Alejandro Carvajal (408798)

Entregables come cocos

1. Código fuente del juego
   Corrección de errores
   Error 1: el juego no reiniciaba correctamente las bolitas después de ganar
   Solución: Se modificó la función para que, al reiniciar, todas las casillas vacías (0) vuelvan a tener bolitas (2).

```
def reiniciar_juego():
    global pac_x, pac_y, fan_x, fan_y, puntos
    pac_x, pac_y = 1, 1
    fan_x, fan_y = 10, 5
    puntos = 0
    for fila in range(len(mapa)):
        for columna in range(len(mapa[fila])):
            if mapa[fila][columna] == 0:  # Si es un espacio vacío, se repone una bolita
                mapa[fila][columna] = 2
```

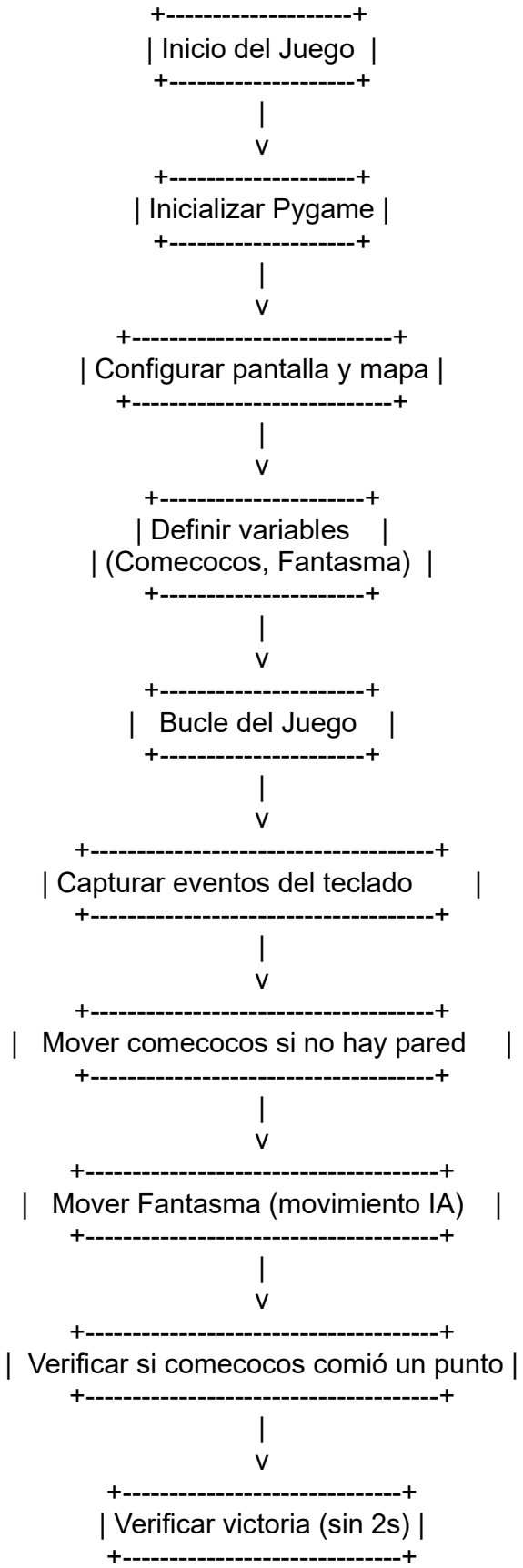
2. Cuando el comecos intentaba moverse más allá de los bordes del mapa, el código intentaba acceder a índices fuera de rango en la lista mapa, lo que causaba un error.
   Solución: Se agregó una verificación en mover_comecocos() para evitar que se salga de los límites

```
: if 0 <= nuevo_x < MAPA_ANCHO and 0 <= nuevo_y < MAPA_ALTO and mapa[nuevo_y][nuevo_x] != 1:
   pac_x, pac_y = nuevo_x, nuevo_y
```

   Esta cambio se verifica que:
   nuevo_x y nuevo_y estén dentro del rango del mapa y que no se pueda mover a una celda que contenga 1 (pared).

