

## Bitácora de Prompts

Proyecto #1

Estructuras de Datos

CRN: 51219

Estudiantes: Ximena Ortiz Llanes

Luis Diego Alvarado Salazar

Profesor: José Pablo Calvo Suárez

Prompt: ¿Cómo puedo destacar un trabajo seleccionado en el inventario?

Respuesta: Para destacar un trabajo seleccionado en el inventario, actualiza el método `draw_inventory`, para visualmente distinguir el trabajo en `inventory.selected_index`, por ejemplo, cambiar el color del fondo del trabajo seleccionado.

Resultado: Se agrega código para poder cambiar el color y cambiar la forma a un rectángulo por medio de la función `pygame.draw.rect()`, así se destaca el trabajo actualmente seleccionado en el inventario.

Prompt: ¿Cómo puedo cargar los datos más recientes en el caché de un JSON?

Respuesta: Realiza una función `load_from_cache()`, la cual cargue los contenidos y los copie a `json_files` para tener una consistencia.

Resultado: Se implementa una función que restaura los últimos datos en el cache para la clase específica. Un enfoque modular para guardar datos de API JSON es en dos lugares: una carpeta principal (`json_files`) y una carpeta de cache (`api_cache`) con una marca de tiempo como el nombre del archivo. Se utiliza `save_api_data` para guardar los datos en ambas carpetas y se usa `datetime.now().strftime('%Y%m%d_%H%M%S')` para la marca de tiempo.

Prompt: ¿Qué es una cadena de Markov y como puedo implementarlo en un videojuego por medio de un JSON?

Respuesta: Una cadena de Markov es un modelo matemático que describe un sistema que cambia de estado a otro de forma probabilística, donde el siguiente estado depende sólo del estado actual. En una clase `Weather`, cargas un archivo JSON que contenga los climas que quieres utilizar y se aplica una lógica de cambio automático.

Resultado: En la clase `Weather`, se implementan métodos diferentes para: actualizar el clima, la transición del mismo hacia otro y controlar la duración del clima (de 45 a 60 segundos), además, se dibuja en el HUD el clima actual.

Prompt: ¿Cómo podría realizar la lógica tras los trabajos basado en la relación que tiene con el mapa, inventario y el estado del jugador?

Respuesta: A como vienen los trabajos en el JSON, cada pedido es una estructura de datos con identificador, coordenadas de recogida y entrega, peso, prioridad, hora límite y tiempo. Se debe crear una clase `Job` para encapsular la lógica y no

depender del JSON directamente. Algunas estructuras de datos recomendadas para gestionar la lógica son:

- Cola: pedidos que van saliendo con el tiempo.
- Heap: mostrar o pedir el pedido más urgente.
- Lista enlazada: manejar el inventario del repartidor.

Cada trabajo tiene coordenadas de pickup y dropoff, así que puedes dibujar íconos pequeños sobre el mapa: un paquete en la ubicación pickup y un destino (por ejemplo, una casa) en dropoff.

Resultados: Se implementa la lógica del trabajo en la clase Job, se integra una lista `jobs_pending` para los trabajos que aún no están activos. Se agrega una cola `available_jobs` para los pedidos que se pueden aceptar. En el inventario, se agrega una lista enlazada para visualizar los pedidos que el jugador lleva actualmente. Finalmente, se integra otra lista `delivered_jobs` que sirve como historial para puntuación y reputación del usuario.

Prompt: ¿Cómo puedo mostrar un scoreboard, que sea un top 5 mejores y, además, destacar una nueva entrada en el mismo en una pantalla?

Respuesta: para mostrar un scoreboard, usa la clase Scoreboard para obtener los puntajes y dibujarlos en la pantalla. Destaca el nuevo puntaje con un color diferente.

Código de ejemplo:

```
import pygame
```

```
from scoreboard import Scoreboard
```

```
def draw_scoreboard(screen, new_score):
    scoreboard = Scoreboard('data/json_files/scores.json')
    highlight_idx = scoreboard.add_score(new_score)
    scores = scoreboard.get_scores()

    font = pygame.font.SysFont(None, 36)
    y = 100
    screen.fill((0, 0, 0))
    title = font.render("TOP 5 SCORES", True, (255, 255, 0))
    screen.blit(title, (100, y))
    y += 50

    for idx, entry in enumerate(scores[:5]):
        if idx == highlight_idx:
            color = (255, 255, 0) # Highlight color
            prefix = "► "
        else:
            color = (255, 255, 255)
            prefix = " "
        text = font.render(f"{prefix}{idx+1}. ${entry['score']}", True, color)
```

```
screen.blit(text, (100, y))  
y += 40
```

```
pygame.display.flip()
```

Esto dibuja el scoreboard y destaca la nueva entrada en amarillo con un prefijo

Prompt: Se consultó a la AI a obtener la información de una api y guardarla en json con tres endpoints

<https://chatgpt.com/share/68d0af89-dcf0-800b-af71-d0b59efe5bb3>