Affline Transformation

In this section, I'll be implementing some Affline Transformations. This program allows the user to create some polygons by a series of left mouse clicks followed by one middle mouse click. After that, the user can choose to select a polygon and do the following transformation with it:

- Rotate left/right by 10 degrees
- Zoom in/out by 10%
- Move the image left/right/up/down by one pixel at a time

So now I will illustrate how I implemented these functionalities. For Affline Transformations, there is a follow framework:

- Keep the coordinate of all vertices of the polygon
- Suppose that we have a function f that transform the coordinates of each point to the new point according to the Transformation
- Delete the old vertices from the buffer
- Use the transformed vertices to draw the edges for this polygon

That was the common framework for Affline Transformation. The transformation-dependent function f needs to be implemented differently for each transformation. Next I am going to describe them. For each transformation, we suppose that we have the coordinate of a point x, y, and we want to return new point x', y'. In the case of rotation/zooming, we need to go through to following steps:

- Apply a transformation that move the center (xt, yt) of rotation/zoom to (0, 0):
    - x' = x - xt, y' = y - yt
- Apply the rotation/scaling transformation f' around (0, 0)
    - For rotation, this transformation is:
        - x' = x' ** cos(alpha) - y' ** sin(alpha)
        - y' = x' ** sin(alpha) + y' ** cos(alpha)
    - For scaling, this transformation is:
        - x' = x' * (1 + scale)
        - y' = y' * (1 - scale)
- Apply a transformation that move the center of rotation/zoom back to where it was:
    - x' = x' + xt, y' = y' + yt

In the case of moving the point (x, y) either up, down, left, right d pixels, we do the following:

- x' = x + dx, y' = y + dy

with the value of (dx, dy) be:

- Up: dx = 0, dy = -d
- Down: dx = 0, dy = d
- Left: dx = -d, dy = 0
- Right: dx = d, dy = 0