## **IA Bitacora**

## Mariela:

**Prompt:**

Necesito que después que la partida termine se pueda volver al menú principal, esto es lo que actualmente pasa (inserte imagen), lo que quiero es que cuando presione la tecla esc se devuelva al menú principal

**Edición:**

src/logic/game.py

Añade una “razón de salida” y captura ESC cuando game\_over es True.

# en \_\_init\_\_ (al final)  
self.exit\_reason = "menu" # por defecto: volver al menú

# en run() – devuelve la razón al terminar  
def run(self):  
 while self.running:  
 dt = self.clock.tick(60) / 1000.0  
 self.player\_moved\_this\_frame = False  
 self.handle\_input()  
 self.update(dt)  
 self.draw()  
 return self.exit\_reason

# en handle\_input() – reemplaza SOLO la parte del loop de eventos por esto  
for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 self.exit\_reason = "exit"  
 self.running = False  
 return  
  
 # >>> NUEVO: cuando ya terminó la partida, ESC vuelve al menú <<<  
 if self.game\_over:  
 if event.type == pygame.KEYDOWN and event.key == pygame.K\_ESCAPE:  
 self.exit\_reason = "menu"  
 self.running = False  
 # no procesar más input de gameplay si game\_over  
 continue  
  
 if event.type == pygame.KEYDOWN:  
 # ... (tu manejo normal de teclas durante el juego)  
 pass

### **main.py**

Cuando lances el juego, **usa lo que devuelve game.run()** para decidir si vuelves al menú o cierras la app.

# al iniciar una nueva partida  
game = Game(player\_name)  
result = game.run()  
if result == "exit":  
 return # o break si estás en un while del menú  
# si fue "menu", simplemente continúa y el menú se vuelve a dibujar

Y si tienes una función para “Cargar partida”, haz lo mismo:

game = Game(saved\_name)  
if game.load\_game(1):  
 result = game.run()  
 if result == "exit":  
 return # o break

**Prompt:**

Al guardar el juego, la idea es que sea como salvarlo es decir que cierre y luego en el menú principal en la parte de cargar el juego se pueda volver acceder a él, sin embargo, no está pasando aparece un mensaje del juego fue cargado, pero continua y si cierro y el juego y voy al menú aparece un error al intentar cargarlo

**Edición:**

def handle\_input(self): """Maneja input del jugador (eventos + movimiento por polling).""" for event in pygame.event.get(): # Cerrar ventana -> salir completamente if event.type == pygame.QUIT: self.exit\_reason = "exit" self.running = False return

# Si el juego ya terminó (victoria/derrota): ESC -> volver al menú  
 if self.game\_over:  
 if event.type == pygame.KEYDOWN and event.key == pygame.K\_ESCAPEself.exit\_reason = "menu"  
 self.running = False  
 continue # no procesar gameplay cuando ya terminó  
  
 # Atajos/acciones por tecla (NO movimiento)  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_n:  
 self.player.inventory.view\_next\_order()  
 self.show\_message("Siguiente pedido")  
 elif event.key == pygame.K\_p:  
 self.player.inventory.view\_prev\_order()  
 self.show\_message("Pedido anterior")  
 elif event.key == pygame.K\_s:  
 self.player.inventory.sort\_inventory(lambda o: o.priority)  
 self.show\_message("Ordenado por prioridad")  
 elif event.key == pygame.K\_d:  
 self.player.inventory.sort\_inventory(  
 lambda o: self.\_normalize\_datetime(o.deadline)  
 )  
 self.show\_message("Ordenado por deadline")  
  
 elif event.key == pygame.K\_a:  
 self.accept\_order\_at\_location()  
  
 elif event.key == pygame.K\_RETURN:  
 self.complete\_delivery()  
  
 elif event.key == pygame.K\_c:  
 if self.player.cancel\_order():  
 self.show\_message("Pedido cancelado (-4 reputación)")  
  
