Presentación del proyecto final IPOO

Grupo 6

Integrantes

Camilo Andrés Alarcón Villareal 202519844-3744

Santiago Arce Núñez 202516846-3744

Francisco Popo - 202515537-3744

Alejandro Martínez Ruiz 202510009-3744

Maestro

José Antonio García Mecinas

Se desarrollará un Sistema de Daños en Batalla por turnos.

En cada combate:

- **Una entidad controlada por el jugador se enfrenta a una entidad controlada por la computadora.**
- **La batalla avanza en rondas de turnos; cada turno se elige una acción (atacar, ataque especial, cambiar arma, etc.) y se resuelve con probabilidad y reglas de daño.**
- **No hay interfaz gráfica: toda la experiencia es textual. Los mensajes deben ser comprensibles, contextuales y, cuando proceda, polimórficos, reflejando el tipo concreto de arma, armadura, extremidad afectada, etc.**

¿En qué consiste el sistema?

Cada **Entidad** (jugador y oponente) representa un personaje con vida, resistencia (“stamina”), atributos físicos y un cuerpo dividido en partes (cabeza, torso, brazos, piernas). Cada parte puede recibir daño, portar armas (brazos) o equipar armaduras (todas, según configuración). Los turnos se juegan por rondas: ambos personajes deciden su acción (atacar o defender), el sistema resuelve y aplica simultáneamente ambos resultados. El combate sigue hasta que uno de los personajes pierde toda su vida.

Requerimientos

R#	Descripción
R1	El sistema debe permitir que 2 entidades (jugador y oponente) participen en un combate por turnos.
R2	Cada entidad debe tener atributos como vida, constitución, resistencia, armadura y capacidad de movimiento.
R3	El jugador debe poder elegir acciones desde un menú textual (atacar, ataque especial, cambiar arma, pasar turno).
R4	El oponente debe tomar decisiones automáticas según reglas básicas de inteligencia artificial.
R5	El sistema debe calcular la iniciativa de cada entidad combinando su capacidad de movimiento y velocidad de ataque.
R6	Al atacar el sistema debe determinar probabilidad de acierto, calcular daño bruto y aplicar mitigación por armaduras.
R7	Las armas y armaduras deben degradarse (disminuir durabilidad) con el uso o al recibir daño.
R8	El sistema debe mostrar mensajes claros y polimórficos describiendo las acciones y resultados de combate.
R9	El combate debe finalizar cuando una de las entidades tenga vida igual o menor a cero.
R10	El sistema debe permitir equipar y cambiar armas o armaduras en las partes del cuerpo correspondientes
R11	Debe existir una clase base abstracta para entidad, arma y armadura con métodos virtuales puros donde corresponda.
R12	El sistema debe implementar la regla de los tres (destructor, constructor de copia y operador =) para las clases que gestionan recursos dinámicos.

R13	El sistema debe estar desarrollado en C++03 y ejecutarse en consola de texto.
R14	El código debe compilar sin advertencias con las banderas: -std=c++03 -wall -Wextra -pedantic -Wshadow -Wconversion.
R15	La salida textual debe ser clara, legible y contextual, incluyendo información de cada ronda y acción.
R16	El sistema debe documentarse con Doxygen, incluyendo pre/postcondiciones e invariantes.
R17	Debe existir un archivo README explicando compilación, ejecución y ejemplos de salida.
R18	El sistema debe seguir una estructura modular include/, src/, demo/, bin/, y un Makefile explícito.

