SnakeFight – Análisis Técnico del Módulo Core

Descripción del problema

El proyecto *SnakeFight* crea una versión competitiva del juego clásico Snake, donde un jugador humano compite contra una serpiente controlada por IA dentro de un tablero discreto. El objetivo principal es mantener la jugabilidad fluida mientras se integran estructuras de datos claras como listas ligadas, árboles de decisión y hash maps conceptuales para gestionar la lógica del juego.

Justificación de las estructuras utilizadas

Listas ligadas: Las serpientes se representan usando una lista ligada simple mediante la estructura `SnakeNode`. Esto permite insertar un nodo en la cabeza en tiempo constante y remover la cola sin necesidad de mover grandes bloques de memoria. Aunque Python ofrece listas dinámicas, usar una lista ligada hace que la estructura del cuerpo de la serpiente sea más natural y extensible.

Árbol de decisión para la IA: La función `ai_choose_direction` implementa un árbol de decisión muy compacto de dos niveles. Primero selecciona la dirección ideal hacia la comida y después evalúa la seguridad de cada posible dirección en orden de prioridad. Esto mantiene la IA reactiva y ligera sin algoritmos pesados.

Hash maps con diccionarios: Los usuarios se almacenan internamente como diccionarios después de leerse desde un archivo CSV. Los diccionarios funcionan como hash maps, permitiendo acceso a sus campos en tiempo constante. El sistema usa además un hash derivado de SHA-256 para simular IDs únicos.

Arquitectura del programa

El módulo Core se divide en piezas bien organizadas:

- **utils.py**: Contiene las estructuras principales del juego: nodos, serpientes, comida, botones y sistema de usuarios.
- **game.py**: Ejecuta el bucle principal del juego, administra colisiones, movimiento y renderizado.
- **menu.py**: Controla el menú de inicio, botones y logo.
- **config.py**: Espacio para configuraciones futuras o pantallas auxiliares.

El flujo central comienza en `run_game`, que coordina todos los elementos de lógica y visualización.

Análisis de complejidad temporal y espacial

Movimiento de la serpiente:

- Insertar nuevo nodo en la cabeza: O(1)
- Recorrer la lista ligada para eliminar la cola: O(n)

Colisiones:

- Comparar cabeza con cuerpo: O(n)
- Comparar serpientes entre sí: O(n)
- Colisiones contra bordes: O(1)

IA: Evaluación del árbol de decisión: O(1)

Comida: Las posiciones se almacenan en un conjunto, permitiendo búsquedas O(1).

Espacio: La serpiente ocupa O(n) nodos. El resto de las estructuras tienen tamaño pequeño y estático.

Pruebas realizadas y casos borde

Para garantizar un comportamiento estable, se consideraron pruebas (manuales y simuladas) sobre distintos componentes del sistema:

Pruebas de movimiento: Se verificó que la serpiente nunca pueda moverse hacia atrás respecto a su dirección actual. También se probó que la IA mantuviera direcciones válidas incluso cuando la comida se genera en esquinas.

Pruebas de colisiones: Se realizaron casos donde ambas serpientes se mueven hacia la misma celda, comprobando que el juego priorice correctamente la detección según el orden de actualización. Se ejecutaron pruebas con serpientes enormes para asegurar que la detección de colisiones internas (self-hit) no fallara.

Pruebas de comida: Se generaron situaciones donde casi todo el tablero estaba ocupado para confirmar que la comida no apareciera en posiciones prohibidas. También se evaluó consumo simultáneo entre IA y jugador.

Casos borde:

- Generación de comida en la última celda libre.
- IA encerrada en un corredor de un solo bloque.
- Jugador quedando atrapado por error en una esquina.
- Reinicio correcto de IA tras colisiones forzadas múltiples.

- Tablero vacío donde ambas serpientes están alineadas verticalmente y apuntando al mismo objetivo.

Conclusión

El módulo Core de *SnakeFight* usa estructuras clásicas como listas ligadas, árboles de decisión simples y hash maps ligeros mediante diccionarios. Esto permite una arquitectura clara, ordenada y fácil de extender, manteniendo un desempeño rápido y un código accesible y entendible.