**David Santiago Martínez**

**Miguel Briceño**

**Christopher Pinzón**

**Documentación del Juego de Pac-Man**

**1. Introducción**

Este documento describe un juego de Pac-Man desarrollado en Python utilizando la biblioteca Pygame para la interfaz gráfica y Tkinter para el manejo de ventanas emergentes. El juego implementa las mecánicas básicas de Pac-Man: el jugador controla un personaje que debe recolectar todos los puntos del laberinto mientras evita ser capturado por un fantasma.

**2. Estructura del Código**

**2.1 Módulos Importados**

* **pygame**: Gestiona la ventana del juego, los eventos del teclado y el renderizado gráfico.
* **random**: Genera movimientos aleatorios para el fantasma.
* **sys**: Controla la finalización del programa.
* **tkinter**: Muestra cuadros de diálogo al usuario.

**2.2 Constantes del Juego**

* **Colores**: NEGRO (0,0,0), BLANCO (255,255,255), AZUL (0,0,255), AMARILLO (255,255,0), ROJO (255,0,0).
* **TAMANO\_CELDA**: Define el tamaño de cada celda del mapa (40 píxeles).

**2.3 Variables Globales**

* **mapa\_original**: Matriz que define el estado inicial del laberinto.
* **mapa**: Copia de trabajo del mapa original que se modifica durante el juego.
* **puntos**: Contador de puntos recolectados por el jugador.
* **puntos\_totales**: Número total de puntos disponibles en el mapa.
* **pac\_x, pac\_y**: Coordenadas actuales de Pac-Man.
* **fantasma\_x, fantasma\_y**: Coordenadas actuales del fantasma.
* **posicion\_inicial\_pacman**: Coordenadas iniciales de Pac-Man.

**3. Sistema de Representación del Mapa**

El laberinto se representa como una matriz donde cada celda contiene uno de los siguientes caracteres:

* **'#'**: Pared (obstáculo)
* **'.'**: Punto (comestible)
* **'P'**: Posición de Pac-Man
* **'G'**: Posición del fantasma
* **' '**: Espacio vacío (ya visitado)

**4. Funciones Principales**

**4.1 Inicialización**

* **Inicialización de Pygame**: Configuración inicial de la biblioteca.
* **Cálculo de dimensiones**: El tamaño de la ventana se determina a partir del mapa.
* **Búsqueda de posiciones iniciales**: Se localizan las coordenadas iniciales de Pac-Man y el fantasma.

**4.2 Funciones de Renderizado**

def dibujar\_mapa():

* **Propósito**: Muestra el estado actual del juego en la pantalla.
* **Funcionamiento**: Recorre la matriz del mapa, dibujando cada elemento según su tipo (paredes, puntos, Pac-Man, fantasma).
* **Elementos adicionales**: Muestra el contador de puntos en la esquina superior izquierda.

**4.3 Control del Juego**

def reiniciar\_juego():

* **Propósito**: Restablece el estado del juego a sus valores iniciales.
* **Acciones**: Reposiciona a Pac-Man y al fantasma, reinicia el contador de puntos y reconstruye el mapa.

def mover\_pacman(dx, dy):

* **Propósito**: Gestiona el movimiento de Pac-Man según la entrada del usuario.
* **Parámetros**: dx y dy indican la dirección del movimiento.
* **Validaciones**: Comprueba si el movimiento es válido (no atraviesa paredes).
* **Acciones**: Actualiza la posición de Pac-Man, recolecta puntos si los hay y verifica colisiones con el fantasma.
* **Condiciones de victoria**: Detecta cuando se han recolectado todos los puntos.

def mover\_fantasma():

* **Propósito**: Controla el movimiento aleatorio del fantasma.
* **Algoritmo**: Selecciona aleatoriamente una de las cuatro direcciones posibles.
* **Validaciones**: Verifica que el movimiento sea válido (no atraviesa paredes).
* **Gestión de estados**: Preserva el estado de la celda que ocupaba el fantasma.
* **Detección de colisiones**: Verifica si el fantasma ha capturado a Pac-Man.