Tabla de normalización del vocabulario del problema

Término normalizado	Tipo	Definición	Sinónimos	Clase/atributo/método
clasificación	clasificación	Baja, Media, Alta. Clasifica atributos de Entidad como fuerza o movimiento.		Clase
entidad	Abstracta	Representa un combatiente: gestiona vida, stamina, fuerza, movimiento y acciones del turno.	personaje	Clase
jugador	entidad	Entidad controlada por el usuario.		Clase
oponente	entidad	Entidad controlada por IA.	enemigo	Clase
Vida	entero	Puntos de vida totales.	HP	Atributo
Fuerza	clasificación	Aumenta el daño final al atacar.		Atributo
Resistencia	entero	Energía disponible (stamina). Afecta acierto y capacidad de actuar.	stamina	Atributo
CapacidadMovimiento	clasificación	Influye en la evasión y la probabilidad de acierto.	movilidad	Atributo
Armadura	armadura	Referencia a armadura general si el sistema es simplificado.		Atributo
elegirAccion		Devuelve la acción del turno (atacar, defender, cambiar arma). Polimórfico.	decidirAccion	Método
recibirDaño(parte, cantidad)		Recibe daño dirigido y lo delega al Cuerpo.		Método
equiparArma		Equipa un arma en una parte capaz de portarla.	setArma	Método
equiparArmadura		Equipa armadura en una parte del cuerpo.	setArmadura	Método
mostrarEstado		Muestra vida, stamina y equipamiento.		Método

cuerpo	Compuesta	Conjunto de partes del cuerpo pertenecientes a una Entidad.	anatomía	Clase
Partes	Vector<parteCuerpo>	Lista de todas las partes del cuerpo.		Atributo
mostrarEstado		Muestra estado de cada ParteCuerpo.		Método
recibirDaño		Envía daño a la ParteCuerpo correspondiente.		Método
parteCuerpo	cuerpo, Abstracta	Representa una sección anatómica que puede equipar elementos y recibir daño.		Clase
cabeza	parteCuerpo	Parte del cuerpo susceptible a daño.		Clase
torso	parteCuerpo	Parte del cuerpo susceptible a daño.		Clase
piernas	parteCuerpo	Parte del cuerpo susceptible a daño.		Clase
brazos	parteCuerpo	Parte del cuerpo susceptible a daño.		Clase
porcentajeImpacto	entero	Probabilidad relativa de ser impactada.	probImpacto	Atributo
llevaArma	bool	Indica si esta parte puede portar un arma.		Atributo
arma	referencia arma	Arma equipada en esta parte.		Atributo
armadura	referencia armadura	Armadura equipada en esta parte.		Atributo
recibirDaño		Procesa absorción por armadura, actualiza daño local y devuelve daño neto.		Método
equiparArma		Equipa un arma en esta parte.		Método
equiparArmadura		Equipa armadura en esta parte.		Método
arma	Abstracta	Define daño base, precisión, peso y durabilidad. No ejecuta ataques ni elige objetivo.		Clase
espada	arma	Variante concreta de arma.		Clase
hacha	arma	Variante concreta de arma.		Clase
lanza	arma	Variante concreta de arma.		Clase
arco	arma	Variante concreta de arma.		Clase
Durabilidad	entero	Nivel de desgaste del arma.		Atributo
DañoAtaque	entero	Daño base del arma.		Atributo
VelAtaque	entero	Velocidad del arma.		Atributo
Peso	entero	Penalización por peso.		Atributo
Precisión	entero	Precisión base del arma.		Atributo
calcularDañoBase		Devuelve el daño base del arma.		Método
getPrecision		Devuelve la precisión del arma.		Método
describir		Devuelve descripción polimórfica.		Método
armadura	Abstracta	Reduce daño recibido en la parte y se degrada con uso.	protección	Clase
tipo	clasificación	tipo de armadura		Atributo
Durabilidad	entero	Durabilidad restante de la armadura.		Atributo

