PROYECTO NEOSNAKE

INTEGRANTES

MARIANA HENAO ECHEVERRI

MARIA FERNANDA PIEDRAHITA MONTOYA

CURSO

ALGORITMOS Y PROGRAMACIÓN ORIENTADA A OBJETOS

PROFESOR

JESÚS ANDRÉS HINCAPIÉ LONDOÑO

Descripción del Problema.

Este proyecto tiene como objetivo rediseñar y mejorar el clásico juego Snake, incorporando una jugabilidad más dinámica, nuevas mecánicas, y un entorno visual más intuitivo. Se busca ofrecer una experiencia interactiva que combine desafío, adaptabilidad y control preciso por parte de los jugadores.

La lógica del juego se organiza a partir de clases que representan los elementos esenciales del mundo: la serpiente, sus movimientos, su crecimiento al alimentarse, la aparición de obstáculos, la gestión del tiempo y la puntuación en pantalla. El sistema debe permitir una interacción fluida y coherente entre estos elementos, garantizando una partida funcional y equilibrada.

Además, se incorporan diferentes niveles de dificultad que modifican la velocidad, los recursos disponibles y las condiciones del entorno, lo que amplía la rejugabilidad del juego. La interacción con los jugadores se refuerza mediante una interfaz que muestra el estado de la partida en tiempo real, incluyendo el puntaje, los power-ups activos y el tiempo restante cuando aplique.

En conjunto, Neo-Snake propone una versión enriquecida del clásico, manteniendo su esencia pero apostando por una estructura más sólida, modular y escalable que permita futuras expansiones.

Modelo del mundo.

Identificación de entidades y características:

Con base en el enunciado se identifican las siguientes entidades (clases) y características (atributos):

Neo_Snake

- Dificultad
- Jugador
- Nivel
- Puntaje

Serpiente

- Cabeza
- Cuerpo
- Cola
- Segmentos
- Dirección
- Velocidad

Alimento

- Tipo
- Posición

Obstáculos

- Tipo
- Posición

PowerUp

- Tipo
- Posición
- Duración

Puntaje

- Puntos

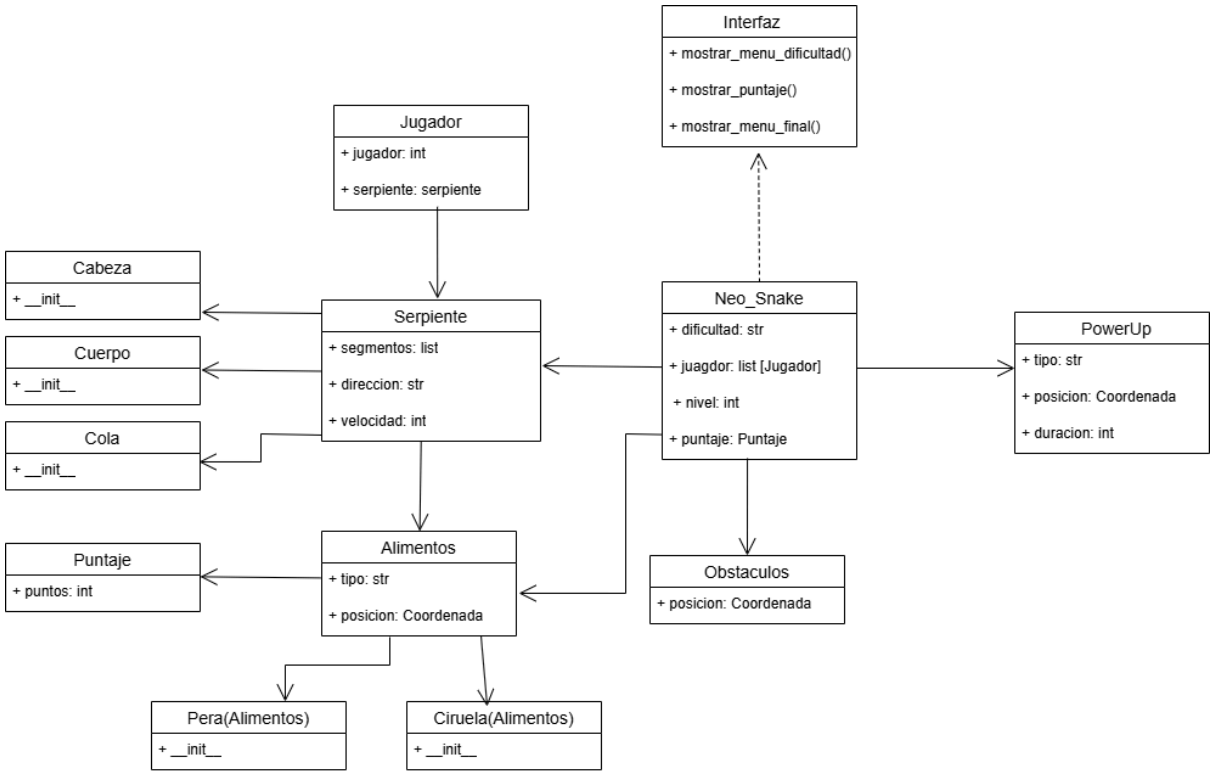
Jugador

- Jugadores

Interfaz

- Pantallas (menú, juego)

2. Modelo de las clases



Requisitos Funcionales Neo-Snake.

