**Desafío #3 Video Juego**

**Momento II**

**Keiner Marcelo Torres Villeros**

**Nikolas Geovanny Ortega Suarez**

**Informática 2**

**Augusto Enrique Salazar**

**Ingeniería en telecomunicaciones**

**Universidad de Antioquia**

**2025-1**

**1. Diagrama de Clases de la Capa Lógica**

**Objetivo**

Diseñar la estructura de clases que gestionará la lógica principal del juego, incluyendo:

* Entidades (jugador, enemigos, obstáculos).
* Gestión de niveles.
* Física y colisiones.

**Contenedores Utilizados**

* **std::vector<Obstáculo\*>**: Para manejar obstáculos en el Nivel 1.
* **std::vector<Hitbox\*>**: Para gestionar ataques en los Niveles 2 y 3.

**Código del Diagrama (PlantUML)**

CLASES BASE

clase Entidad {

#posX : float

#posY : float

#ancho : float

#alto : float

#velocidadX : float

#velocidadY : float

+ actualizar(deltaTiempo: float)

+ dibujar(painter: QPainter\*)

+ getBoundingRect() : QRectF

+ getPosicion() : QPointF

}

clase Obstáculo se extiende Entidad {

- miMovimiento : TipoMovimiento

- amplitud : float

- frecuencia : float

+ Obstáculo(tipo: TipoMovimiento)

}

PERSONAJES

clase Luchador se extiende Entidad {

#vida : float

#dañoBase : float

#estaAtacando : bool

#cooldownAtaque : float

+ recibirDaño(daño: float)

+ estaVivo() : bool

+ saltar()

+ actualizar(deltaTiempo : float)

- aplicarGravedad(deltaTiempo : float)

- morir()

- estaEnElSuelo() : bool

}

clase PersonajeJugador se extiende Luchador {

+ procesarInput(teclas: QSet<int>)

- atacar(tipo : TipoAtaque)

}

clase Enemigo se extiende Luchador {

- estadoIA : EstadoIA

-temporizadorEstado : float

+ actualizar(deltaTiempo: float)

- decidirAcción()

- decidirSiguienteEstado()

- estaEnRangoDeAtaque() : bool

- atacarAlJugador()

}

GESTIÓN DE NIVELES

clase Nivel1 {

- tiempoParaGanar : float

- aceleracionGlobal : float

- obstáculos : vector<Obstáculo\*>

- jugador : PersonajeJugador\*

+ Nivel1()

+ actualizar(deltaTiempo: float)

+ dibujar(painter: Qpainter\*)

+ nivelTerminado() : bool

- generarObstáculos()

- actualizarTimer(deltaTiempo : float)

- revisarColisiones()

- reiniciarNivel()

}

clase Nivel2\_3 {

- jugador : PersonajeJugador\*

- enemigo : Enemigo\*

- hitboxesActivas : vector <Hitbox\*>

+ Nivel2\_3(nivel: int)

+ actualizar(deltaTiempo: float)

+ dibujar(painter: QPainter\*)

+ getEstadoJuego() : EstadoJuego

- revisarColisionesDeCombate()

- limpiarHitboxesExpiradas()

- crearHitboxPara(luchador: Luchador\*, tipo: TipoAtaque)

' --- RELACIONES ---

Entidad <-- Obstáculo

Entidad <-- Luchador

Luchador <-- PersonajeJugador

Luchador <-- Enemigo

Nivel1 o-- PersonajeJugador

Nivel1 \*-- Obstáculo

Nivel2\_3 o-- PersonajeJugador

Nivel2\_3 o-- Enemigo

**2. Cambios y Mejoras Respecto al Momento I**

**Sistema de Combate**

* **Problema inicial**: Usar el BoundingBox del personaje para ataques era poco realista.
* **Solución**: Implementar Hitboxes temporales que aparecen al atacar y desaparecen tras un breve tiempo.

**IA del Enemigo**

**- Enfoque**: Máquina de Estados Finitos (FSM) con estados:

* + ESPERAR, ATACAR, DEFENDER.

**- Dificultad**: En Nivel 3, el enemigo reacciona más rápido.

**Nivel 1 – Modificaciones**

**- Objetivo**: Sobrevivir 60 segundos sin chocar.

**- Física**:

* + Obstáculos con movimiento rectilíneo **acelerado** (aumenta la dificultad).
  + Algunos obstáculos tienen movimiento **sinusoidal** para mayor variedad.

**Simplificación de Features**

* Se eliminó la barra de "Ki" por no aportar a la jugabilidad central.

**3. Sprites por Nivel**

| **Nivel** | **Sprites (Descripción)** |
| --- | --- |
| **1** |
| **2** |
| **3** |

**4. Descripción Detallada de Niveles**

**Nivel 1: Esquiva Obstáculos**

* **Vista**: Cenital (desde arriba).
* **Interacciones**:
  + **Jugador**: Se mueve horizontalmente con ←/→.
  + **Obstáculos**: Caen desde arriba con MRUA.
  + **Colisiones**: Si choca, el temporizador se reinicia.

**Dibujo Informal**:

Diagrama

El contenido generado por IA puede ser incorrecto.

CAMBIANDO LA PUNTUACION POR UN CONTRARELOJ

**Niveles 2 y 3: Combate**

* **Vista**: Lateral (como un juego de peleas).
* **Interacciones**:
  + **Ataques**: Crean Hitboxes temporales (ej: puño → daño leve, patada → daño alto).
  + **IA**: Enemigo alterna entre atacar y defenderse.

**Dibujo Informal**:

#EL MISMO DEL MOMENTO 1

**Físicas Implementadas**

* **Gravedad**: Afecta a saltos y caídas.
* **Knockback**: Empuja al personaje al recibir daño.

**Notas Finales**:

* El diseño prioriza funcionalidad sobre complejidad.
* Se usaron patrones como FSM para la IA y composición para las Hitboxes.
* Pendiente: Optimizar colisiones con spatial partitioning (opcional para futuras iteraciones)