DañoAbs	entero	Cantidad de daño que absorbe.		Atributo
ReducMovimiento	entero	Penalización a movimientos según tipo.		Atributo
absorber(daño)		Absorbe parte del daño que llega a la parte.		Método
degradar()		Reduce durabilidad.		Método
describir()		Devuelve descripción polimórfica.		Método
sistema	Orquestador	Sistema de Batalla: calcula acierto, objetivo, daño final y aplica resultados.	SistemaDeBatalla	Clase
turno	Clase	Registra las acciones declaradas por jugador y oponente en una ronda, así como los resultados generados por el Sistema de Batalla. No resuelve lógica de combate.		Clase
accionJugador	texto	Acción que el jugador declara para este turno (atacar, defender, cambiar arma, etc.).		Atributo
accionOponente	texto	Acción que el oponente declara para este turno.		Atributo
resultadoJugador	texto	Resultado del turno para el jugador (ej.: “acierta al torso por 8 daño”).		Atributo
resultadoOponente	texto	Resultado del turno para el oponente (ej.: “falló el ataque”).		Atributo
registrarAccion		Registra una acción declarada por jugador u oponente en el turno. No resuelve la acción, solo la almacena.		Método
mostrarResultados		Muestra o devuelve el resumen de acciones y resultados del turno.		Método

Clases, atributos y métodos principales

Entidad (jugador/oponente)

Atributos

- vida
- fuerza
- resistencia (stamina)
- capacidadMovimiento
- armadura (armadura global opcional, según diseño del sistema)
- cuerpo (referencia a Cuerpo)

Métodos

- Getters y setters para cada atributo.
- **elegirAccion()** – Devuelve la acción que la entidad realizará este turno.
- **recibirDaño(parte, cantidad)** – Recibe daño dirigido a una parte y lo delega a su Cuerpo.
- **equiparArma(parte, arma)**
- **equiparArmadura(parte, armadura)**
- **mostrarEstado()** – Muestra vida y equipamiento actual.
- **estaViva()**
- **reducirResistencia(cantidad)** / **getResistenciaActual()**

Cuerpo

Atributos

- partes (lista de ParteCuerpo)

Métodos

- **obtenerParte(nombre)** – Devuelve la referencia a una parte del cuerpo.
- **recibirDañoEnParte(parte, cantidad)**
- **mostrarEstado()**
- **partesVivas()**

ParteCuerpo (cabeza, torso, brazos, piernas)

Atributos

- nombre
- dañoLocal (multiplicador según la parte del cuerpo)
- integridad
- llevaArma
- arma (referencia si porta una)
- armadura (referencia)

Métodos

- **equiparArma(arma)**
- **equiparArmadura(armadura)**
- **recibirDaño(cantidad)**
- **mostrarEstado()**

Arma (abstracta) y derivados (espada, hacha, lanza, arco, etc.)

Atributos

- durabilidad
- dañoAtaque
- velAtaque
- peso
- precision

Métodos

- Getters y setters.
- **describir()** – Mensaje contextual polimórfico.

Armadura (abstracta) y derivados

Atributos

- durabilidad
- dañoAbs
- reduccMovimiento

Métodos

- **absorber(daño)** – Devuelve cuánto daño absorbe.
- **degradar()**
- **describir()**

SistemaDeBatalla

Atributos

- jugador
- oponente
- ronda actual

Métodos

- **iniciarCombate()**
- **resolverTurno()**
- **resolverAtaque(atacante, defensor)**
- **seleccionarObjetivo(entidad)** – Determina la parte del cuerpo objetivo.

- **calcularProbabilidadDeAcierto(atacante, arma)**
- **aplicarDaño(defensor, parte, cantidad)**
- **mostrarResumenRonda(turno)**

Turno

Atributos

- acciónJugador
- acciónOponente
- resultadoJugador
- resultadoOponente

Métodos

- **registrarAccion(entidad, acción)**
- **registrarResultado(entidad, resultado)**
- **mostrarResultados()**

Responsabilidades

- **SistemaDeBatalla** coordina el combate: pide a cada Entidad su acción en la ronda, calcula aciertos, selecciona la parte objetivo, determina el daño y aplica los resultados de forma simultánea.
- **Entidad** decide su acción del turno (atacar, defender, etc.), administra su stamina, maneja sus armas y armaduras, y actualiza su vida cuando recibe daño.
- **ParteCuerpo** recibe el daño, consulta su Armadura (si tiene), calcula cuánto daño absorbe, actualiza su daño local y devuelve al Cuerpo el daño neto que atraviesa la protección.
- **Cuerpo** recibe el daño neto desde la ParteCuerpo y lo envía a la Entidad para que reduzca su vida total.
- **Arma** aporta valores como daño base, precisión, peso y durabilidad. El SistemaDeBatalla usa esos valores en sus cálculos.
- **Armadura** absorbe parte del daño cuando una ParteCuerpo es impactada, y se degrada según el impacto recibido.
- **Turno** funciona como registro: guarda las acciones elegidas por jugador y oponente y permite mostrar el resumen de la ronda, pero no resuelve la lógica del combate

