

5 de octubre de 2025

Estructuras de Datos

Prof. José Calvo Suarez

Universidad Nacional de Costa Rica

Documento bitácora de IA

Arianna Barquero Ortiz

Sebastián Castro Guevara

Yoel Hernández Zamora

UNA

como puedo hacer para que se desplace de cuadro a cuadro de manera mas fluida (como en PacMan)

MapaPrueba.py

Para lograr un movimiento más fluido (tipo Pac-Man), debes separar la lógica de movimiento en dos partes:

1. **Destino objetivo:** Cuando el usuario presiona una tecla, estableces la celda destino (row, col) a la que el repartidor debe ir.
2. **Animación:** En cada frame, mueves el sprite poco a poco hacia el centro de la celda destino, usando una velocidad constante.

Esto se logra usando el método `on_update` de la ventana para animar el movimiento. El sprite debe tener atributos de posición actual (en píxeles) y destino (en píxeles), y moverse gradualmente hasta llegar al destino.

quiero cambiar para que en vez de dibujar el sprite de un edificio y parque con un simple color, le coloque un png

Para mostrar imágenes PNG en vez de colores para edificios y parques, debes:

1. Cargar los sprites (por ejemplo, `Sprite` o `load_texture`) para cada tipo de celda (edificio, parque, etc.) en el método `__init__`.
2. En el método `on_draw`, en vez de dibujar un rectángulo, dibujar el sprite o textura correspondiente en cada celda.

haz lo mismo de slots y eso dandole a la L y que diga "cual espacio deseas cargar?"

Adding a load slot popup triggered by 'L', with the message "¿Cuál espacio deseas cargar? Selecciona 1, 2 o 3", and key handling for 1-3 and ESC.

MapaPrueba.py +59 -0

```
def mostrar_popup_cargar(self):
    """Muestra el popup para seleccionar cual de cargar si no hay otros popups activos."""
    if getattr(self, "mostrar_popup_cargar", False):
        return
    if getattr(self, "mostrar_popup_cargar", False):
        return
    if getattr(self, "mostrar_popup_cargar", False):
        return
    if getattr(self, "mostrar_popup_cargar", False):
        return
    if getattr(self, "mostrar_popup_cargar", False):
        return
    self.mostrar_popup_cargar = True
    self.clicked_cargar_seleccionado = None

def draw_popup_cargar(self):
    if not getattr(self, "mostrar_popup_cargar", False):
        return
    ancho = 200 - 100, 100
    x = self.window_width // 2 - ancho // 2
    y = self.window_height // 2 - 100 // 2
    arcade.draw_lwrh_rectangle_filled(x, y, ancho, alto, arcade.color.DARK_GRAY, 200)
    arcade.draw_lwrh_rectangle_outline(x, y, ancho, alto, arcade.color.WHITE, 3)
    texto = "¿Cuál espacio deseas cargar? Selecciona 1, 2 o 3"
    arcade.draw_text(texto, x + 50, y + 50, arcade.color.WHITE, 16, anchor="center", align="center", justify="center", width=ancho-10)
    if self.clicked_cargar_seleccionado is not None:
        arcade.draw_text("Seleccionado: " + str(self.clicked_cargar_seleccionado), x + 20, y + 20, arcade.color.YELLOW, 14)
        arcade.draw_text("Presiona ESC para cancelar", x + 20, y + 30, arcade.color.LIGHT_GRAY, 12)

def cargar_en_slot(self, slot_id):
    """Repartidor de carga final de implementación la logica de restaurar el estado."""
    print(f"Repartidor de carga final de implementación la logica de restaurar el estado.")
    self.clicked_cargar_seleccionado = slot_id
    self.mostrar_popup_cargar = False
```

haz que si se mantiene presionada la tecla, se mueva constantemente

Para lograr que el personaje se mueva constantemente mientras se mantiene presionada una tecla, debes:

1. Registrar la dirección actual cuando se presiona una tecla.
2. Borrar la dirección cuando se suelta la tecla.
3. En el método `on_update`, si no está moviéndose y hay una dirección activa, iniciar el movimiento automáticamente.

como podría hacer que entre clima y clima haya una transicion más suave?