 # --- GUARDAR: F5 guarda y vuelve al menú ---  
 elif event.key == pygame.K\_F5:  
 self.save\_game(1)  
 self.show\_message("Juego guardado. Volviendo al menú...")  
 self.exit\_reason = "menu"  
 self.running = False  
  
 # --- CARGAR: solo desde el MENÚ (F9 no carga) ---  
 elif event.key == pygame.K\_F9:  
 self.show\_message("Para cargar, usa el menú principal")  
  
 elif event.key == pygame.K\_U:  
 state = self.game\_state.undo(1)  
 if state:  
 self.restore\_state(state)  
 self.show\_message("Deshacer último movimiento")  
  
# Movimiento por polling (una casilla por pulsación)  
if not self.game\_over:  
 keys = pygame.key.get\_pressed()  
 dx = (1 if keys[pygame.K\_RIGHT] else 0) - (1 if keys[pygame.K\_LEFT] else 0)  
 dy = (1 if keys[pygame.K\_DOWN] else 0) - (1 if keys[pygame.K\_UP] else 0)  
  
 if not hasattr(self, "\_arrow\_was\_pressed"):  
 self.\_arrow\_was\_pressed = False  
 arrow\_pressed\_now = (dx != 0 or dy != 0)  
  
 if arrow\_pressed\_now and not self.\_arrow\_was\_pressed:  
 new\_x = self.player.x + dx  
 new\_y = self.player.y + dy  
  
 if not self.city.is\_blocked(new\_x, new\_y):  
 if self.player.can\_move():  
 self.player.consume\_stamina(self.current\_weather)  
 self.player.x = new\_x  
 self.player.y = new\_y  
 self.player\_moved\_this\_frame = True  
  
 self.game\_state.save\_state(  
 self.player,  
 self.player.inventory,  
 self.elapsed\_time,  
 self.current\_weather  
 )  
 self.check\_delivery\_points()  
 else:  
 self.show\_message("¡Exhausto! Descansa para recuperarte")  
 else:  
 self.show\_message("No puedes moverte ahí")  
  
 self.\_arrow\_was\_pressed = arrow\_pressed\_now

Nota: Eliminé cualquier llamada a self.load\_game(1) desde el juego. Cargar solo se hace por el menú.

1. Game.save\_game(...) — ya pasas player\_name (bien) Asegúrate que tu save\_game llame a GameState.save\_game con player\_name (por lo que pegaste, ya está así):

python Copiar código def save\_game(self, slot): game\_data = { 'elapsed\_time': self.elapsed\_time, 'weather\_state': self.current\_weather, 'weather\_timer': self.weather\_timer, 'game\_start\_datetime': self.game\_start\_datetime.isoformat(), } self.game\_state.save\_game(slot, self.player, self.player.inventory, game\_data, self.player\_name) 3) main.py — cargar solo desde el menú (tu versión ya hace eso) Tu load\_and\_start\_game actual ya respeta la carga por menú, pasando el dict al Game.load\_game(data) y luego game.run():

python Copiar código def load\_and\_start\_game(screen, clock): save\_path = Path("saves/slot1.sav") if not save\_path.exists(): show\_message\_screen(...); return "menu"

try:  
 game\_state = GameState()  
 data = game\_state.load\_game(1)  
 if data:  
 player\_name = data.get('player\_name', 'Player')  
 game = Game(player\_name)  
 game.load\_game(data) # <--- Carga desde dict  
 print("Partida cargada exitosamente!")  
 return game.run() # devuelve 'menu' o 'exit'  
 else:  
 show\_message\_screen(...); return "menu"  
except Exception as e:  
 show\_message\_screen(...); return "menu"

**Prompt:**

mi idea es que solo con f5 se guarde, y está bien que haya un límite de 3 partidas guardas, lo que me gustaría es que en cargar partida se puede seleccionar cualquiera de las tres como un tipo menú para visualizarlas