CRC

Entidad

Responsabilidades

- Mantener atributos globales (vida, fuerza, resistencia, capacidadMovimiento, armadura, cuerpo).
- Elegir la acción del turno.
- Delegar daño a su Cuerpo.
- Equipar armas y armaduras en las partes del cuerpo.
- Mostrar estado y saber si sigue viva.

Colaboradores

Cuerpo, ParteCuerpo, Arma, Armadura, SistemaDeBatalla

Mensajes

- **SistemaDeBatalla → Entidad:** elegirAccion()
- **SistemaDeBatalla → Entidad:** recibirDaño(parte, cantidad)
- **Entidad → Cuerpo:** recibirDañoEnParte(parte, cantidad)
- **Entidad → ParteCuerpo:** equiparArma(arma)
- **Entidad → ParteCuerpo:** equiparArmadura(armadura)
- **SistemaDeBatalla → Entidad:** estaViva()
- **SistemaDeBatalla → Entidad:** mostrarEstado()

Cuerpo

Responsabilidades

- Contener las partes del cuerpo.
- Redirigir daño a la ParteCuerpo correspondiente.
- Entregar daño neto procesado a la Entidad.
- Mostrar estado físico del personaje.

Colaboradores

ParteCuerpo, Entidad

Mensajes

- **Entidad → Cuerpo:** recibirDañoEnParte(parte, cantidad)
- **Cuerpo → ParteCuerpo:** recibirDaño(cantidad)
- **ParteCuerpo → Cuerpo:** *devuelve daño neto*
- **SistemaDeBatalla / Entidad → Cuerpo:** mostrarEstado()
- **Entidad → Cuerpo:** obtenerParte(nombre)

ParteCuerpo

Responsabilidades

- Mantener integridad, dañoLocal, nombre.
- Equipar arma o armadura.
- Procesar daño (aplicar armadura, disminuir integridad, devolver daño restante).
- Mostrar su estado.

Colaboradores

Arma, Armadura, Cuerpo

Mensajes

- **Cuerpo → ParteCuerpo:** recibirDaño(cantidad)
- **Entidad → ParteCuerpo:** equiparArma(arma)
- **Entidad → ParteCuerpo:** equiparArmadura(armadura)
- **SistemaDeBatalla / Cuerpo → ParteCuerpo:** mostrarEstado()
- **ParteCuerpo → Armadura:** absorber(daño)
- **ParteCuerpo → Armadura:** degradar()

Arma

Responsabilidades

- Proveer atributos del arma necesarios para cálculos de combate.
- Describir el arma.

Colaboradores

ParteCuerpo, Entidad, SistemaDeBatalla

Mensajes

- **Entidad → Arma:** se consulta vía getters (daño, precisión, peso...)
- **SistemaDeBatalla → Arma:** getters para cálculos de probabilidad y daño
- **SistemaDeBatalla → Arma:** describir()

Armadura

Responsabilidades

- Absorber daño y degradarse.
- Proveer reducciones de movimiento y defensa.
- Describir la armadura.

Colaboradores

ParteCuerpo

Mensajes

- **ParteCuerpo → Armadura:** absorber(daño)
- **ParteCuerpo → Armadura:** degradar()
- **SistemaDeBatalla → Armadura:** describir()

SistemaDeBatalla

Responsabilidades

- Iniciar y controlar el combate.
- Solicitar acciones a ambos combatientes.
- Resolver ataques: calcular aciertos, seleccionar objetivo, aplicar daño.
- Registrar resultados en cada turno.
- Determinar el fin del combate.