3. error: Antes de corregir el código, si verificar_victoria() encontraba que todas las bolitas habían sido comidas, llamaba varias veces a messagebox.showinfo(), lo que hacía que el mensaje de victoria apareciera repetidamente.
   Solución: se retorna False en verificar_victoria() si aún hay bolitas, y solo se ejecuta messagebox.showinfo() una vez cuando todas las bolitas han desaparecido.

```
def verificar_victoria():
    for fila in mapa:
        if 2 in fila:
            return False  # Si quedan bolitas, el juego continúa
    messagebox.showinfo("¡Ganaste!", "You are a winner!")  # Mensaje de victoria
    reiniciar_juego()  # Reiniciar el juego
```


Inicio del Juego
Inicializar Pygame
Configurar pantalla y mapa
Definir variables
(Comecocos, Fantasma)
Bucle del Juego
Capturar eventos del teclado
Mover comecocos si no hay pared
Mover Fantasma (movimiento IA)
Verificar si comecocos comió un punto
Verificar victoria (sin 2s)

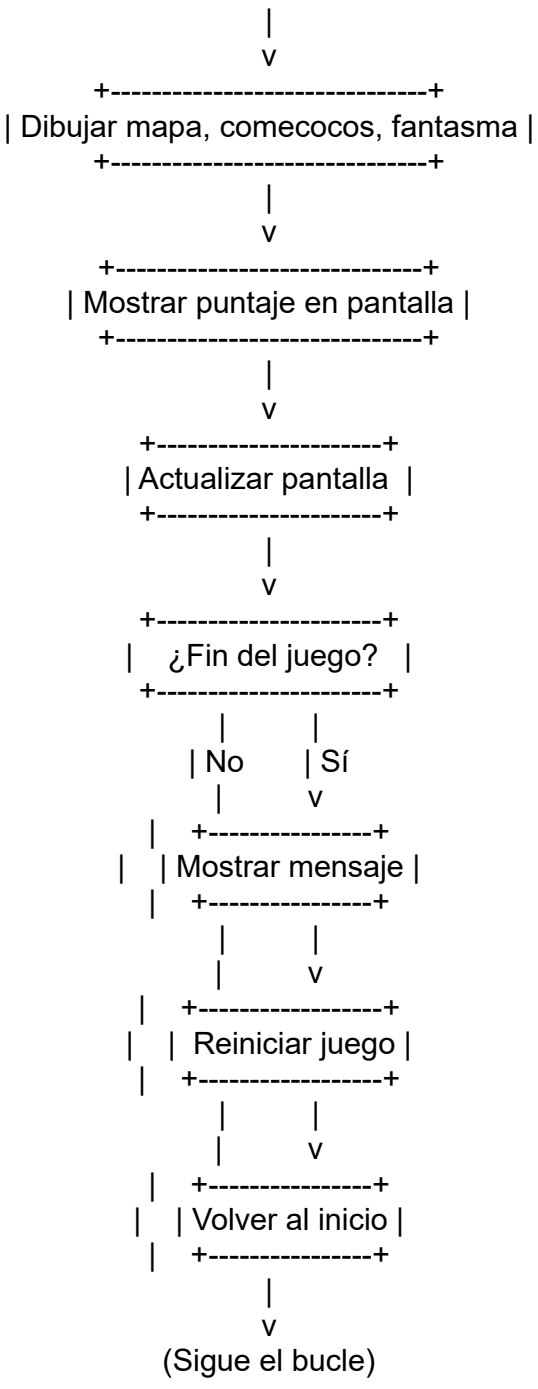

Dibujar mapa, comecocos, fantasma
Mostrar puntaje en pantalla
Actualizar pantalla
¿Fin del juego?
No
Sí
Mostrar mensaje
Reiniciar juego
Volver al inicio
(Sigue el bucle)

Diagrama de flujo

Explicación de las funciones

```
pygame.init()
pantalla = pygame.display.set_mode((MAPA_ANCHO * TAMANO_CELDA,
MAPA_ALTO * TAMANO_CELDA))
pygame.display.set_caption("Comecocos")
```

