NAVEEN
1601CS28

CS 372
February 20, 2019

Graphics Assignment No:- 3

1. Homogenous Coordinate

we can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system. In this system, we can represent all the transformation equations in matrix multiplication. Any Cartesian point P(X, Y) can be converted to homogenous coordinates by P (Xh, Yh, h).

2D Transformation:

1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shear

2. Translation

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X, Y).
X = X + tx
Y = Y + ty

Python code

```
   #translation
%matplotlib inline

f=()
points = ((100, 100), (200, 100), (200, 200), (100, 200))
w=np.array([[3,0],[0,3]])
for i in points:
    a=i[0]+100
    b=i[1]+200
    z=np.array([[a],[b]])
    transformed=np.dot(w,z)
    z=[transformed[0][0],transformed[1][0]]


    z=tuple(z)
    f=f+z
image = Image.new("RGB", (1640, 1480))
draw = ImageDraw.Draw(image)
draw.polygon((f), fill=200)
import matplotlib.pyplot as plt
plt.imshow(image)


%matplotlib inline

f=()
points = ((100, 100), (200, 100), (200, 200), (100, 200))
w=np.array([[3,0],[0,3]])
for i in points:
    a=i[0]+100
```

```
        b=i[1]+200
        z=np.array([[a],[b]])
        transformed=np.dot(w,z)
        z=[transformed[0][0],transformed[1][0]]


        z=tuple(z)
        f=f+z
    image = Image.new("RGB", (1640, 1480))
    draw = ImageDraw.Draw(image)
    draw.polygon((f), fill=200)
    import matplotlib.pyplot as plt
    plt.imshow(image)
```

3. Rotation

In rotation, we rotate the object at particular angle (theta) from its origin. From the following figure, we can see that the point P(X, Y) is located at angle from the horizontal X coordinate with distance r from the origin.Let us suppose you want to rotate it at the angle . After rotating it to a new location, you will get a new point P (X, Y).
x=xcosysin
y=xsin+ycos

Python code

```
    %matplotlib inline

  f=()
  points = ((100, 100), (200, 100), (200, 200), (100, 200))
  w=np.array([[3,0],[0,3]])
  for i in points:
      a=i[0]+100
      b=i[1]+200
      z=np.array([[a],[b]])
      transformed=np.dot(w,z)
      z=[transformed[0][0],transformed[1][0]]


      z=tuple(z)
      f=f+z
  image = Image.new("RGB", (1640, 1480))
  draw = ImageDraw.Draw(image)
  draw.polygon((f), fill=200)
  import matplotlib.pyplot as plt
  plt.imshow(image)
```

4. Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.Let us assume that the original coordinates are (X, Y), the scaling factors are (SX, SY), and the produced coordinates are (X, Y). This can be mathematically represented as shown below
X' = X . SX and
Y' = Y . SY

Python code

```
#scaling
%matplotlib inline
f=()
points = ((100, 100), (200, 100), (200, 200), (100, 200))
w=np.array([[3,0],[0,3]])
for i in points:
    a=i[0]
    b=i[1]
    z=np.array([[a],[b]])
    transformed=np.dot(w,z)
    z=[transformed[0][0],transformed[1][0]]


    z=tuple(z)
    f=f+z
image = Image.new("RGB", (1640, 1480))
draw = ImageDraw.Draw(image)
draw.polygon((f), fill=200)
import matplotlib.pyplot as plt
plt.imshow(image)
```

5. Reflection

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180. In reflection transformation, the size of the object does not change.The following figures show reflections with respect to X and Y axes, and about the origin respectively.

Python code

```
%matplotlib inline

f=()
points = ((100, 100), (200, 100), (200, 200), (100, 200))
w=np.array([[3,0],[0,3]])
for i in points:
    a=i[0]+100
    b=i[1]+200
    z=np.array([[a],[b]])
    transformed=np.dot(w,z)
    z=[transformed[0][0],transformed[1][0]]


    z=tuple(z)
    f=f+z
image = Image.new("RGB", (1640, 1480))
draw = ImageDraw.Draw(image)
draw.polygon((f), fill=200)
import matplotlib.pyplot as plt
plt.imshow(image)


%matplotlib inline

f=()
points = ((100, 100), (200, 100), (200, 200), (100, 200))
w=np.array([[3,0],[0,3]])
```

```
    for i in points:
        a=i[0]+100
        b=i[1]+200
        z=np.array([[a],[b]])
        transformed=np.dot(w,z)
        z=[transformed[0][0],transformed[1][0]]


        z=tuple(z)
        f=f+z
    image = Image.new("RGB", (1640, 1480))
    draw = ImageDraw.Draw(image)
    draw.polygon((f), fill=200)
    import matplotlib.pyplot as plt
    plt.imshow(image)
```

6. Reflection

A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations X-Shear and Y-Shear. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values. Shearing is also termed as Skewing.
X-Shear
The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left as shown in below figure.