Colaboradores

Entidad, Turno, Arma, ParteCuerpo, Cuerpo

Mensajes

- **SistemaDeBatalla → Entidad:** elegirAccion()
- **SistemaDeBatalla → Turno:** registrarAccion(entidad, acción)
- **SistemaDeBatalla → SistemaDeBatalla:** resolverAtaque(atacante, defensor)
- **SistemaDeBatalla → SistemaDeBatalla:** seleccionarObjetivo(entidad)
- **SistemaDeBatalla → SistemaDeBatalla:** calcularProbabilidadDeAcierto(entidad, arma)
- **SistemaDeBatalla → Entidad:** recibirDaño(parte, cantidad)
- **SistemaDeBatalla → Turno:** registrarResultado(entidad, resultado)
- **SistemaDeBatalla → Turno:** mostrarResultados()
- **SistemaDeBatalla → Jugador/Oponente:** estaViva()

Turno

Responsabilidades

- Registrar acciones declaradas por jugador y oponente.
- Registrar los resultados de la ronda.
- Mostrar resumen del turno.

Colaboradores

SistemaDeBatalla, Entidad

Mensajes

- **SistemaDeBatalla → Turno:** registrarAccion(entidad, acción)
- **SistemaDeBatalla → Turno:** registrarResultado(entidad, texto)
- **UI / SistemaDeBatalla → Turno:** mostrarResultados()

EPS: SistemaDeBatalla.resolverTurno()

`void SistemaDeBatalla::resolverTurno(Turno& turno)`

Toma las acciones del turno, calcula aciertos y daño, aplica efectos simultáneos y deja registrados los resultados

PRECONDICIONES

- **jugador y oponente** no son null.
- El Turno entregado **ya tiene asignadas**:
  - turno.accionJugador
  - turno.accionOponente
- Ambas entidades tienen:
  - un **Cuerpo** con sus partes creadas.
  - su arma equipada.
- El combate sigue activo:
  - jugador->estaVivo() == true
  - oponente->estaVivo() == true

ENTRADAS

- Referencia a un objeto Turno

PROCESO

1. **Interpretar las acciones**
   a. Se leen accionJugador y accionOponente.
   b. Se determinan los efectos: ¿ambos atacan?, ¿uno defiende?, etc.
2. **Si una entidad ataca:**
   a. Llamar a calcularProbabilidadDeAcierto(entidad, arma)
   b. Hace cálculos para decidir si acierta.
   c. Si acierta:
      i. Llamar a seleccionarObjetivo(entidad) → devuelve una ParteCuerpo.
      ii. Calcular daño base usando:
         1. daño del arma
         2. fuerza
         3. resistencia (stamina)
         4. dañoLocal de la parte objetivo
      iii. Llamar a aplicarDaño(defensor, parte, dañoCalculado)
3. **Aplicar resultados de manera simultánea**
   a. Ataque del jugador y del oponente se resuelven, los efectos se aplican luego del cálculo de ambos
4. **Registrar los resultados en Turno**
   a. turno.registrarResultado(jugador, textoResultado)
   b. turno.registrarResultado(oponente, textoResultado)
5. **Actualizar ronda**
   a. ronda++

SALIDAS

- El Turno queda con:
  - resultadoJugador
  - resultadoOponente
- La vida y la stamina de las entidades pueden haber cambiado.
- Algunas armaduras pueden haber perdido durabilidad.
- Las partes del cuerpo pueden haber reducido integridad.
- Se incrementa el número de ronda en el Sistema.

POSTCONDICIONES

- El Turno tiene resultados coherentes con las acciones tomadas.
- La vida de jugador u oponente queda actualizada según el daño recibido.
- Las armaduras afectadas están degradadas correctamente.
- Las partes del cuerpo impactadas actualizan su integridad.
- Las entidades pueden haber quedado sin vida (estaViva() == false).
- La ronda del sistema aumentó exactamente en 1.

Diagrama UML