Se inicia **Pygame** y se configura la ventana del juego y se establece el tamaño y el título de la pantalla.

```
def verificar_victoria():
    for fila in mapa:
        if 2 in fila:
            return False
    messagebox.showinfo("¡Ganaste!", "You are a winner!")
    reiniciar_juego()
```

Recorre el mapa buscando si quedan bolitas (valor 2).
Si no hay más bolitas, muestra un mensaje de victoria y reinicia el juego.

```
def mover_comecocos(dx, dy):
    global pac_x, pac_y, puntos
    nuevo_x = pac_x + dx
    nuevo_y = pac_y + dy
    if mapa[nuevo_y][nuevo_x] != 1:  # Verifica que no haya una pared
        pac_x, pac_y = nuevo_x, nuevo_y
        if mapa[pac_y][pac_x] == 2:  # Si hay una bolita
            puntos += 10
            mapa[pac_y][pac_x] = 0  # Elimina la bolita
            verificar_victoria()  # Verifica si ha ganado
```

Calcula la nueva posición de Comecocos. Si la celda no es una pared, permite el movimiento. Si hay una bolita, suma puntos y la elimina.
Luego verifica si todas las bolitas fueron comidas.

```
def mover_fantasma():
    global fan_x, fan_y
    direcciones = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    random.shuffle(direcciones)  # Mezcla las direcciones para que el
movimiento sea aleatorio
    for dx, dy in direcciones:
        nuevo_x = fan_x + dx
        nuevo_y = fan_y + dy
```

```
            if mapa[nuevo_y][nuevo_x] != 1:  # Si no es una pared
                fan_x, fan_y = nuevo_x, nuevo_y
                break
```

Se generan movimientos aleatorios. Se verifica que el fantasma no atraviese paredes

```
        if pac_x == fan_x and pac_y == fan_y:
            mostrar_game_over()
```

Si la posición de Comecocos coincide con la del fantasma, se muestra el mensaje "Game Over" y el juego se reinicia.

Dificultades Encontradas y Cómo Fueron Superadas

Durante el desarrollo del juego, enfrenté varios desafíos, entre ellos:

Manejo de colisiones y detección de victoria

Dificultad: Inicialmente, el juego no detectaba correctamente cuándo Comecocos había comido todas las bolitas.

Solución: Implementé la función verificar_victoria(), que revisa el mapa en busca de bolitas restantes después de cada movimiento de Comecocos.

Movimiento del fantasma

Dificultad: El fantasma a veces se quedaba atascado sin moverse.

Solución: Agregué random.shuffle(direcciones), permitiendo que el fantasma intente moverse en todas las direcciones antes de quedarse quieto.

Control del juego con el teclado

Dificultad: Al principio, Comecocos no respondía bien a las teclas.

Solución: Usé pygame.key.get_pressed() en lugar de pygame.event.get(), asegurando una detección fluida del teclado.

Aspectos que Mejoraría en el Código con Más Tiempo o Experiencia

Si tuviera más tiempo o experiencia en desarrollo de videojuegos, mejoraría lo siguiente:

Organización del Código

Actualmente, todo el código está en un solo archivo.

Sería mejor separarlo en módulos como:

jugador.py → Para manejar a Comecocos.

fantasma.py → Para la IA del fantasma.

mapa.py → Para la lógica del nivel.

IA del Fantasma

El movimiento del fantasma es aleatorio.

Podría implementarse una IA más avanzada, como el algoritmo *A o BFS**, para que el fantasma persiga a Comecocos de manera inteligente.

Mejoras en la Detección de Colisiones

Actualmente, Comecocos y el fantasma ocupan una sola celda, lo que hace que las colisiones sean simples.

Si el juego tuviera sprites más grandes, necesitaría detección de colisiones más precisa con pygame.Rect.

