estos componentes serán como se explica a continuación:

Modelo – Para la primera parte se le solicitará que cree todas las entidades necesarias que servirán de estructura base del proyecto y las interacciones posibles entre dichas entidades. Las entidades en este caso se refieren a los elementos que componen el juego.

Vista – Se le pedirá, de manera opcional, que cree una interfaz gráfica o de consola simple para el juego que pueda responder al input de un usuario y mostrar toda la información relevante del juego en pantalla.

Controlador – Servirá de conexión lógica entre la vista y el modelo, se espera que el controlador pueda ejecutar todas las operaciones que un jugador podría querer efectuar, que entregue los mensajes necesarios a cada objeto del modelo y que guarde información importante del estado del juego en cada momento.

4. Evaluación

4.1. Puntaje

- **Diseño del código fuente (3.0 pts.):** Se evaluará la calidad de su código, exigiendo que este cumpla con los principios de diseño enseñados en el curso.
- **Testing (1.2 pts.) y Coverage (0.8 pts.):** Se evaluará que su código tenga pruebas de su funcionamiento con una cobertura de al menos el 50% de las **líneas** de código. Para obtener el puntaje completo, el coverage debe ser de al menos 90%. Cabe destacar que las pruebas deben no solo comprobar que el fragmento a testear cumple con lo solicitado, sino que deberá también incluir casos de borde.
- **Documentación (1 pts.):** Cada clase, interfaz y método público debe estar documentado. Siga la guía de documentación que encontrará en el [FAQ](#) del curso.

Para guiar la realización de este proyecto, se plantearán objetivos pequeños en forma de *Entregas Parciales*.

- **Entrega Parcial (0.5 pts. c/u):** Dichas entregas deberán ser enviadas en la fecha señalada y no serán evaluadas directamente con nota. Solamente será evaluado que al momento de entregarla se cumpla con el objetivo planteado, **independientemente de su diseño**.

La no entrega de una entrega parcial supondrá un descuento en su nota final, en la cantidad de puntos señalada. Su finalidad es solamente incentivar el trabajo constante y continuo, por sobre realizar las tareas a último minuto, por lo que las entregas parciales **no tendrán feedback**.

4.2. Modalidad de entrega

Deberá subir todo su trabajo al repositorio privado proporcionado a través de *GitHub Classroom*.

Las entregas consistirán en un archivo de texto (.txt) que se subirá en la sección *Tareas* de *U-Cursos*. **Dicho archivo deberá tener siempre el mismo formato. El no cumplimiento de este implica un descuento en su nota final, específicamente severo si es un archivo comprimido (.rar, .zip)**. El formato es el siguiente:

Nombre: <Nombre completo> // Sin las llaves!
PR: <URL del pull request> // Sin las llaves!

La correcta URL de un pull request es de la forma:

<https://github.com/dcc-cc3002/final-reality-YourGithubName/pull/x>

Cuando considere que su entrega está lista para ser evaluada, realice un *pull request* a la rama *main* de su repositorio. Un *pull request* es básicamente proponer cambios y solicitar que alguien revise y aplique esos cambios en Github. En caso de que realizara un *push* de más commits a esa misma rama, luego de hecho el *pull request*, este se actualizará automáticamente con el commit más reciente. Sin embargo, para la revisión, se considerará siempre el último **que esté dentro del plazo de entrega**.

Si necesita seguir trabajando en su tarea después de haber creado el *pull request*, simplemente cree una nueva rama y continúe su trabajo desde ahí.

4.3. Bonificaciones

Puede ganar puntos adicionales al implementar lo siguiente:

- **TAREA 1, 2, 3 – Readme (0.3 pts.):** Obtenga puntos adicionales¹ detallando en su README.md aspectos que considere relevantes de aclarar en su proyecto. Algunos ejemplos pueden ser responder las siguientes preguntas:
 - *¿Por qué tomó ciertas decisiones en su diseño?*
 - *¿Cómo está organizado su código?*
 - *¿Qué patrones de diseño utilizó?*
- **TAREA 1, 2, 3 – Código y documentación en inglés (0.1/0.2 pts.):** Todo respetable código fuente y su documentación están en inglés. Nombres de clases, interfaces, métodos y variables debiesen estar en este idioma, así como también la documentación asociada.

- **TAREA 1, 2, 3 – Buen uso de Git (hasta 0.8 pts.):** Se premiará sustancialmente el uso apropiado del controlador de versiones, registrando pequeños cambios en cantidad, con descripciones adecuadas del registro, contrario a entregar toda una tarea en un solo registro, sin descripción:
 - **(0.1/0.2/0.3/0.4 pts.):** Crea múltiples commits, siendo cada uno un avance pequeño (al menos 10/15/20/25 por Tarea). Contar con este bonus es **necesario** para poder acceder a los siguientes.
 - **(0.2/0.4 pts.):** Utiliza descripciones de commits apropiados, indicando claramente lo que se realizó en el avance. Bonus completo si además dichas descripciones son basadas en [Conventional Commits](#).
- **TAREA 2, 3 – Manejo de excepciones con Require (0.4 pts.):** Se valorará la lectura y utilización de código extra si desarrolla sus excepciones utilizando correctamente la librería Require proporcionada.

5. Recomendaciones

- No se aferre al primer diseño; busque oportunidades de mejora y extensibilidad. Pregúntese, ¿Qué pasaría si en el futuro se quisiera implementar X funcionalidad nueva? ¿Mi código tendría que ser modificado si quisiera hacerlo? De esta manera, la mayor parte de su código soportaría extensibilidad para cualquier funcionalidad nueva que se deseé agregar.
- **NO DEJE LA TAREA PARA ÚLTIMO MOMENTO.** Aunque es un consejo común, y sabemos que es algo que se dice en todos los cursos, aquí es particularmente MUY IMPORTANTE. Trabajar apresuradamente no solo le resultará en un diseño descuidado, sino que no tendrá tiempo de pensar en uno mejor, y finalmente no podrá aplicar las buenas prácticas enseñadas en el curso.
- **Si algo no se especifica en el enunciado, tiene libertad creativa.** Esto es, manejar ciertos casos de borde o ciertos aspectos funcionales que no se detallan, pero asegúrese de cumplir con lo solicitado.

¹Hay bastantes más «Puntos adicionales» de los que usted cree que hay aquí. Lo primero que hace su ayudante al revisar su tarea, es leer su readme. Explicarle adecuadamente su trabajo le ayudará a entenderlo de mejor manera, haciéndole más feliz, y muy posiblemente evitándole a usted tediosos reclamos sobre algo que no se entendió.