**Edición:**  
  
 **GameState**

* + MAX\_SLOTS = 3.
  + Función list\_slots\_metadata() para vista previa por slot (nombre, ingresos, reputación, tiempo jugado, fecha).
  + Función auto\_save(...): guarda con **F5** usando el **primer slot libre**; si los 3 están ocupados, sobrescribe el **más antiguo**.
  + Uso explícito por slot mediante save\_game(slot, ...) y load\_game(slot).
* **Game**
  + Tecla **F5** → llamada a auto\_save(...).
  + Carga in-game deshabilitada (**F9** muestra aviso: “cargar desde el menú”).
  + Overlay de guardado breve; al finalizar, retorno automático al **menú** (exit\_reason = "menu").
* **Menú principal (main.py)**
  + Opción **Cargar Partida** con selector de **Slots 1–3** y **vista previa** por slot.
  + Navegación: ↑/↓, teclas **1–3**, **Enter** para confirmar, **ESC** para cancelar.
  + Al seleccionar slot válido: load\_game(slot) y arranque de la partida; manejo de error si el slot no existe/corrupto.
* **UX**
  + Mensaje de confirmación al guardar: “**Guardado en Slot X. Volviendo al menú…**”.
  + Regla: **guardar solo con F5**; **cargar solo desde el menú**.

**Prompt:** todo está muy bien, sin embargo, creo que si cargo una partida y la termino debería de ser eliminada del slot

**Edición:**  
def delete\_slot(self, slot) -> bool: """Elimina el archivo del slot indicado. Devuelve True si se borró.""" path = Path(f"saves/slot{int(slot)}.sav") try: if path.exists(): path.unlink() print(f"Slot {slot} eliminado.") return True except Exception as e: print(f"No se pudo eliminar el slot {slot}: {e}") return False 2) Game: marca de qué slot fue cargada la partida a) En el **init** de Game, añade el atributo: python Copiar código # Slot desde el que se cargó esta partida (None si es nueva) self.loaded\_slot = None b) En end\_game, borra el slot si venía de carga Dentro de end\_game(self, victory), al final (después de guardar el score y los prints), agrega:

python Copiar código # Si esta partida fue cargada desde un slot, lo eliminamos al terminar if getattr(self, "loaded\_slot", None): self.game\_state.delete\_slot(self.loaded\_slot) self.loaded\_slot = None Esto libera automáticamente el slot cuando la partida cargada termina (ya sea victoria o derrota). No afecta partidas nuevas ni guardados que hagas con F5.

main.py: pasa el número de slot al Game cuando cargas En tu función load\_and\_start\_game(...), después de crear el Game y antes de game.run(), asigna el slot:

python Copiar código data = gs.load\_game(slot) if data: player\_name = data.get('player\_name', 'Player') game = Game(player\_name) game.load\_game(data) game.loaded\_slot = slot # <<--- IMPORTANTE print(f"Partida cargada exitosamente desde Slot {slot}!") return game.run()

**Prompt:**

el proyecto está prácticamente completo, sin embargo, quiero agregar unas cosas, por ejemplo, me gustaría una ventana antes de entrar a la partida que explique cómo se juega, también me gustaría que exista una ventana para ingresar el nombre del jugador, podría ser después o antes de las instrucciones

**Edición:**  
**Archivo src/main.py**

Se realizaron las siguientes modificaciones:

1. **Se importó una nueva clase:** Se agregó la línea from src.logic.input\_box import InputBox para poder usar la clase que maneja la entrada de texto.
2. **Se añadió una nueva función ask\_player\_name:** Esta función crea una pantalla con una caja de texto donde el usuario puede escribir su nombre. El nombre se retorna cuando el usuario presiona Enter.
   1. **Función ask\_player\_name:**
      1. Crea una instancia de InputBox.
      2. Dibuja el título del juego y un mensaje pidiendo el nombre del jugador.
      3. Maneja los eventos del teclado para procesar la entrada de texto.
      4. Retorna el nombre del jugador o "Player" si no se ingresa ninguno.
3. **Se añadió una nueva función show\_instructions\_screen:** Esta función crea una pantalla de instrucciones con un fondo degradado y el texto de los controles. Espera a que el usuario presione una tecla o haga clic para continuar.
4. **Se modificó la función main:**
   1. En el bloque if action == "start\_game":, se agregó una llamada a ask\_player\_name para obtener el nombre del jugador antes de iniciar el juego.
   2. Luego, se llamó a show\_instructions\_screen para mostrar las instrucciones.
   3. Finalmente, la instancia de Game se crea con dos argumentos: game = Game(player\_name, screen). Esto se hizo para que la clase Game supiera el nombre del jugador y pudiera acceder a la pantalla.
   4. Se corrigió la llamada a Game() en el bloque elif action == "load\_game": para que también se cree con los argumentos correctos (player\_name y screen).

### **Archivo src/logic/game.py**

Se realizaron las siguientes modificaciones:

1. **Se modificó el constructor \_\_init\_\_:**
   1. Se cambió la definición a def \_\_init\_\_(self, player\_name, screen): para aceptar el nombre del jugador y el objeto screen como argumentos.
   2. Se eliminó la línea pygame.init() del constructor, ya que ahora el inicializador del juego se llama solo una vez en main.py.
   3. Se eliminó la línea self.screen = pygame.display.set\_mode((1200, 800)) para evitar duplicar la creación de la ventana.
   4. Se agregó la línea self.screen = screen para usar la pantalla que se pasa desde main.py.
   5. Se agregó la línea self.player\_name = player\_name para almacenar el nombre del jugador.
2. **Se modificó la función end\_game:**
   1. Se cambió la llamada a GameState.save\_score("Player", ...) para que use la variable self.player\_name en su lugar. Esto asegura que el nombre del jugador ingresado se guarde en la tabla de récords

### **Archivo src/logic/input\_box.py**

Se creó un nuevo archivo para encapsular la lógica de la caja de texto. Esta clase InputBox maneja los eventos del mouse y el teclado, así como el renderizado de la caja de texto y el texto en la pantalla.

**Prompt:**

podrías hacer un menú con esa imagen de fondo (inserte imagen), me gustaría que fuera de forma animada, con movimiento

**Edición:**

Se crearon varias clases para los elementos animados del menú principal:

AnimatedCyclist, AnimatedCloud, AnimatedWindow y RoadLine.

La clase MainMenu se modificó para incluir un fondo procedural y gestionar la lógica de animación y el dibujo de todos estos elementos.

Se implementaron efectos visuales como el balanceo del ciclista y los árboles, la rotación de las ruedas y la pulsación del título.

## Alcides:

**Prompt:**  
Help me make the movement so the player only moves n spaces in game ticks  
**Edición:**  
self.player\_is\_moving = False

self.player\_start\_pos = (spawn\_x, spawn\_y)

self.player\_target\_pos = (spawn\_x, spawn\_y)

self.movement\_progress = 0.0

self.movement\_speed = 0.15 # How fast to move (0.1 = slow, 0.5 = fast)  
 # Movement timing control

self.last\_movement\_time = 0

self.movement\_delay = 200 # milliseconds between movements (200ms = 5 moves/second)

**Prompt:**  
make the player be able to use the "Z" key to revert movess, and its move its going to be the last movement done, that is save in the lifo queue "self.player\_movements".

**Edición:**

Actions taken

Added a LIFO list self.player\_movementes to track last movement directions as (dx, dy) in pixels; kept your existing self.player\_movements queue of positions as-is.

Implemented Z-key repeat:

When Z is held and the stack isn’t empty, we peek the last direction and attempt the same move.

If the move is valid (walkable), we pop it at the moment we start the movement.

Normal WASD moves push their direction onto the stack.

Capped history length at 50 to avoid unbounded growth.