Posibles Mejoras al Juego

Si quisiera expandir el juego en el futuro, estas son algunas ideas:
3.1. Agregar Niveles
Después de completar un nivel, podría generarse un nuevo mapa con mayor dificultad.
Se podrían incluir laberintos más complejos con más obstáculos.
Nuevos Enemigos con Diferente IA
Agregar más fantasmas, cada uno con una estrategia diferente:
Uno que siga a Comecocos directamente.

Otro que intente interceptarlo en las esquinas.

Mejoras Visuales y Sonido

Usar sprites animados en lugar de cuadrados.

Incluir efectos de sonido cuando Comecocos come una bolita o es atrapado.

Agregar una pantalla de inicio y de Game Over más atractivas.

Power-Ups y Modos de Juego

Un power-up que haga que Comecocos pueda comer a los fantasmas por unos segundos.

Un modo "infinito" donde el objetivo sea sobrevivir el mayor tiempo posible.

prueba	movimiento	Resultado que se espera	Resultado obtenido	
Movimiento de Comecocos	Presionar la tecla de dirección	comecocos se mueve sin atravesar paredes	Comecocos se mueve correctamente	Aprobado
Comer un punto	Mover comecocos sobre un punto	El punto desaparece y la puntuación aumenta	Funciona correctamente	Aprobado
Detección de colisión con fantasma	Mover comecocos hacia el fantasma	El juego muestra mensaje de derrota	No muestra mensaje	Falló

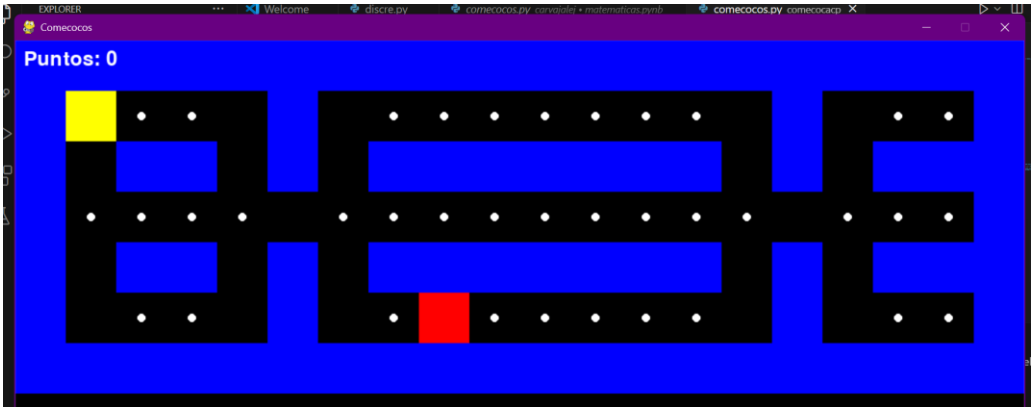
Registro de Errores y Soluciones

Error 1: Comecocos no colisiona correctamente con las paredes

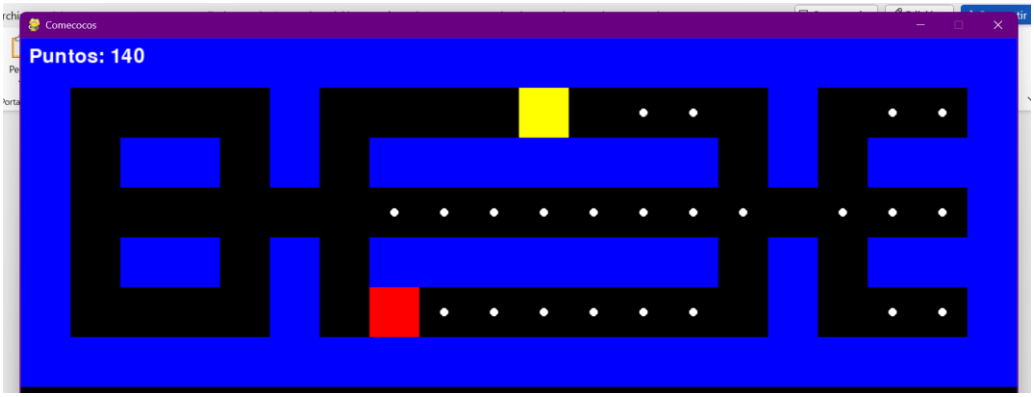
- Descripción: Comecocos atravesaba las paredes al moverse.
- Causa: La función mover_comecocos(dx, dy) no verificaba correctamente si la nueva posición era una pared.
- Solución: Se agregó la validación if mapa[nuevo_y][nuevo_x] != 1: antes de actualizar la posición de Comecocos.

