ALGORITMOS PROYECTO FINAL

ARCHIVO: tres_en_raya_supremo.py

Clase TableroPequeño.

- **Algoritmo de Verificación de Victoria:**

  Se utiliza un recorrido lineal sobre filas, columnas y diagonales para verificar si un jugador ha ganado.

- Justificación de la Elección:

  La simplicidad y la baja complejidad hacen que este algoritmo sea ideal.

- **Algoritmo para Determinar si el Tablero está Lleno:**

  Itera sobre todas las casillas del tablero para verificar si están ocupadas.

- Justificación:

  Es eficiente en términos de rendimiento y claridad.

Clase TableroGrande

- **Algoritmo de Verificación de Victoria:**

  Similar al TableroPequeño, pero operando sobre los ganadores de tableros pequeños en lugar de fichas individuales.

- Justificación:

  Este enfoque asegura claridad y rendimiento al trabajar sobre un diseño de tablero jerárquico.

Clase Juego

- **Algoritmo para Manejar Restricciones**:

  Al seleccionar la casilla pequeña, el jugador debe jugar en el tablero correspondiente. Esto se implementa verificando coordenadas y disponibilidad.

- Justificación:

  Este diseño asegura que las reglas se cumplan sin afectar el rendimiento.

ARCHIVO: interfaz_grafica.py

Dibujar el Tablero y las Fichas (dibujar_tablero)

- Algoritmo:

  Recorre cada tablero pequeño y sus casillas para renderizar los gráficos y las fichas.

- Justificación:

  El enfoque iterativo es claro y suficientemente rápido para un juego con un diseño visual limitado.

Animar las Casillas Ganadoras (animar_casillas_grandes_ganadoras)

- Algoritmo:

  Parpadeo basado en un bucle corto con cambios de color.

- Justificación:

  La animación se mantiene simple y rápida para no afectar la experiencia del usuario.

Manejar el Clic del Jugador (manejar_clic)

- Algoritmo:

  Convierte coordenadas de clic en posiciones del tablero, respetando restricciones.

- Justificación:

  La conversión directa asegura que la interacción sea precisa y rápida.

Menú Post-Partida (menu_post_partida)

- Algoritmo:

  Usa detección de colisión para botones interactivos.

- Justificación:

  Los botones simplifican la interacción del usuario sin penalizar el rendimiento.

ARCHIVO: menu.py

Dibujar el Menú Principal (menu_principal)

- Algoritmo:

  Usa rectángulos para representar botones y detecta clics mediante colisiones.

- Justificación:

  Este diseño es estándar y eficiente para interfaces gráficas simples.

Personalización de Colores (personalizar_juego)

- Algoritmo:

  Itera entre opciones de colores usando índices y confirma la selección.

- Justificación:

  Proporciona una forma simple y funcional de personalización sin afectar la jugabilidad.

Selección de Colores (cambiar_color)

- Algoritmo:

  Cicla entre opciones de colores predefinidos y confirma con la tecla Enter.

- Justificación:

  Simplicidad y claridad en la implementación.