# Line Representation and simple Transformation

## Line Representation

Line can be represented mathematically by:
P = Po + tV

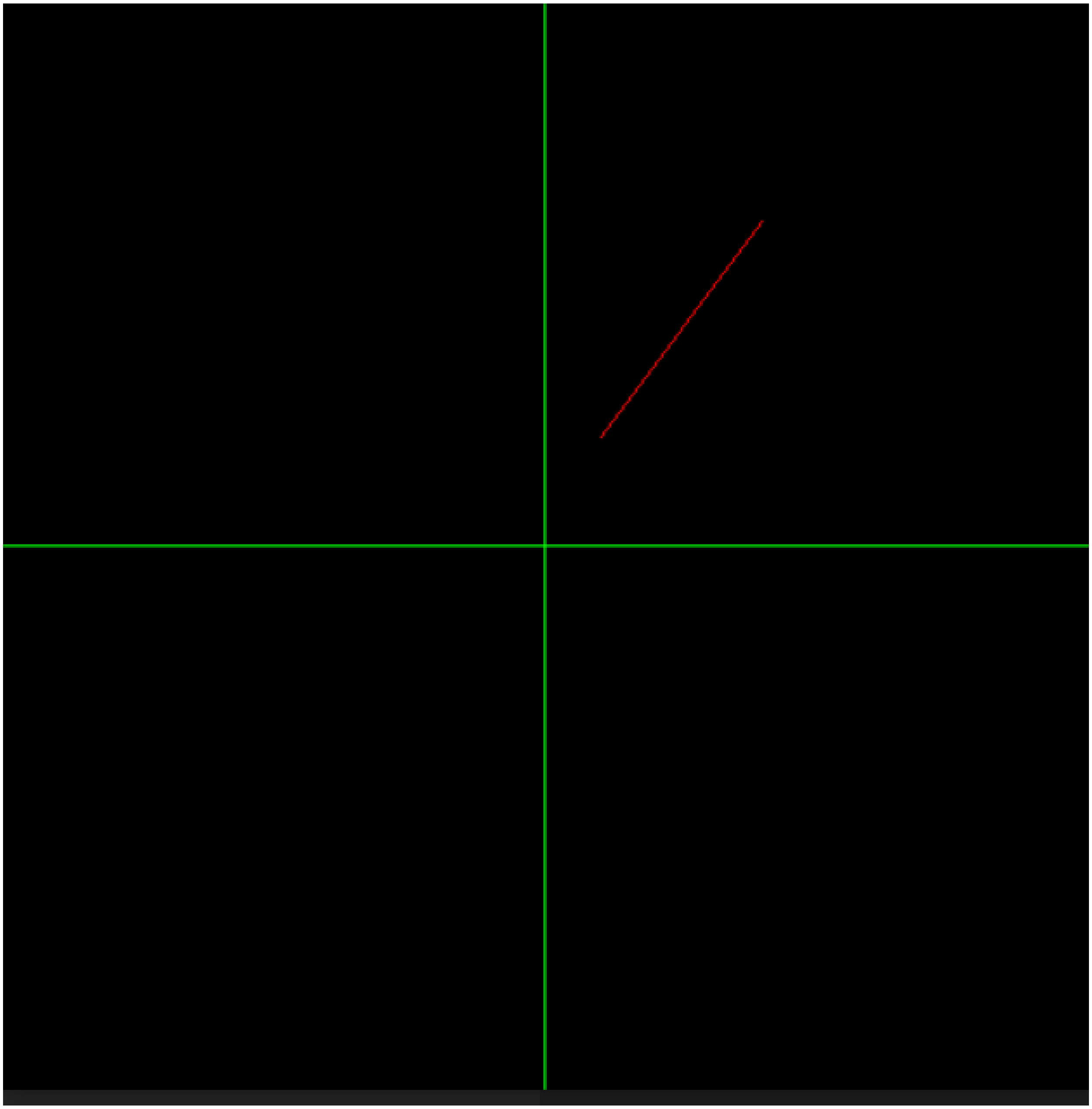
This means we have an initial point Po and on that point, we add some ‘scaled’ vector V.
Let our vector V = [0.3, 0.4].

The length (Norm) of V is √ ((0.3)2 + (0.4)2) = 0.5 units

The direction of the vector is about 53 degree, calculated by tan inverse function.
Let the Po = (0.1, 0.2)

Now if we calculate P = Po + tV, for t = 1, we will get a line that starts from Po in a direction of 53’ with length of 0.5 units
a. If t < 1 and t > 0 then the line will be less than 0.5 units in length.
b. If t > 1 then the line will be more that 0.5 units in length
c. If t < 0 then the line will be drawn backwards.