Evidencia visual de su funcionamiento

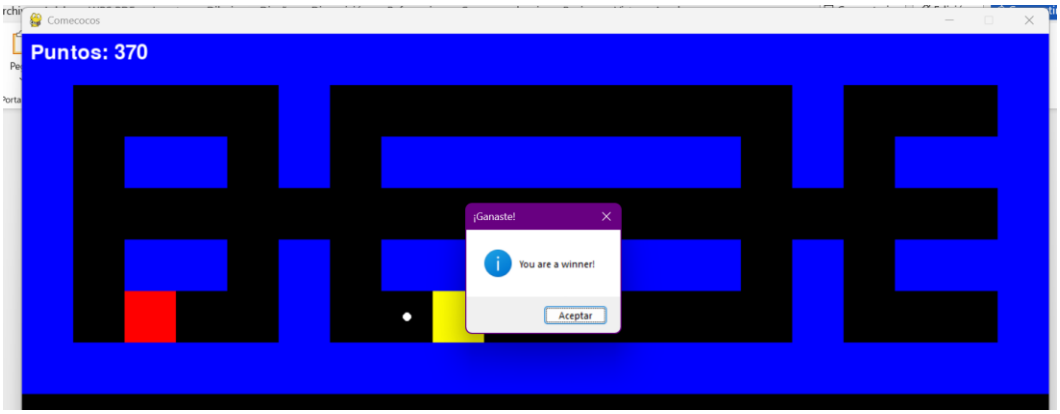
Inicio



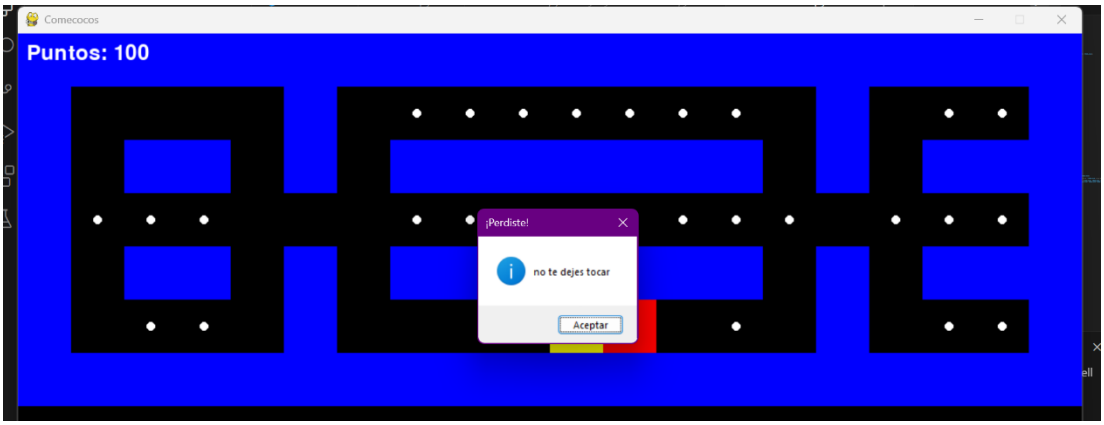
Come y genera los puntos correctamente



Muestra “ganador” cuando come todas las bolas



En caso de ser atrapado aparece el aviso y se reinicia el juego



Conclusión

Desarrollar este juego fue un excelente ejercicio de lógica y programación en Python. Aprendí sobre manejo de eventos, colisiones, IA básica y la estructura de un videojuego. Con más tiempo y práctica, podría hacer mejoras significativas para crear una versión más completa y desafiante.