In [61]:
```
from PIL import Image
from matplotlib.pyplot import imshow
import numpy as np
%matplotlib inline
```

In [69]:
```
tri1 = [[150,200],[250,200],[200,300]]

tri2 = [[180, 120],[220, 80],[220, 150]]
```

In [70]:
```
def isWithinBoundaries(x, y, nRows, nCols):
    # Return True if (x,y) is within the boundaries of matrix A(nRows,nCols)
    return (x>=0 and x<nRows and y >= 0 and y < nCols)
```

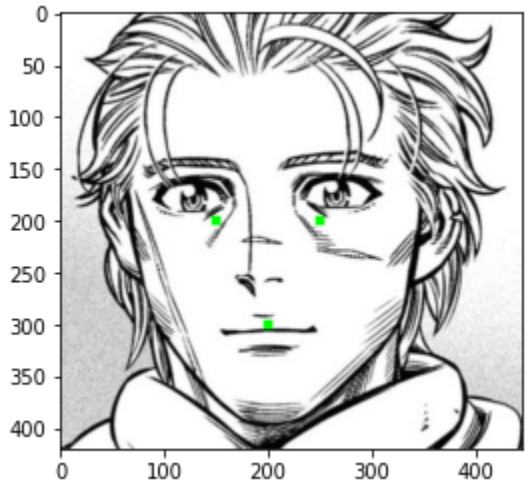
In [71]:
```
def inverseMapping(u, v, affine_matrix):
    point_matrix = np.array([
                             [u],
                             [v],
                             [1],
                             ])
    new_point_matrix = np.matmul(affine_matrix, point_matrix)
    x = new_point_matrix.item(0)
    y = new_point_matrix.item(1)

    return x,y
```

In [72]:
```
def getAffineMatrix(tri1, tri2):
    point_matrix = np.array([
                             [tri1[0][0], tri1[0][1], 1, 0, 0, 0],
                             [0, 0, 0, tri1[0][0], tri1[0][1], 1],
                             [tri1[1][0], tri1[1][1], 1, 0, 0, 0],
                             [0, 0, 0, tri1[1][0], tri1[1][1], 1],
                             [tri1[2][0], tri1[2][1], 1, 0, 0, 0],
                             [0, 0, 0, tri1[2][0], tri1[2][1], 1],
                            ])
    inv_point_matrix = np.linalg.inv(point_matrix)
    tri2_point_matrix = np.array([
                                  [tri2[0][0]],
                                  [tri2[0][1]],
                                  [tri2[1][0]],
                                  [tri2[1][1]],
                                  [tri2[2][0]],
                                  [tri2[2][1]],
                                 ])
    affine_var_matrix = np.matmul(inv_point_matrix, tri2_point_matrix)
    affine_matrix = np.array([
                              [affine_var_matrix.item(0,0), affine_var_matrix.item(1,0), af
                              [affine_var_matrix.item(3,0), affine_var_matrix.item(4,0), af
                              [0, 0, 1],
                             ])
    return affine_matrix
```

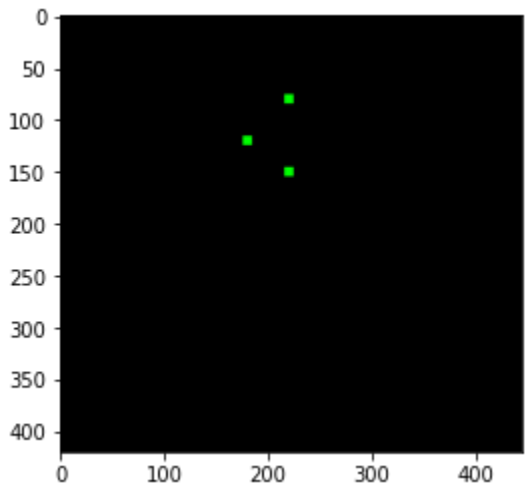
In [73]:
```
# Get Source Image
im = Image.open('test.JPG', 'r')
im_tri = im.copy()
# Show Triangle 1
for point in tri1:
    for dx in range(-4,5):
        for dy in range(-4,5):
            x = point[0]+dx
            y = point[1]+dy
            if isWithinBoundaries(x, y, M, N):
                im_tri.putpixel( (x, y), (0,255,0) )
imshow(np.asarray(im_tri))
```

Out[73]: <matplotlib.image.AxesImage at 0x1b0c6b662e0>



In [74]:
```
# Create New Empty Image
M,N = im.size
new_im = Image.new(mode='RGB', size=(M,N), color=0)
new_im_tri = new_im.copy()
# Show Triangle 2
for point in tri2:
    for dx in range(-4,5):
        for dy in range(-4,5):
            x = point[0]+dx
            y = point[1]+dy
            if isWithinBoundaries(x, y, M, N):
                new_im_tri.putpixel( (x, y), (0,255,0) )
imshow(np.asarray(new_im_tri))
```

Out[74]: <matplotlib.image.AxesImage at 0x1b0c6bcf580>



In [75]:
```
# Apply Affine Warping
affine_matrix = getAffineMatrix(tri1, tri2)
for u in range(0,M):
    for v in range(0,N):
        x,y = inverseMapping(u, v, affine_matrix)
        # Color Pixels
        if isWithinBoundaries(x, y, M, N):
            new_im.putpixel( (int(x),int(y)), im.getpixel((u,v)))
new_im_tri = new_im.copy()
# Show Triangle 2
for point in tri2:
    for dx in range(-4,5):
        for dy in range(-4,5):
            x = point[0]+dx
            y = point[1]+dy
            if isWithinBoundaries(x, y, M, N):
                new_im_tri.putpixel( (x, y), (0,255,0) )
imshow(np.asarray(new_im_tri))
```

Out[75]: <matplotlib.image.AxesImage at 0x1b0c803cbb0>