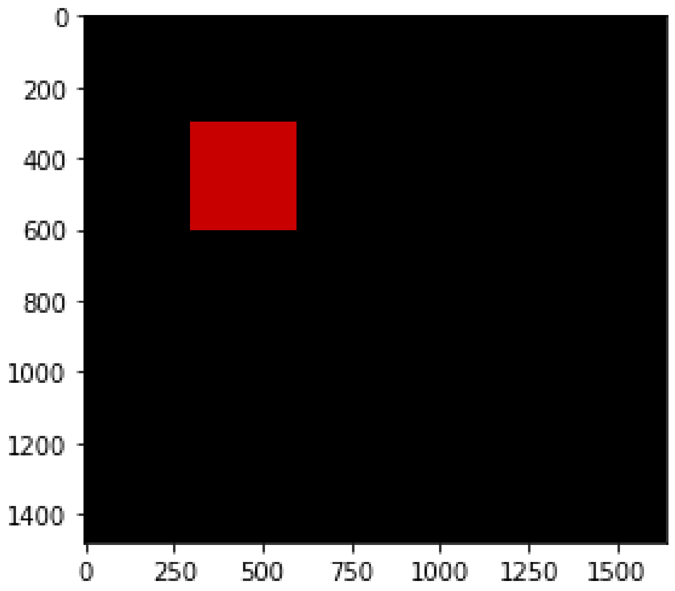
Y-Shear

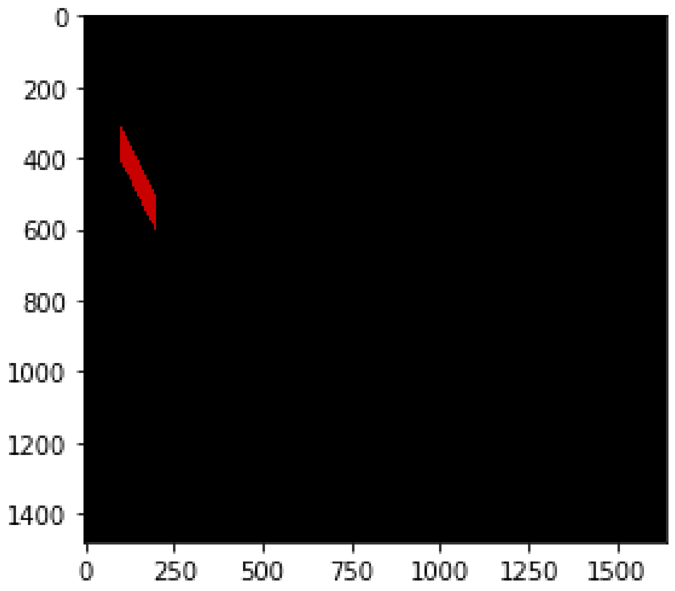
The Y-Shear preserves the X coordinates and changes the Y coordinates which causes the horizontal lines to transform into lines which slopes up or down as shown in the following figure. **Python code**

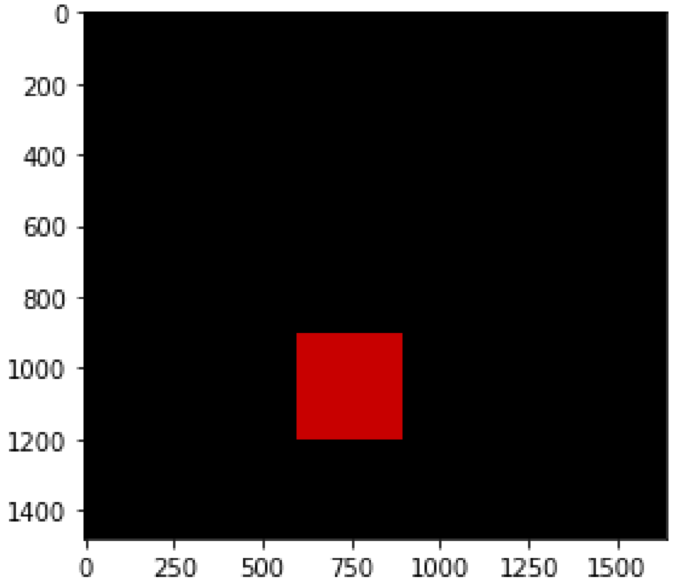
```
  #shear
 %matplotlib inline
 f=()
 points = ((100, 100), (200, 100), (200, 200), (100, 200))
 w=np.array([[1,0],[2,1]])
 for i in points:
     a=i[0]
     b=i[1]
     z=np.array([[a],[b]])
     transformed=np.dot(w,z)
     z=[transformed[0][0],transformed[1][0]]


     z=tuple(z)
     f=f+z
 image = Image.new("RGB", (1640, 1480))
 draw = ImageDraw.Draw(image)
 draw.polygon((f), fill=200)
 import matplotlib.pyplot as plt
 plt.imshow(image)
```

5. Result

2D transformation


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500