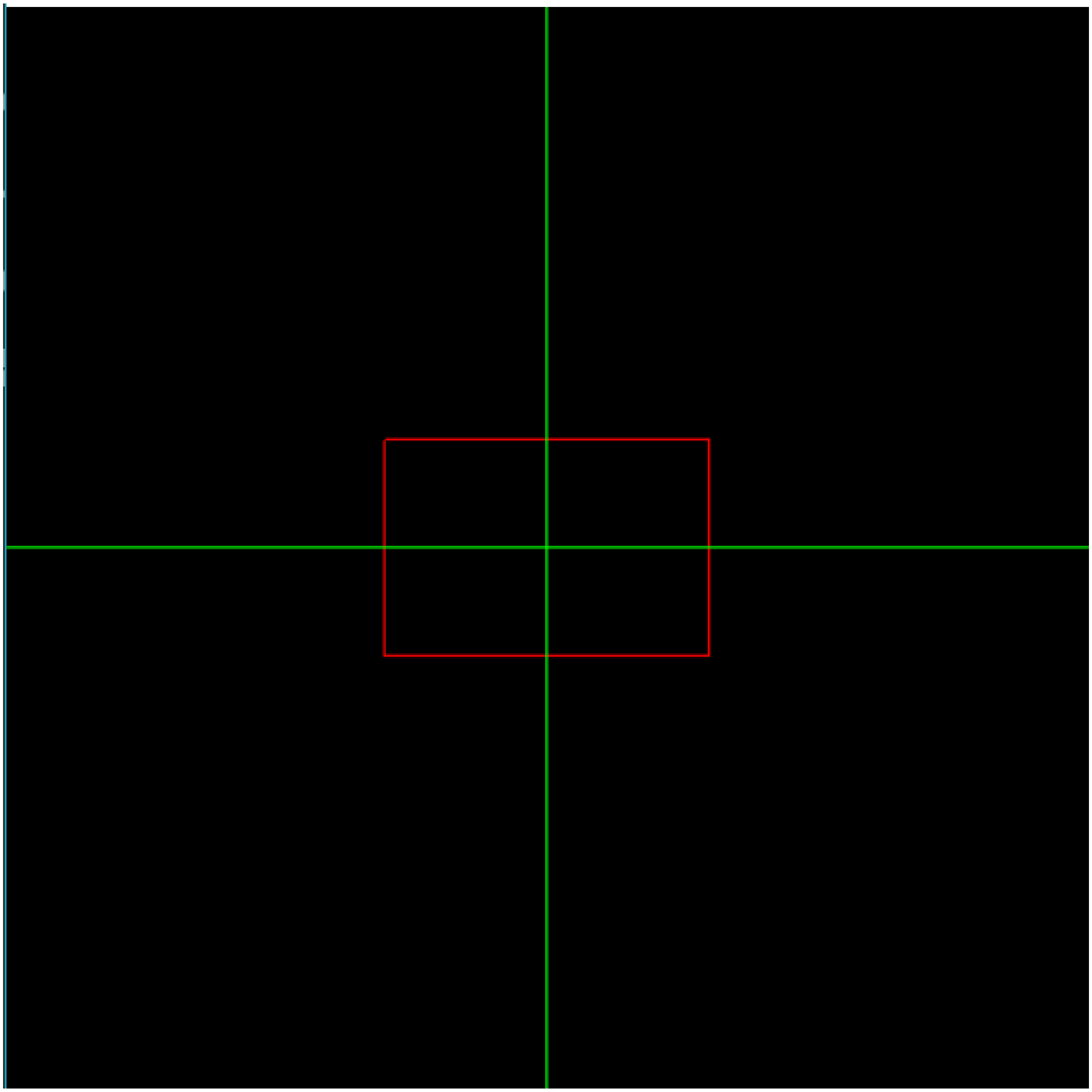
Try to program this in PyOpenGL. Start with the basic.py file and follow the instructions.
1. Create a numpy vector that contains [0.3, 0.4]
2. Create a point with numpy that is (0.1, 0.2)
3. Calculate P using P = Po + tV with t = 1
4. Draw the line with PyOpenGL
5. Recalculate and draw the line with t = 0.5 and t = 1.2 and see the difference
6. Draw a grid line X-Axis and Y-Axis with different color



**Exercise**:

a. Draw a rectangle by drawing four sides of the rectangles as a line. Its side must be a width of 0.6 and height of 0.4 units and the rectangle must be in the center. It might be easy for you if you first calculate on paper.

Here is a screenshot that is expected.



## Transformation

Transformation is generally a change applied to a certain object. For example, change in size of an object, change in orientation (Rotation).

On this exercise, we will see a simple rotation of the box we drew on Exercise 1.

Import everything from transorm.py in your exercise 1 python file.

In transform.py we have a function rotationMat function which returns a rotation matrix. A rotation matrix is a matrix that rotates an object along an axis when it is multiplied with the object's vertices.

Example. Let P be a point and M be a rotation matrix.
P X M = P rotated

So to rotate an object, we simply need to multiply all our points with a rotation matrix.
The rotationMat function in transform.py accepts parameters of degree by which we rotate an object.

**Exercise 2:**
Rotate the box you have made on Exercise 1
a. Create a variable mat and assign it to rotation matrix of 60'
b. Using np.dot function multiply the points of the box with mat
c. Add .4 to all points and see what happens.