Proyecto POO en C++ – Natal Combat

Descripción General

Natal Combat es un juego de rol (RPG) por turnos que se ejecuta íntegramente en consola. El proyecto busca que se apliquen los principios de Programación Orientada a Objetos (POO) aprendidos durante el curso: diseño UML, encapsulación, herencia, polimorfismo, librería STL, contenedores, manejo de excepciones y buenas prácticas de código.

Los jugadores formarán un grupo de **tres héroes** y atravesarán una mazmorra de diez salas, enfrentando enemigos, obteniendo equipamiento y acumulando puntuación. Cada partida se espera que dure aproximadamente 15 min.

Mecánicas del Juego

En **Natal Combat** tu grupo de tres héroes avanza por una mazmorra compuesta de diez salas consecutivas. Cada héroe se describe mediante cinco atributos principales:

- **Salud (HP):** determina los puntos de vida.
- **Ataque (ATK):** influye en el daño infligido.
- **Defensa (DEF):** reduce el daño recibido.
- **Velocidad (SPD):** decide el orden de los turnos.
- **Suerte (LCK):** interviene en la probabilidad de fallar o realizar golpes críticos.

Debes tener creado para el juego un pull de 6 personajes (héroes), de ellos tu escogerás 3 como tus héroes para atravesar las salas de la Mazmorra (pueden ser escogidos o generados al azar). Adicional a ello deberás haber credo un grupo de 15 enemigos, los enemigos pueden ser Soldado, Mini Jefe o Gran Jefe. Los Mini jefe siempre estarán en la sala 8, mientras que los de tipo Gran Jefe siempre estarán en la sala final (la 10).

Los combates son **por turnos** (uno de tus 3 héroes golpea, el enemigo responde, y vuelve a golpear el Héroe, así sucesivamente hasta que caiga el enemigo o 2 de tus 3 héroes). En cada ronda, los personajes actúan de mayor a menor velocidad. Cuando un héroe ataca, el juego calcula primero si el golpe falla (para ello se basa en el atributo LCK, a mayor LCK, mayor probabilidad de no fallar). En caso de impactar, el daño infringido disminuye inmediatamente la cantidad de HP del oponente. Ten en cuenta que el contrincante también tiene la función de LCK así que al tener mucha suerte, puede suceder que tu golpe sea fácilmente esquivado (LA FÓRMULA DE GOLPE INFRINGIDO, SUERTE Y AFECTACIÓN LA DEBES AJUSTAR TU, PARA LOGRAR UNA BUENA JUGABILIDAD)

A lo largo de la mazmorra encontrarás **cuatro momentos con ítems para tus héroes**:

1. **Mercado inicial (sala 1):** puedes adquirir un arma y una armadura comunes para empezar con ventaja, debes elegir a cual héroe equipar con ellas.
2. **Cofre (sala 3):** concede un accesorio que aumenta la Suerte en +2, o 3 pociones de curación (debes elegir a cuales héroes se los darás)
3. **Tesoro (sala 6):** al derrotar al Soldado obtienes tres armas raras que añade +8 ATK y un 5 % en LCK (un arma para cada Héroe).
4. **Santo Grial (sala 8):** Todos tus héroes recuperan el 100% de su salud y están listos para enfrentar los últimas 2 salas.

Sin importar cómo haya sido la batalla, al ganar una de las salas, cada Héroe debe aumentar un 2% su capacidad de ataque y de defensa.

El **inventario** posee (50) items como **armas, armaduras y pociones** de diferentes tipos, sé creativo con esta parte, ya que el inventario es lo que le brindará jugabilidad a los elementos con los que se equipan a los héroes cada que vez que se juega una partida. Cada uno de los 3 posibles elementos del inventario pueden **influir en máximo 2 de los cinco atributos** de un personaje.

Ya sea que ganes o pierdas en la mazmorra, el juego registra únicamente tu **puntuación** en un archivo que se llama “scores”. El registro guardado debe tener el alias del jugador, fecha-hora, máxima sala superada, cantidad salud perdida acumulada (sumando los 3 héroes). Al iniciar una nueva sesión se muestran los cinco mejores resultados para fomentar la re-jugabilidad. Se considera mejor el que haya llegado más lejos en las salas y si en ello hay empate, es mejor quien haya perdido menos salud en el juego.

En síntesis: crea héroes equilibrando sus estadísticas, administra un inventario limitado, supera encuentros cada vez más exigentes y compite por la mejor puntuación. Todo ello implementado aplicando herencia, polimorfismo, manejo de excepciones y contenedores STL.

Ejemplo de un Balance inicial de un Héroe

- HP 60
- ATK 6
- DEF 3
- SPD 4
- LCK 1

Requerimientos Técnicos

- **C++** como lenguaje de programación.
- Interfaz exclusivamente en consola, gráfica es opcional.
- Uso de contenedores, librería STL y conceptos vistos de POO.
- **El Uso de IA debe ser justificado y haber generado comprensión. El código en el proyecto que no se comprenda puede ser considerado plagio.**
- Uso de archivos para lograr persistencia del score.
- Proyecto enlazado con **CMake**.