Requisito 1 Controlar el Movimiento de la Serpiente:

Nombre	R1- Controlar el movimiento de la serpiente
Resumen	Permite al jugador cambiar la dirección de la serpiente durante la partida, controlando de forma independiente la Cabeza y actualizando la posición del Cuerpo y la Cola.
Entradas	Dirección de movimiento (arriba, abajo, izquierda, derecha)
Resultados	**1.** Mover la Cabeza de la serpiente en la dirección seleccionada. **2.** Actualizar las posiciones del Cuerpo y la Cola siguiendo a la Cabeza. **3.** Cambiar la dirección cuando el jugador lo indique. **4.** Ejecutar el sistema de colisiones (R5) si la serpiente colisiona.

Descomposición:

Pasos	**Métodos**	**Responsable**
Cambiar dirección	cambiar_direccion(nueva_direccion)	Cabeza
Mover cabeza	actualizar_posicion()	Cabeza
Actualizar cuerpo	actualizar_posicion()	Cuerpo
Actualizar cola	seguir_cuerpo()	Cola

Requisito 2 Aumentar la Longitud de la Serpiente al Comer:

Nombre	R2- Aumentar la Longitud de la Serpiente al Comer
Resumen	Incrementa la longitud de la serpiente al consumir un alimento, añadiendo una nueva sección al Cuerpo.
Entradas	Colisión de la cabeza con el alimento.
Resultados	**1.** Eliminar el alimento del mapa. **2.** Añadir una nueva sección al final del Cuerpo. **3.** Generar un nuevo alimento en una posición aleatoria. **4.** Actualizar el puntaje del jugador (R3).

Descomposición:

Pasos	**Métodos**	**Responsable**
Detectar colisión con alimento	colisionar_con_alimento()	Cabeza
Aumentar longitud del cuerpo	agregar_segmento()	Cuerpo
Ajustar posición de la cola	actualizar_posicion	Cola
Generar nuevo alimento	generar()	Alimento
Actualizar puntaje	actualizar_puntaje(puntos)	Jugador

Requisito 3 Registrar y Mostrar el Puntaje del Jugador:

Nombre	R3-Registrar y Mostrar el Puntaje del Jugador
Resumen	El sistema debe registrar y actualizar el puntaje del jugador en función de su desempeño en la partida.
Entradas	Acción realizada por el jugador (comer alimento, usar power-up).
Resultados	**1.** Cada vez que la serpiente consume un alimento, el puntaje del jugador se incrementa en una cantidad fija. **2.** Al finalizar la partida, el sistema guarda el puntaje en la base de datos si está dentro de los mejores registros. **3.** Se muestra el puntaje en pantalla en tiempo real.

Descomposición:

Pasos	**Métodos**	**Responsable**
Incrementar puntaje al comer alimento	actualizar_puntaje(puntos: int)	Jugador
Registrar puntaje al finalizar partida	guardar_puntaje()	Puntaje
Mostrar puntaje en pantalla	mostrar_puntaje()	Interfaz

Requisito 4 Seleccionar el Nivel de Dificultad:

Nombre	R4- Seleccionar el Nivel de Dificultad
Resumen	El sistema debe permitir al jugador elegir entre varios niveles de dificultad antes de iniciar una partida.
Entradas	Nivel de dificultad seleccionado (Fácil, Medio, Difícil, Extremo).
Resultados	**1.** El sistema muestra una pantalla con las opciones de dificultad. **2.** El jugador selecciona un nivel de dificultad **3.** Según el nivel elegido, se configuran los parámetros del juego: **3.1 Fácil:** Baja velocidad, pocos obstáculos, más alimentos, sin límite de tiempo. **3.2 Medio**: Velocidad media, obstáculos moderados, menos alimentos, tiempo límite de 5 minutos. **3.3 Difícil:** Velocidad alta, muchos obstáculos, escasez de alimentos, tiempo límite de 3 minutos. **3.4 Extremo:** Velocidad muy alta, gran cantidad de obstáculos, muy pocos alimentos, tiempo límite de 1 minuto. **4.** Se inicia la partida con los parámetros configurados.