Para lograr una transición más suave entre climas, puedes interpolar gradualmente los valores de clima anterior a clima nuevo durante algunos segundos, en vez de cambiar todo de golpe. Algunas ideas:

1. **Transición de intensidad y multiplicador:**
 - Guarda el clima anterior y el nuevo.
 - Durante, por ejemplo, 2 segundos, interpola (mezcla) intensidad y multiplicador de velocidad del anterior al nuevo usando una variable de progreso (de 0 a 1).
 - Aplica el valor interpolado en cada frame.
2. **Implementación básica:**
 - Al cambiar de clima, guarda los valores actuales y los nuevos, y activa un "modo transición" con un temporizador.
 - En `on_update`, si está en transición, calcula el valor interpolado según el tiempo transcurrido.
 - Al terminar la transición, usa solo el nuevo clima.

haz que al guardarse en la lista, se guarden los puntajes en un archivo .json llamado puntajes.json dentro de la carpeta data

MapaPrueba.py Marcador.py

Marcador.py +12 -0

```
self.guardar_puntajes_json()

def guardar_puntajes_json(self):
    import os, json
    ruta = os.path.join('data', 'puntajes.json')
    os.makedirs('data', exist_ok=True)
    lista_dict = [
        {'nombre': p.nombre, 'score': p.score}
        for p in self.puntajes
    ]
    with open(ruta, 'w', encoding='utf-8') as f:
        json.dump(lista_dict, f, ensure_ascii=False, indent=2)
```

haz que el sprite del personaje se rote en espejo si va hacia la izquierda

MapaPrueba.py +6 -1

```
elif key == arcade.key.LEFT:
    new_col -= 1
    self.player_sprite.scale_x = -abs(self.player_sprite.scale_x) # Espejo a la izquierda
elif key == arcade.key.RIGHT:
    new_col += 1
    self.player_sprite.scale_x = abs(self.player_sprite.scale_x) # Normal a la derecha
```

puedes poner un pequeño cartel abajo a la derecha con el clima, el tiempo restante y la intensidad?

MapaPrueba.py +18 -0

```
def draw_clima_info(self):
    # Texto del clima
    texto = f"Clima: {self.clima.condicion}\n"
    texto += f"Intensidad: {self.clima.intensidad:.2f}\n"
    texto += f"Tiempo restante: {int(self.clima.tiempo_restante)}s"
    ancho = 180
    alto = 70
    x = self.window_width - ancho - 10
    y = 10
    arcade.draw_lwrh_rectangle_filled(x, y, ancho, alto, arcade.color.LIGHT_GRAY)
    arcade.draw_lwrh_rectangle_outline(x, y, ancho, alto, arcade.color.DARK_GRAY, 2)
    arcade.draw_text(
        texto,
        x + 10, y + alto - 10,
        arcade.color.BLACK, 13,
        anchor_x="left", anchor_y="top"
    )
```

puedes hacer un metodo que al acabar el tiempo del juego, guarde el puntaje en base al dinero conseguido * la reputación

MapaPrueba.py Puntaje.py Marcador.py

Marcador.py +11 -0

Listo. Ahora el método `guardar_puntaje_final` en la clase `Marcador` guarda el puntaje al acabar el juego usando `ingresos * reputación`. Si quieres que también lo guarde en un archivo, avísame.