**4.4 Estados del Juego**

def mostrar\_game\_over():

* **Propósito**: Muestra la pantalla de fin de juego cuando Pac-Man es capturado.
* **Interfaz**: Renderiza el mensaje "Game Over" y espera 2 segundos.
* **Acción posterior**: Llama a la función para preguntar si se desea reiniciar.

def mostrar\_victoria():

* **Propósito**: Muestra la pantalla de victoria cuando se recolectan todos los puntos.
* **Interfaz**: Renderiza el mensaje "¡Victoria!" y espera 2 segundos.
* **Acción posterior**: Llama a la función para preguntar si se desea reiniciar.

def preguntar\_volver\_a\_jugar(mensaje):

* **Propósito**: Consulta al usuario si desea reiniciar el juego o salir.
* **Parámetro**: mensaje es el texto que se muestra en el cuadro de diálogo.
* **Implementación**: Utiliza Tkinter para crear una ventana emergente con opciones Sí/No.
* **Acciones**: Si el usuario elige "Sí", reinicia el juego; si elige "No", cierra la aplicación.

**4.5 Bucle Principal**

def main():

* **Propósito**: Constituye el bucle principal del juego.
* **Control de eventos**: Captura y procesa los eventos de teclado y cierre de ventana.
* **Temporizador**: Controla la frecuencia de movimiento del fantasma.
* **Renderizado**: Actualiza la pantalla en cada iteración.
* **Control de FPS**: Limita la velocidad del juego a 30 fotogramas por segundo.

**5. Flujo del Juego**

**5.1 Inicialización**

1. Se carga el mapa inicial.
2. Se localizan las posiciones de Pac-Man y el fantasma.
3. Se calcula el número total de puntos disponibles.
4. Se configura la ventana del juego.

**5.2 Bucle de Juego**

1. Se procesan los eventos de entrada del usuario (teclas de dirección).
2. Se actualiza la posición de Pac-Man según la entrada.
3. Periódicamente, se actualiza la posición del fantasma.
4. Se verifica si hay colisión entre Pac-Man y el fantasma.
5. Se comprueba si se han recolectado todos los puntos.
6. Se actualiza la pantalla.
7. Se repite el bucle hasta que el juego termine o el usuario salga.

**5.3 Finalización**

* Si Pac-Man es capturado: Se muestra la pantalla de "Game Over".
* Si se recolectan todos los puntos: Se muestra la pantalla de "¡Victoria!".
* En ambos casos: Se pregunta al usuario si desea reiniciar o salir.

**6. Interacción del Usuario**

* **Teclas de dirección**: Controlan el movimiento de Pac-Man (arriba, abajo, izquierda, derecha).
* **Tecla Escape**: Cierra el juego.
* **Cuadros de diálogo**: Permiten decidir si reiniciar o salir del juego.

**7. Estructura de Datos Principales**

* **Matriz del mapa**: Representación bidimensional del laberinto y sus elementos.
* **Coordenadas**: Pares de valores (x,y) que representan posiciones en el mapa.
* **Contadores**: Variables que rastrean el progreso (puntos recolectados).

**8. Posibles Mejoras**

* Implementación de múltiples niveles con diferentes mapas.
* Adición de más fantasmas con comportamientos distintos.
* Inclusión de power-ups que permitan a Pac-Man capturar fantasmas.
* Mejora de los gráficos con sprites más elaborados.
* Implementación de un sistema de puntuación más complejo.
* Adición de sonidos y efectos de audio.

**9. Observaciones Técnicas**

* El juego utiliza un sistema de coordenadas basado en la matriz del mapa, no en píxeles directos.
* La lógica de colisión se basa en la comparación de coordenadas, no en detección de píxeles.
* El movimiento del fantasma es completamente aleatorio, sin algoritmos de persecución.
* La interfaz gráfica es simple, utilizando formas básicas de Pygame.