Descomposición:

Pasos	**Métodos**	**Responsable**
Mostrar opciones de dificultad	mostrar_menu_dificultad()	Interfaz
Seleccionar dificultad	mostrar_menu_dificultad()	Main
Configurar parámetros según dificultad	configurar_dificultad(nivel: str)	Main
Iniciar partida con configuración seleccionada	iniciar_juego()	Main

Requisito 5 Detectar Colisiones Durante la Partida:

Nombre	R5-Detectar colisiones durante la partida
Resumen	El sistema debe detectar cuando la serpiente colisiona con los bordes, su propio cuerpo u obstáculos y actuar en consecuencia.niveles de dificultad antes de iniciar una

	partida.
Entradas	**1.** Posición de la Cabeza de la serpiente en el mapa **2.** Posiciones del Cuerpo de la serpiente **3.** Posición de los obstáculos en el mapa **4.** Configuración del modo de juego
Resultados	**1.** Si la cabeza choca con los bordes del mapa: **1.1** Se finaliza el juego. **2.** Si la cabeza choca con alguna parte del cuerpo: **2.1** Se finaliza el juego. **3.** Si la cabeza choca con un obstáculo: **3.1** Se finaliza el juego

Descomposición:

Pasos	**Métodos**	**Responsable**
Detectar colisión con los bordes	verificar_colision_bordes() -> bool	Main
Detectar colisión con obstáculos	verificar_colision_obstaculo() -> bool	Main
Aplicar consecuencia de colisión	manejar_colision()	Main

Requisito 6 Generar Elementos del Juego en el Mapa:

Nombre	R6- Generar elementos del juego en el mapa
Resumen	El sistema debe generar alimentos, obstáculos y power-ups en posiciones aleatorias según las reglas del juego.
Entradas	**1.** Tamaño del mapa **2.** Posición de la serpiente **3.** Nivel de dificultad
Resultados	**1.** Se genera un nuevo alimento cada vez que la serpiente consume uno. **2.** Los obstáculos se generan al inicio del juego según la dificultad. **3.** Los power-ups aparecen de forma aleatoria y desaparecen tras cierto tiempo si no son recogidos.

Descomposición:

Pasos	Métodos	Responsable
Generar alimento en una posición aleatoria	generar_alimento() -> Posición	Alimento
Generar obstáculos según dificultad	generar_obstaculos() -> Lista de posiciones	Main
Generar power-ups en el mapa	generar_powerups() -> Posición	Power-ups
Desaparecer power-up tras tiempo límite	eliminar_powerup()	Power-ups

Requisito 7 - Ajustar Parámetros Según la Dificultad:

Nombre	R7- Ajustar parámetros según la dificultad
Resumen	El sistema debe ajustar la velocidad de la serpiente, la cantidad de obstáculos, alimentos y power-ups según el nivel de dificultad seleccionado al inicio del juego. Si alguna serpiente colisiona, el juego finaliza y se muestra un menú con opciones para reiniciar la partida o volver a seleccionar el nivel.
Entradas	Nivel de dificultad seleccionado
Resultados	1. En modo Fácil, la serpiente es lenta, con pocos obstáculos y alimentos. 2. En modo Medio, la velocidad es moderada, con más obstáculos y alimentos. 3. En modo Difícil, la velocidad es alta, con más power-ups y alimentos. 4. En modo Extremo, la velocidad es muy alta y hay muchos obstáculos. 5. Si alguna serpiente colisiona, el juego termina y aparece el menú final.

Descomposición:

Pasos	Métodos	Responsable
Asignar velocidad y cantidad de elementos según dificultad	configurar_dificultad(dificultad: str)	Main
Aplicar velocidad a la	set_velocidad(valor: int)	Main

serpiente		
Generar obstáculos y alimentos según dificultad	generar_elementos(dificultad: str)	Main
Verificar colisiones	verificar_colisiones()	Main
Finalizar juego si hay colisión	terminar_juego()	Main
Mostrar menú final con opciones	mostrar_menu_final()	Interfaz

Requisito 8 Mostrar Información en la Interfaz Gráfica:

Nombre	R8- Mostrar información en la interfaz gráfica
Resumen	El sistema debe mostrar información relevante al jugador durante la partida y al finalizar.
Entradas	**1.** Estado actual del juego **2.** Puntaje del jugador
Resultados	**1.** Durante la partida, se muestra en pantalla: **1.1** Puntuación actual **2.** Al finalizar la partida, se muestra un resumen con: **2.1** Puntaje final **2.4** Botón para reiniciar el juego o volver al menú

Descomposición:

Pasos	**Métodos**	**Responsable**
Mostrar puntaje en pantalla	actualizar_puntaje(puntos: int)	Interfaz
Mostrar tiempo restante	actualizar_tiempo_visual()	Interfaz
Mostrar power-ups activos	actualizar_powerups_visual()	PoweUps
Mostrar resumen final del juego	mostrar_resumen(puntaje: int)	Puntajes