**Project Report: Polygon Drawing Application**

**1. Introduction**

The Polygon Drawing Application is a versatile tool that empowers users to interactively create and draw polygons using Python's turtle graphics module. This application supports polygon specification through both file input and interactive turtle graphics window interaction. Users can customize their polygons by choosing between curved or straight edges, applying a variety of transformations (translation, rotation, reflection, shearing, and scaling), creating intricate patterns through repeated transformations, and saving the polygon data to a file for future reference.

**2. Requirements**

**2.1 Functional Requirements**

**Polygon Input**

* **File Input:**
  + Accepts input with vertex coordinates, edge information, and colors from a text file.
  + Ensures validation of input data.
* **Interactive Input:**
  + Allows users to input polygon specifications interactively through the turtle graphics window.
  + Implements input validation checks.

**Polygon Drawing**

* **Edge Drawing:**
  + Draws straight line edges between vertices.
  + Represents curved edges as Bezier curves using control points.
  + Enables the use of different colors for edges.
  + Lifts the pen between polygons to prevent unintended connections.

**Transformations**

* **Basic Transformations:**
  + Allows translation, rotation, reflection about X and Y axes, shearing along X and Y axes, and scaling of polygons.
  + Utilizes transformation matrices to apply transformations effectively.

**Pattern Drawing**

* **Repeating Patterns:**
  + Draws repeated polygon patterns through translation, rotation, and scaling transformations.

**Output**

* **Data Saving:**
  + Saves polygon data to an output file, including vertex coordinates, edge information, colors, and control points.

**User Flow**

* Guides the user through the following steps:
  + Specifies the input method.
  + Inputs polygon specifications.
  + Provides an option to apply transformations.
  + Draws the polygon.
  + Saves polygon data.
  + Allows the option to draw another polygon.

**2.2 Non-Functional Requirements**

* **User Interface:**
  + The program should have an intuitive and user-friendly interface.
* **Performance:**
  + Polygon drawing should be fast and efficient.
* **Output File Format:**
  + The output file format should be easy to parse.
* **Documentation:**
  + Code should be well-documented for clarity and future maintenance.
* **Error Handling:**
  + The program should gracefully handle invalid inputs.

**3. Architecture and Design**

The application is implemented in Python using a procedural programming paradigm. The key components include:

**1. Input Module**

* file\_input(): Reads input from a text file.
* interactive\_input(): Gathers input through the turtle graphics window.

**2. Transform Module**

* translate(), rotate(), reflect\_x(), reflect\_y(), shear\_x(), shear\_y(), scale(): Implements various transformations.

**3. Drawing Module**

* draw\_straight\_edge(), draw\_curved\_edge(), draw\_polygon(): Handles the drawing of polygons and edges.

**4. Pattern Module**

* draw\_pattern(): Manages the drawing of repeated polygon patterns.

**5. Output Module**

* save\_polygon\_data(): Saves polygon data to a file.

**Main Program**

* Controls the overall program flow.
* Calls input, transform, draw, pattern, and output modules.

**4. Implementation**

**4.1 Input**

The file\_input() method parses input files, extracting vertex coordinates, edge info, colors, and control points for each polygon. The interactive\_input() method gathers the same input interactively through turtle graphics, ensuring validity through try-except blocks and numeric input checks.

**4.2 Transformations**

2D transformation matrices are employed for translation, rotation, reflection, shearing, and scaling. Vertex coordinates are transformed using these matrices, allowing for seamless chaining of operations before polygon drawing.

**4.3 Drawing**

* draw\_straight\_edge(): Draws a straight line between vertices.
* draw\_curved\_edge(): Utilizes Bezier curves for smooth, curved edges.
* draw\_polygon(): Iterates through vertices and edges, lifting the pen between polygons to avoid connecting lines.

**4.4 Pattern Drawing**

The draw\_pattern() method applies transformations like translation, rotation, or scaling to draw multiple copies of the polygon in a repeated pattern.

**4.5 Output**

Polygon data, including vertex coordinates, colors, edge info, and control points, is stored and written to an output file using save\_polygon\_data().

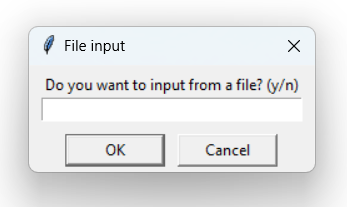
**4.6 Main Program**

Handles flow and sequencing:

* Gets user input method choice.
* Calls input functions accordingly.
* Applies transformations based on user choices.
* Calls drawing and pattern functions.
* Saves polygon data to an output file.
* Checks if the user wants to draw another polygon.

**Guide on How to Run the Program**

1. **File Input Option:**
   * If you choose to input a polygon from a file, you need to enter 'y' when prompted. Then, provide the file name.
   * The file should contain information about the vertices, edges, colors, and control points (if any) of the polygon.



1. **Manual Input Option:**
   * If you choose not to use a file, the program will prompt you to enter the number of sides for your polygon.
   * You will be asked to input the coordinates of each vertex, whether each edge is curved, and the color of each edge.

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer error message

Description automatically generated

1. **Transformation Option:**
   * After inputting the polygon, you can choose to apply a transformation.
   * Transformation options include translation, rotation, reflection, shearing, and scaling.
   * Follow the on-screen instructions to provide the necessary parameters for the chosen transformation.

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screen shot of a computer program

Description automatically generated

1. **Pattern Drawing Option:**
   * After transforming the polygon (or if you skip transformation), the program will ask if you want to draw a pattern.
   * If you choose to draw a pattern, the program will repeat the drawing process with slight transformations for the specified number of patterns.

