Universidad Nacional de Costa Rica

Escuela de informática

Estructura de Datos

# Proyecto I, Estructura de Datos, Prompts

Estudiantes:

Javier Garita Granados

Josué Peñaranda Alvarado

Docente: José Calvo Suarez

## *Chatbots utilizados:*

BlackBox.AI, GitHub Copilot y chatgpt. Se utilizaron las extensions de agente de BlackBox.AI y Github Copilot, en cuanto a Chatgpt se usó para formalizar prompts y resolver preguntas formales acerca de la librería Pygame y Python para nuestro entendimiento y aprendizaje.

## *Prometes de IA utilizados:*

1. Mejóreme la estructura que tengo para cargar las Apis, para tener una mayor eficiencia, donde extraiga la Api, la verifique y mandar la data que es todo el bloque del contenido a otra al main para que la clase Mapa extraiga eso y lo crea

2. Crea una clase Mapa con celdas, métodos tile_at(x,y)/get_tile(x,y) y un draw_map(pantalla, mapa, camara, tile_size) que dibuje sprites por tipo de tile.

3. Generador simple de mapas tile-based y herramienta para editar guardar/cargar mapas en JSON con capas (suelo, objetos, puertas).

4. Implementa una clase Cámara que escale/mueva superficies (apply_surface) y devuelva rects/surfaces adaptados al viewport.

5. Crea un asset loader que cargue y cachee Surfaces (no recargar cada frame) y exponga sprites por clave (ej. mapa.sprites['B']).

6. Implementa Player/Repartidor con position, velocidad, rect, move(dx,dy) y comprobación de colisiones con tiles sólidos.

7. Al terminar movimiento, chequear curr = tile_at(x,y) y front = tile_at(x+dx,y+dy) y cambiar player.imagen_mostrar a 'personaje' si está en B o a 'repartidor' si está en exterior (mantener orientación).

8. Si jugador se para sobre un paquete o buzón, actualizar pedido seleccionado automáticamente al color/ID del objeto y reproducir sonido corto.

9. Implementa gestor_pedidos que genere pedidos con tiempos límite, prioridad y haga queue; exponer obtener_disponibles(timestamp).

10. Proveer funciones de ordenamiento (merge_sort, heap_sort) para ordenar la lista de pedidos por diferentes criterios (deadline, prioridad).

11. Crear HUD que muestre dinero, meta, barra de progreso para recoger/entregar y botones interactivos (con colisiones de mouse).

12. Guardar/cargar estado de juego (repartidor, pedidos activos, mapa) en JSON y binario; mostrar mensajes de éxito/error en pantalla.

13. Quiero que el juego funcione sin internet. Cada vez que se conecta a la API, guarda clima, mapa y pedidos en /data/ (fijo) y en /api_cache/ (con fecha). Si no hay conexión, usa la copia más reciente de /api_cache/.

14. Guarde en un JSON de puntajes, para en un futuro ver los puntajes de cada juego que la persona jugo y en futuro, poder comparar quienes son los mejores puntajes.

15. Clase Clima que use matriz de transición de Markov para cambiar estados (clear, rain, storm, fog, wind), genere bursts con intensidad y haga transiciones suaves.

16. Sistema de partículas para lluvia/niebla/nieve/viento; partículas respetan intensidad del clima y se renderizan encima del mapa.

17. Aplicar multiplicadores/penalizaciones por clima (velocidad, resistencia, prob. de fallo) usando Clima.get_multiplicador().

18. Integrar pygame.mixer: ¡en menú reproducir ‘8bit Menu Music’ loop (-1), al entrar al juego cargar y reproducir ‘Quiz! - Deltarune (8-bit Remix)' en bucle; controlar volumen y stop antes de cambiar.

19. Añadir sonidos para botones, confirmaciones, victoria y errores; reproducir con sound.play() y no bloquear el loop.

20. Mejorar rendimiento del programa corrigiendo errores y optimizando distintas operaciones.

21. Necesito que implementemos un sistema nuevo para el juego, necesito que implementemos un sistema que permita devolverse paso a paso a lo que ha pasado en los 15 ultimos movimientos del jugador, incluyendo si entrega un paquete, si acepta un paquete, si cambio el clima, etc. Para lograrlo quiero que implementes lo siguiente:

-> Crea un sistema de pila que tenga un máximo de 15 elementos de "snapshots", un nuevo objeto que va seguir el patron de diseño de memento para "capturar" todos los datos que nececitamos para devolvernos entre los pasos. Cuando se llena y se quiere ingresar un nuevo elementos, los elementos del 1 al 14 pasas un espacio para adelante y el 15 es sobre escrito por el 14, para finalmente incluirlo el nuevo elemento en el

-> Para las snapshot puedes aprovechar el sistema de guardado y carga ya existente en el programa.