# Description of Program

My program will display 5 images in an “X” pattern. There are five different frames the program will loop through, with the first being the starting point without any transformations. The five images, row-by-row left-to-right, are an alpaca, Pikachu face, the Triforce, Megaman, and a bowl of noodles.

Aside from using a new PixelImage class instead of the ImageTemplate class, some changes have been made. I retained the IMGSIZEX and IMGSIZEY and added an array of 10 colors, whose indices were used in the separate image arrays. The Color class didn’t have a value for BROWN, so I created my own Color object for BROWN, which was included in the array. By using an array to reference colors, I was able to do away with the if-else statement within the double-loop used to iterate through the image arrays. Instead, I was able to get the color from the colors array then assign that color to the appropriate row and column. I had to do some playing around with the order of the row and col assignments to get the image to appear upright, but they all display properly now. I also removed the letterT image and replaced it with five new arrays. Most of the images were translated from pixel art found online; credit is given to the creators as comments within the code.

The biggest changes to the main CMSC325Animate class involved adding the five images to the Graphics2D object. I wanted to make sure that all five images were fully in view for all frames and ended up using the “X” pattern. This gave the images plenty of room to not overlap in any of the required frames. Another change needed to display the images properly was the left, right, top, and bottom parameters to applyWindowToViewportTransformation. The way it was originally set, the images always displayed upside down. I was able to flip it by changing the signs for the top and bottom parameters. Then all I needed to do was adjust the values for all four sides until all images were nicely fit into the window.

Since the original class had an output to console displaying the current frame, I was able to see that a frame 5 was being hit. I had changed the maximum number off frames in the timer to 4, since I had four frames. This frame 5 struck me as odd, so I looked at the timer’s actionPerformed method. I found another if-else statement that could be refactored. I quickly deduced that the block was simply setting the value of frameNumber to 0 when it exceeded the number of frames. I was able to resolve the issue by modified the whole if-else statement into a single MOD statement. Essentially adding one to the current value and getting the remainder after dividing by 5, the total number of frames including the default. This creates a number that loops from 0 through 4, because 5 % 5 is 0. Now, aside from the very beginning, the default frame is never repeated.

Finally, I had to ensure all the required transformations were applied in order. For frame 0, the “default” state, I reset all values to their original state. I don’t like empty default cases, so I used the default case for my Frame 0 reset. For all other cases the transformation was added, or multiplied, to the existing values. This met the requirement to have all transformations “starting from the previous transformation as opposed to the original image.” For the rotations, I discovered the template had used a positive value for counterclockwise rotation, but that same value was clockwise for mine as a result of switching signs for the top and bottom parameters to the earlier function call. To correct the issue, I simply switched the sign for my degrees of rotation.

# Screenshots

1. Default; No Transformations  
   A picture containing LEGO, toy, vector graphics, screenshot

   Description automatically generated
2. Translate -10 in x direction, Translate +12 in the y direction.  
   Logo, company name

   Description automatically generated
3. Rotate 55 degrees counter clockwise.  
   A picture containing text, queen, vector graphics, toy

   Description automatically generated
4. Rotate 75 clockwise  
   A picture containing logo

   Description automatically generated
5. Scale 3 times for the x component, scale 1.5 times for the y component  
   A picture containing engineering drawing

   Description automatically generated