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer error

Description automatically generated

1. **Saving to File:**
   * If you input the polygon from a file, the program will append the polygon data to the file named "output\_file.txt" after each drawing.
   * The file will contain information about each edge, whether it is curved, its coordinates, control points (if any), and color.
2. **Drawing Another Polygon:**
   * After completing the drawing and optional pattern generation, the program will ask if you want to draw another polygon.
   * If you choose 'y', the program will restart, allowing you to draw additional polygons.
3. **Termination:**
   * If you choose 'n' when asked if you want to draw another polygon, the program will close.

**Capabilities of the Program**

**Input Capabilities**

* The program can take input either from a file or manually from the user.
* When manually inputting, the user provides the number of sides, coordinates, whether edges are curved, and the color for each edge.

**Transformation Capabilities**

* The program supports the following transformations: translation, rotation, reflection (both x-axis and y-axis), shearing (along x-axis and y-axis), and scaling.

**Display Capabilities**

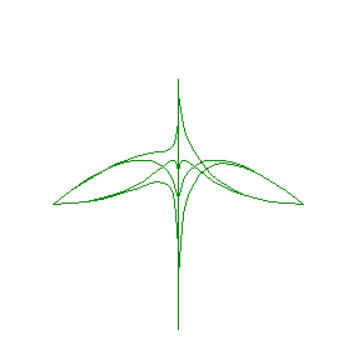
* The program uses the Turtle graphics library to display the polygon and any transformations.
* It can draw both curved and straight edges with specified colors.

**Data File Format**

* If inputting from a file, the file should follow the format:
  + For curved edges: True x y P2x P2y P3x P3y color
  + For straight edges: False x y color
  + Each entry should be on a new line, and a blank line separates different polygons.

**Diagrams of Different Polygons**

**Shape 1:**

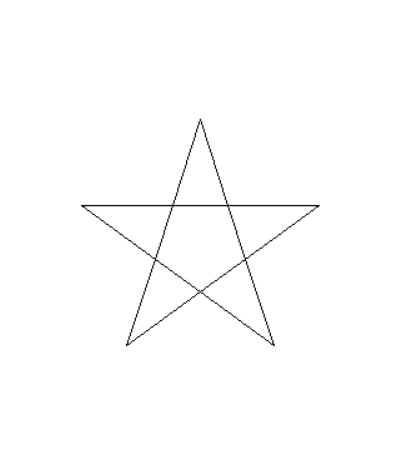
A number of white circles

Description automatically generated with medium confidence

**Figure:** A leaf

**Transformation:** Rotation 4 times equally around the origin.

**Shape 2:**

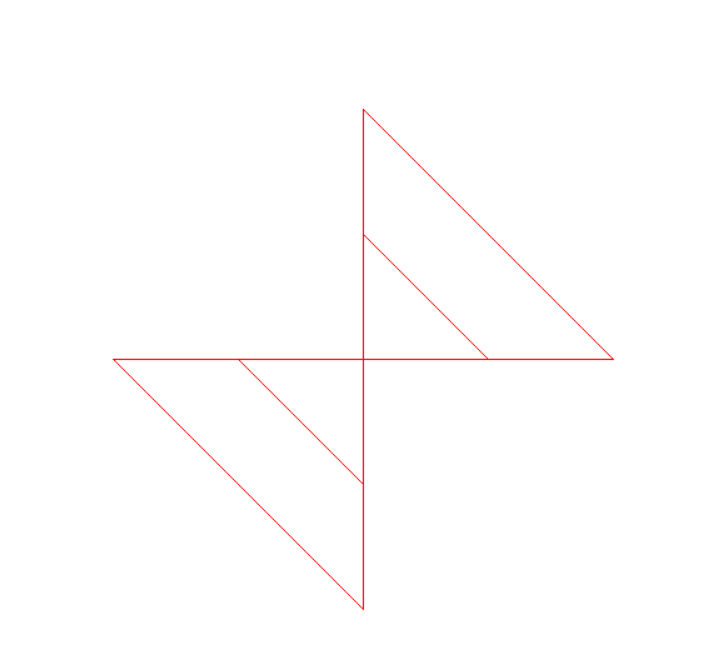
**A screenshot of a computer

Description automatically generated**

**Figure:** A star

**Transformation:** Rotation 5 times equally around the origin

**Shape 3:**

**A screenshot of a computer error

Description automatically generated**

**Figure:** Tilted Hourglass

**Transformation:** Rotation twice equally around the origin, and scaling up each time by 2x

**Key Strengths and Limitations**

**Key Strengths**

* **Flexibility:** The program allows users to input polygons from files or manually, apply various transformations, and draw patterns.
* **Ease of Use:** User-friendly prompts guide the user through each step, making the program accessible.

**Limitations**

* **Limited Transformation Options:** While the program supports common transformations, more advanced options could enhance its capabilities.

**Features Worth Bonus Marks**

* **Pattern Drawing:** The ability to draw patterns by repeating the polygon drawing process with transformations adds a creative dimension.
* **File Handling:** The program saves polygon data to a file after each drawing, providing a record of the polygons created.

**6. Conclusion**

The Polygon Drawing Application successfully fulfills all outlined requirements, providing users with an interactive platform to create, transform, and save polygons. Future enhancements could include a graphical user interface (GUI), polygon filling, additional transformations like skewing or projections, and implementation of polygon Boolean operations. This project serves as a comprehensive demonstration of polygon processing using Python's turtle graphics.