Fixed timer display bug: minutes are now remaining\_ms // 60000 (not double-divided).

Updated the info panel and console “controls” to include Z help.

Fixed two indentation issues that crept in during edits.

**Tribeth:**

**Prompt:**

Help me choose the method that best suits the management of an inventory that goes back and forth through the objects.

**AI answer:**

**Why Doubly Linked List?**

The inventory system requires frequent bidirectional navigation (next/previous order using N/P keys) and deletion of the current order when completing deliveries.

**Key Advantages**

**Bidirectional Navigation: O(1)**

Moving forward/backward between orders is constant time thanks to next and prev pointers

Singly linked list would require O(n) to go backward

**Delete Current Order: O(1)**

Removing the active order only requires reconnecting pointers

Arrays would need O(n) to shift elements

**Insert at End: O(1)**

Adding new orders is constant time with a last pointer

**Complexity Analysis**

**Time Complexity:**

Add order: O(1)

Navigate next/prev: O(1)

Complete order: O(1)

Sort inventory: O(n²) - Insertion Sort

**Space Complexity:**

Memory per node: ~64 bytes

Extra overhead vs singly linked: 8 bytes per node (prev pointer)

For 10 orders: ~80 extra bytes (12.5% more)

Total: O(n)

**Prompt:**

Ayudame a corregir la implementación de la comparación de zonas horarias de los pedidos

**Solución:**

El deadline viene de datetime.fromisoformat(node.order.deadline) y probablemente tiene formato como "2025-09-29T12:00:00+00:00" (con zona horaria), mientras que current\_game\_time se calcula sin timezone.

**Solución**

Te propongo dos opciones:

**Opción 1: Hacer todos los datetimes "naive" (más simple)**

Modifica la función update() en game.py para eliminar la información de timezone al comparar:

Fix: Comparación de datetime (game.py)

Código

# En game.py, reemplaza la sección de verificación de pedidos expirados # (aproximadamente líneas 235-244) con este código: current\_game\_time = self.get\_current\_game\_datetime() if self.player.inventory.first: node = self.player.inventory.first

**Opción 2: Hacer todos los datetimes "aware" (más robusta)**

Si prefieres mantener la información de timezone (mejor práctica para aplicaciones reales), podemos modificar get\_current\_game\_datetime():

Fix alternativo: datetime con timezone (game.py)

Código

# Alternativa: Agregar timezone a current\_game\_time # En game.py, reemplaza get\_current\_game\_datetime(): from datetime import datetime, timedelta, timezone def get\_current\_game\_datetime(self): """Obtiene la fecha/hora actual del juego basada en

**Mi recomendación**

Te recomiendo la **Opción 1** porque:

1. Es más simple de implementar
2. No requiere cambios en múltiples lugares
3. Para este juego no necesitamos manejo de zonas horarias reales

**Implementación paso a paso**

1. **Abre src/logic/game.py**
2. **Busca la línea 235** (aproximadamente) donde dice:

python

if self.player.inventory.first:

node = self.player.inventory.first

orders\_to\_expire = []

while node:

deadline = datetime.fromisoformat(node.order.deadline)

if current\_game\_time > deadline: *# ← AQUÍ ESTÁ EL ERROR*

1. **Reemplaza con**:

python

if self.player.inventory.first:

node = self.player.inventory.first

orders\_to\_expire = []

while node:

deadline = datetime.fromisoformat(node.order.deadline)

*# Convertir a naive si tiene timezone*

if deadline.tzinfo is not None:

deadline = deadline.replace(tzinfo=None)

if current\_game\_time > deadline:

1. **También actualiza player.py** en la función complete\_delivery() (línea ~140):

python

deadline = datetime.fromisoformat(order.deadline)

*# Convertir a naive si tiene timezone*

if deadline.tzinfo is not None:

deadline = deadline.replace(tzinfo=None)

time\_diff = (deadline - current\_time).total\_seconds()