

**Instituto Politecnico Nacional**

**Escuela superior de computo**

**Oscar Humberto Mayoleytte Paredes**

**Armando Jimenez Galvan**

**Materia: Algebra Lineal**

**Profesor: David Correa Coyac**

**Grupo: 2BM2**

**Proyecto: Visualizador de Transformaciones Geométricas en 2D**

**Documentación detallada para stream\_reader.pyPropósito del archivo**

**El código fuente implementa una aplicación gráfica que permite cargar y procesar transformaciones geométricas en 2D desde archivos JSON. Los usuarios pueden visualizar los resultados en gráficos y aplicar transformaciones adicionales como rotaciones, escalas, traslaciones y reflexiones.El objetivo principal es ofrecer una herramienta interactiva para analizar figuras geométricas y sus transformaciones.**---**Código explicado1. Importación de bibliotecas**pythonimport json # Para manejar archivos JSONimport numpy as np # Biblioteca para cálculos matemáticos y manejo de arreglosimport tkinter as tk # Biblioteca para crear interfaces gráficasfrom tkinter import filedialog, ttk # Widgets avanzados y diálogos para seleccionar archivosimport matplotlib.pyplot as plt # Biblioteca para crear gráficos y visualizacionesfrom datetime import datetime # Biblioteca para manejar fechas y horasjson: Carga y procesa datos estructurados en formato JSON.numpy: Realiza cálculos numéricos avanzados como multiplicación de matrices.tkinter y ttk: Proveen herramientas para la creación de interfaces gráficas.filedialog: Permite al usuario seleccionar archivos desde su sistema.matplotlib.pyplot: Se usa para visualizar las figuras geométricas y sus transformaciones.datetime: Ayuda a generar nombres únicos para guardar gráficos con marcas de tiempo.**2. Clase principal: TransformationApp**

Definición y método de inicialización

pythonclass TransformationApp: def \_\_init\_\_(self, root): self.root = root self.root.title("Transformaciones desde Archivo") self.result\_dict = {} # Diccionario para almacenar transformaciones aplicadas self.max\_value = 1 # Límite inicial de los ejes en las gráficasself.result\_dict: Almacena las transformaciones realizadas y sus resultados.self.max\_value: Define los límites de las gráficas para visualizar adecuadamente las figuras transformadas.**Configuración de la interfaz gráfica**pythonBotones principalesttk.Button(root, text="Cargar Archivo", command=self.load\_file).pack(pady=5)ttk.Label(root, text="(Carga un archivo JSON con vértices y configuraciones)").pack(anchor="w")ttk.Button(root, text="Aplicar Rotación", command=self.apply\_rotation).pack(pady=5)ttk.Label(root, text="(Aplica rotación según el ángulo especificado)").pack(anchor="w")ttk.Button(root, text="Aplicar Escala", command=self.apply\_scale).pack(pady=5)ttk.Label(root, text="(Escala los vértices según los valores proporcionados)").pack(anchor="w")ttk.Button(root, text="Aplicar Traslación", command=self.apply\_translation).pack(pady=5)ttk.Label(root, text="(Traslada los vértices según los valores especificados)").pack(anchor="w")ttk.Button(root, text="Aplicar Reflexión", command=self.apply\_reflection).pack(pady=5)ttk.Label(root, text="(Refleja la figura en los ejes X o Y)").pack(anchor="w")ttk.Button(root, text="Graficar Resultados", command=self.plot\_results).pack(pady=5)ttk.Label(root, text="(Genera una gráfica con los resultados de las transformaciones)").pack(anchor="w")ttk.Button(root, text="Guardar Gráfica", command=self.save\_graphic).pack(pady=5)ttk.Label(root, text="(Guarda la gráfica como archivo PNG)").pack(anchor="w")**Cada botón está asociado a un método específico para ejecutar acciones como cargar archivos, aplicar transformaciones y graficar resultados.Contenedor para entradas de transformación**pythonself.transformation\_frame = ttk.Frame(root, padding=10)self.transformation\_frame.pack(pady=10)self.add\_transformation\_inputs()**Este marco contiene las entradas para las transformaciones geométricas: rotación, escala, traslación y reflexión.3. Métodos principales**Cargar archivo JSONpythondef load\_file(self): file\_path = filedialog.askopenfilename(filetypes=[("Archivos JSON", "\*.json")]) if file\_path: try: self.result\_dict, self.max\_value = self.load\_from\_file(file\_path) print(f"Archivo cargado: {file\_path}") except Exception as e: print(f"Error al cargar el archivo: {e}")**Permite al usuario seleccionar un archivo JSON desde el sistema.Los datos del archivo se procesan en el método load\_from\_file.**

**Transformaciones geométricas**python@staticmethoddef rotation(vertices, angle): return np.dot(vertices, [[np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)]])@staticmethoddef scale(vertices, sx, sy): return np.dot(vertices, [[sx, 0], [0, sy]])@staticmethoddef translation(vertices, tx, ty): return vertices + [tx, ty]@staticmethoddef reflection(vertices, axis): if axis == "horizontal": return np.dot(vertices, [[1, 0], [0, -1]]) # Reflexión horizontal elif axis == "vertical": return np.dot(vertices, [[-1, 0], [0, 1]]) # Reflexión vertical**rotation: Aplica una rotación utilizando matrices de rotación estándar.scale: Escala los vértices según los factores proporcionados.translation: Mueve los vértices a nuevas coordenadas.reflection: Refleja los vértices sobre el eje horizontal o vertical.Graficar resultados**pythondef plot\_results(self): if not self.result\_dict: print("No hay resultados para graficar.") return fig, ax = plt.subplots(figsize=(8, 8)) for key, data in self.result\_dict.items(): points = np.array(data["value"]) points = np.vstack([points, points[0]]) ax.fill(points[:, 0], points[:, 1], alpha=0.5, label=key.capitalize(), color=data["color"]) ax.plot(points[:, 0], points[:, 1], linestyle="--", color="black") ax.set\_aspect("equal") ax.legend() plt.show() **Las figuras se dibujan utilizando `matplotlib`. Las figuras originales y transformadas se diferencian por colores y etiquetas.4. Ejecución del programa1. Instalar dependencias:** bash pip install numpy matplotlib **2. Estructura del archivo JSON:** json { "points": [[0, 0], [5, 0], [5, 5], [0, 5]], "rotation": {"angle": 45}, "scale": {"value": [2, 2]}, "translation": {"value": [3, 3]}, "reflection": {"axis": "horizontal"} } **3. Iniciar el programa:** bash python stream\_reader.py **Resumen**

**El programa ofrece una herramienta flexible para explorar transformaciones geométricas en 2D. Además, gracias a su interfaz gráfica, resulta intuitivo para usuarios no técnicos. Las reflexiones añadidas amplían las capacidades del programa al incluir simetrías básicas.**