👍 🗨️ GPT-4.1 · 0x

```
def guardar_puntaje_final(self, nombre, ingresos, reputacion):
    """
    Guarda el puntaje final al acabar el juego.
    Puntaje = ingresos * reputacion
    """
    score = ingresos * reputacion
    nuevo_puntaje = Puntaje(score, nombre)
    self.agregar_puntaje(nuevo_puntaje)
    # Aquí podrías guardar en archivo si lo deseas
    return nuevo_puntaje
```

```
self.guardar_puntajes_json()

def guardar_puntajes_json(self):
    import os, json
    ruta = os.path.join('data', 'puntajes.json')
    os.makedirs('data', exist_ok=True)
    lista_dict = [
        {'nombre': p.nombre, 'score': p.score}
        for p in self.puntajes
    ]
    with open(ruta, 'w', encoding='utf-8') as f:
        json.dump(lista_dict, f, ensure_ascii=False, indent=2)
```

```
def guardar_en_slot(slot, slotInt):
    """Guarda el estado actual del juego en binario usando pickle."""
    import pickle, os
    ruta = os.path.join('saves', f'slot{slot}.bin')
    estado = {
        'total_time': self.total_time,
        'player': {
            'row': self.player_sprite.row,
            'col': self.player_sprite.col,
            'center_x': self.player_sprite.center_x,
            'center_y': self.player_sprite.center_y,
            'resistencia': self.player_sprite.get_resistencia_actual(),
            'nombre': getattr(self.player_sprite, 'nombre', ''),
            'ingresos': getattr(self.player_sprite, 'ingresos', 0),
            'reputacion': getattr(self.player_sprite, 'reputacion', 1),
            'peso_total': getattr(self.player_sprite, 'peso_total', 0),
            # Inventario: lista de IDs de pedidos
            'inventario': [modo_pedido.id for modo in self.itear_inventario()]
        },
        'clima': {
            'condicion': self.clima.condicion,
            'intensidad': self.clima.intensidad,
            'tiempoRestante': self.clima.tiempoRestante,
            'multiplicadorVelocidad': self.clima.multiplicadorVelocidad,
        },
        'pedidos_activos': list(self.pedidos_activos.keys()),
        'pedidos_pendientes': [p.id for p in self.pedidos_pendientes],
        'estado_actual': self.pedido_actual.id if self.pedido_actual else None,
    }
    with open(ruta, 'wb') as f:
        pickle.dump(estado, f)
    print(f"Guardado en {ruta}")
```

[illegible]

haz que si la carga del API falla,
use el backup, porfavor

Agrega manejo de errores para que si falla la carga del API, use los datos de `data/backup.json`.

```

-- city/map --
try:
    city_data = requests.get(f"{BASE_URL}/city/map", timeout=5).json()
except Exception as e:
    print(f"[WARN] city/map API failed: {e}")
    backup = cargar_backup()
    if backup and "city_map" in backup:
        city_data = backup["city_map"]
    else:
        raise RuntimeError("No se pudo obtener city/map ni backup.json")

```

[illegible]

```

imgur y mostrar putajes[sal]:
import os, json
ruta = os.path.join("data", "putajes.json")
if os.path.exists(ruta):
    with open(ruta, "r", encoding="utf-8") as f:
        try:
            putajes = json.load(f)
        except Exception:
            putajes = []
else:
    putajes = []
# Ordena los datos de menor a mayor
putajes.sort(key=lambda p: p["score"], reverse=True)
stil.putaj.putajes = putajes
stil.score.putaj.putajes = None
# Se genera el canvas de la imagen
drag.pup.putaj.putaj(sf):
    ancho, alto = 400, 300
    x = stil.window.width // 2 - ancho // 2
    y = stil.window.height // 2 - alto // 2
    arcade.draw_3ba_rectangle_filled(x, y, ancho, alto, arcade.color.DARK_GRAY)
    arcade.draw_text("rectangulo_texto", x, y, ancho, alto, arcade.color.WHITE, 20, anchor="left", anchor_y="top")
    arcade.draw_text("rectangulo_numero", x + 30, y + 10, 20, anchor="left", anchor_y="top")
    for i, p in enumerate(stil.putaj.putajes[10]):
        nombre = p["score"]
        score = p["score"]
        arcade.draw_text(f"[{i+1}, {nombre}, {score}]", x + 30, y + alto - 10 - i * 20, arcade.color.YELLOW, 16, anchor="left", anchor_y="top")
    else:
        arcade.draw_text("No hay putajes guardados", x + 30, y + alto - 60, arcade.color.LIGHT_GRAY, 16, anchor="left", anchor_y="top")

```

```
def save_backup():
    """Guarda BACKUP_DATA en data/backup.json."""
    try:
        os.makedirs("data", exist_ok=True)
        ruta = os.path.join("data", "backup.json")
        with open(ruta, "w", encoding="utf-8") as f:
            json.dump(BACKUP_DATA, f, ensure_ascii=False, indent=2)
        # print("[DEBUG] Backup actualizado en (ruta)")
    except Exception as e:
        print(f"[WARN] No se pudo escribir backup.json: {e}")
```

```
print(f"[WARN] No se pudo escribir backup.json: {e}")
```