Mínimos Esperados (Condición de Calificación)

1. Uso **continuo** de **git** con mensajes descriptivos (en el Github Classroom). Tiene una reducción de 5 décimas de la nota definitiva si sólo tiene un único push como entrega
2. Menús con las opciones de jugabilidad al comenzar e ir avanzando en el juego.
3. Estándar de nombramiento lowerCamelCase y código documentado.
4. Diseño UML con el diagrama de clases de la solución.
5. Coherencia entre el diseño y el código

Entregables:

Diagrama UML con el diseño de la solución: **8 de mayo**

Avance de creación de clases y bosquejo de la jugabilidad: **15 de mayo**

Entrega final y sustentación: **28 de mayo**

Calificación

- **Autoevaluación**: 5%
- **Calificación del compañero**: 5%. En trabajos individuales la autoevaluación tendrá un 5% de peso. Equipos de máximo tres personas.
- **Calificación del profe**: compuesta por entregables, diseño, funcionalidad, estilo de codificación y mejores prácticas: 90% 0 95% si trabajaste individualmente. La rúbrica de evaluación explica los elementos que se consideran para la calificación del proyecto.
- **Propiedad intelectual**: valor entre 0 y 1 que multiplica la calificación total. Se demuestra durante la sustentación.
- **Nota final** = criterios evaluación * propiedad Intelectual

Rúbrica de evaluación

La nota del profe será dada al finalizar la sustentación según los criterios que se describen a continuación y su capacidad para sustentar su trabajo.

	5	4	3	2	1	0
Entregables (10%)	Los entregables contienen toda la información solicitada y tiene alta calidad en cuanto a estilo y formato.	Se entregaron todos los artefactos. En términos de contenido están completos pero podría mejorar en cuanto a estilo o formato	Se entregaron todos los artefactos. En términos de contenido están completos pero podría mejorar en cuanto a estilo Y formato	Faltan algunos de los artefactos pero los entregados tiene buen estilo y formato	Faltan algunos de los artefactos y los entregados necesitan mejoras de estilo y formato	No se entregaron los artefactos
Diseño (30%)	El diseño responde a los requisitos. Se detectaron todas las clases importantes y para cada clase se detectaron los atributos y métodos importantes. Usa las relaciones correctas	El diseño responde a los requisitos. Se detectaron todas las clases importantes y para cada clase se detectaron los atributos y métodos importantes. Tiene algunas relaciones incorrectas o faltantes en el diseño de los métodos como por ejemplo los atributos.	El diseño responde a los requisitos. Faltó detectar algunas clases importantes. Faltó detectar algunos de los atributos y métodos importantes. Tiene algunas relaciones incorrectas. El diseño no corresponde a lo codificado.	Faltó detectar la mayoría de clases importantes Faltó detectar muchos de los atributos y métodos importantes. Tiene muchas relaciones incorrectas	El diseño no satisface los requisitos	No se entregó el diseño
Funcionalidad (30%)	Cumplió con todos los requisitos.	Fueron desarrollados mínimo el 75% de los requisitos	El diseño responde al 75% de los requisitos / podría mejorarse	Fueron desarrollados mínimo el 25% de los requisitos	Fueron desarrollados menos del 25% de los requisitos	La funcionalidad no responde a lo solicitado
Estilo de codificación (5%)	El código se encuentra correctamente indentado, los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. El código tiene documentación interna para facilitar la revisión.	La mayoría del código se encuentra correctamente indentado- La mayoría de los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. La mayoría del código tiene documentación interna para facilitar la revisión	Falta alguno de los ítems de calidad del estilo de codificación o alguna se cumple con mala calidad	Faltan dos de los ítems de calidad del estilo de codificación o dos se cumple con mala calidad	No hay código fuente suficiente para evaluar el estilo de codificación.	No cumple con el estándar de nombramiento No se encuentra correctamente indentado No está divido adecuadamente
Mejores prácticas (5%)	El código muestra mejores prácticas de desarrollo siempre. Reúso,	El código muestra mejores prácticas de desarrollo en la mayoría de los	El código muestra buenas prácticas de desarrollo pero falta mejorar	El código aplica pocas buenas prácticas de desarrollo. Falta	No hay código fuente suficiente para evaluar las buenas prácticas.	No se entregó proyecto o el proyecto no cubre al menos el 25% de

	separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.	casos, pero falta mejorar algunos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones	dos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones	mejorar tres de los siguientes aspectos: Reúso, separación de operaciones, buen manejo de ciclos, simplicidad, programación defensiva validaciones.		las funcionalidades. No es posible evaluarlo .

Rúbrica de propiedad intelectual

	1	0.8	0.6	0.4	0.2	0
Sustentación	Es evidente que el estudiante entiende el código que desarrolló lo explica con claridad y responde correctamente a las preguntas.	La sustentación es buena pero se evidenció inseguridad del estudiante para explicar algunas partes del trabajo desarrollado o para responder algunas preguntas.	La sustentación es aceptable se evidencia que el estudiante desarrolló el código pero le cuesta trabajo explicar aspectos del código.	La sustentación es regular se evidenció inseguridad del estudiante para explicar gran parte del trabajo desarrollado o para responder muchas de las preguntas. Parece que el código no hubiera sido desarrollado por el estudiante.	El estudiante demuestra que entiende partes del código, pero no tiene claro cómo se relacionan con la funcionalidad solicitada.	Se evidencia que el estudiante no entiende el código desarrollado, no es capaz de responder a las preguntas formuladas de manera correcta.