Introduction to Programming - Homework 4

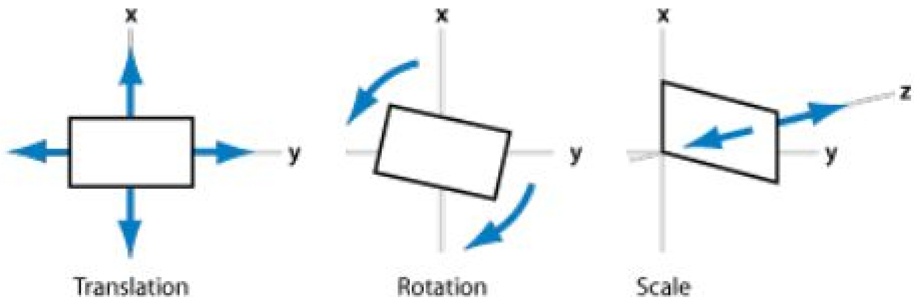
Transformations of 2-Dimensional Geometric Objects

A fundamental aspect of 'Computer Graphics', this concept is crucial in:

- Object modeling
- Object visualization
- Creating projections of objects

Objects

- Objects in a scene are a collection of points
- Objects have a **location, orientation,** and **size**
- Correspond to transformations:
    - **Translation**(T) ↔ location,
    - **Rotation** (R) ↔ orientation, and
    - **Scaling** (S) ↔ size



We can   transformations into two types:

- **Linear Transformations**
    - Can be represented as invertible (non-singular) matrices
    - In 2D, any linear transformation **T** can be represented by 2x2 matrices:

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

        - Where a, b, c, and d are the parameters of the transformation
    - A basic transform of a point (x,y) looks like:

$$Tp = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

        - Where (ax+by, cx+dy) is the resulting transformed point.
    - eg. Scaling, Rotation

- **Non- Linear Transformations**
    - In 2D, a non - linear transformation **cannot** be represented by 2x2 matrices
    - We use **Homogeneous coordinates** to represent such transformations

- Add an additional dimension, the w-axis, and an extra coordinate, the w-component.
- Effectively known as the hyperspace for embedding the 2D space
- In 2D, any non - linear transformation **T** can be represented by 3x3 matrices:

$$T = \begin{bmatrix} m_{11} & m_{12} & x_t \\ m_{21} & m_{22} & y_t \\ 0 & 0 & 1 \end{bmatrix}$$

- Where m_{ij}, x_t and y_t are the parameters of the transformation.
- A basic transform of a point (x,y) looks like:

$$Tp = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & x_t \\ m_{21} & m_{22} & y_t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}x + m_{12}y + x_t \\ m_{21}x + m_{22}y + y_t \\ 1 \end{bmatrix}$$

- Where (x', y') is the resulting transformed point.
- eg. Translation

<u>Scaling</u>

$$\text{scale}(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

<u>Rotation</u>

$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

<u>Translation</u>

$$\text{translate}(dx, dy) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

💡 Why not remove the inconsistency by expressing **all three** 2D transformations as **3x3 matrices** !?

- For linear transformations (i.e., scaling and rotation), embed the existing 2x2 matrix in the upper-left of a new 3x3 matrix: i.e

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of Transformations

- Transformation is a function; by associativity, we can compose functions

$$(f \circ g)(x) \equiv f(g(x))$$

- Given transformations $\boldsymbol{M_1}$, $\boldsymbol{M_2}$ and input vertex $\boldsymbol{v}$, our composition is equivalent to

$$v' = M_1 M_2 v$$

- However, **Order** of transformations **matters.** Matrix multiplication is **NOT commutative.**

(The above content has been taken from lecture slides of Prof. Ojaswa Sharma)

All read and understood ? Moving on to the actual task...

Your task, should you choose to accept it, is to write an **interactive python program** to apply transformations (both linear and non - linear) to an object and plot it using **matplotlib.**

Your program should support the following objects:
- A Disc (of radius **r** centered at **(a,b)**)
- A polygon (vertices of which are specified by lists **X** and **Y**)

Input
The first contains the word **'disc'** or **'polygon'** denoting the figure you have to use.
If the word is '**disc'**, the next line contains three space-separated integers **a b r** as specified above.
If the word is **'polygon'**, the next **two lines** contain space separated lists **X[]** and **Y[]** of equal length, denoting the x-y co-ordinates of the vertices of the polygon.

The next few lines contain a single query each, denoting the transformation you have to perform. Each query will be of the form:
- **S x y :** scale the object by a factor of **x** along the x-axis, and **y** along the y-axis.
- **R theta :** rotate the object by angle theta(in degrees, 0 <= theta <= 360) in the **counter-clockwise** direction about the origin.
- **T dx dy :** translate the object by **dx** units along the x-axis, and by **dy** units along the y-axis.

Each transformation has to be performed on the shape obtained as a result of all the previous transformations, ie the effect of the transformations should be cumulative.

The final line of the input contains the word **'quit'**, denoting that no further transformations are required and you should exit the program.

Output
Plot the input object according to the input specifications.
For each transformation, you should :

- print the new resulting parameters(**a b r** for disc and **x[] y[]** for polygon) of the object in the format specified in the input.
- update the plot to show the new object position.

Sample Input

polygon
1 -1 -1 1
1 1 -1 -1
S 2 1
R 90
T 0 -2
quit

Sample Output : **The plot should be updated at each step of the output**

2 -2 -2 2
1 1 -1 -1

-1 -1 1 1
2 -2 -2 2

-1 -1 1 1
0 -4 -4 0

Sample code to make an interactive plot:

```
import matplotlib.pyplot as plt
plt.ion()           # makes the plot interactive
for i in range(5):
      x,y = map(int, input("Enter two space separated numbers : ").split())
      plt.plot([x,y])
```