

Computer Animation Lab: 04

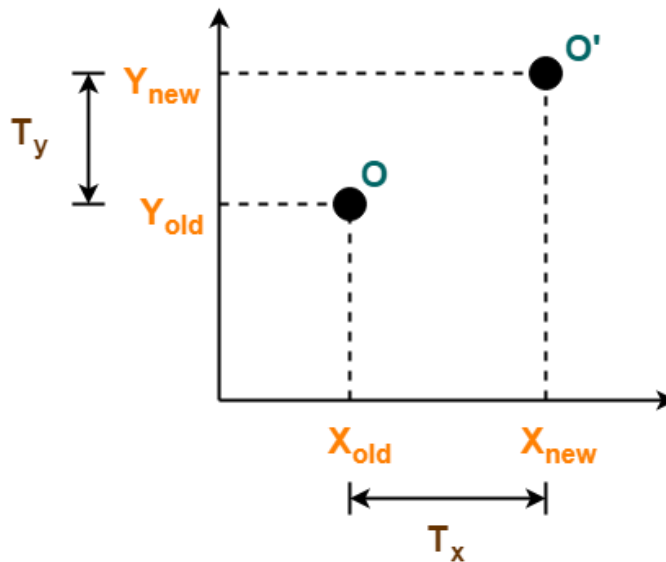
Contents

Translation	3
Code	4
Scaling	5
Code	6
Rotation	8
Code	9

- In this lab, we will apply the animation of the 2D transformations.
- Transformations are the processes of re-positioning or modifying an object in a two dimensional plane.
- Transformation techniques:
 - Translation
 - Rotation
 - Scaling
 - Reflection
 - Shear
- We will apply the translation, rotation, and scaling only.

Translation

- Translation is a process of moving an object from one position to another.
- Suppose that we have a point O that we need to translate in a 2D plane.
- Let:
 - Initial coordinates of the point are $O = (X_{old}, Y_{old})$
 - X-axis translation distance is T_x , and y-axis translation is T_y
 - New coordinates of the point are $O = (X_{new}, Y_{new})$



- The translation can be represented as:

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$X_{new} = X_{old} + T_x$$

$$Y_{new} = Y_{old} + T_y$$

Code

```
from tkinter import *

def translate(cx, cy, T):
    tcx = cx + T[0]
    tcy = cy + T[1]
    return tcx, tcy

window = Tk()
cnvs = Canvas(window, width=500, height=500)
cnvs.pack()
cx = 250; cy = 250
width = 100; height = 200
x1 = cx - (width / 2)
y1 = cy - (height / 2)
x2 = cx + (width / 2)
y2 = cy - (height / 2)
x3 = cx + (width / 2)
y3 = cy + (height / 2)
x4 = cx - (width / 2)
y4 = cy + (height / 2)
cnvs.create_polygon(x1, y1, x2, y2, x3, y3, x4, y4, tags="p1")

endx = 200; endy = 80
tx = 5; ty = 5
while tx < endx and ty < endy:
    tcx, tcy = translate(cx, cy, (tx, ty))

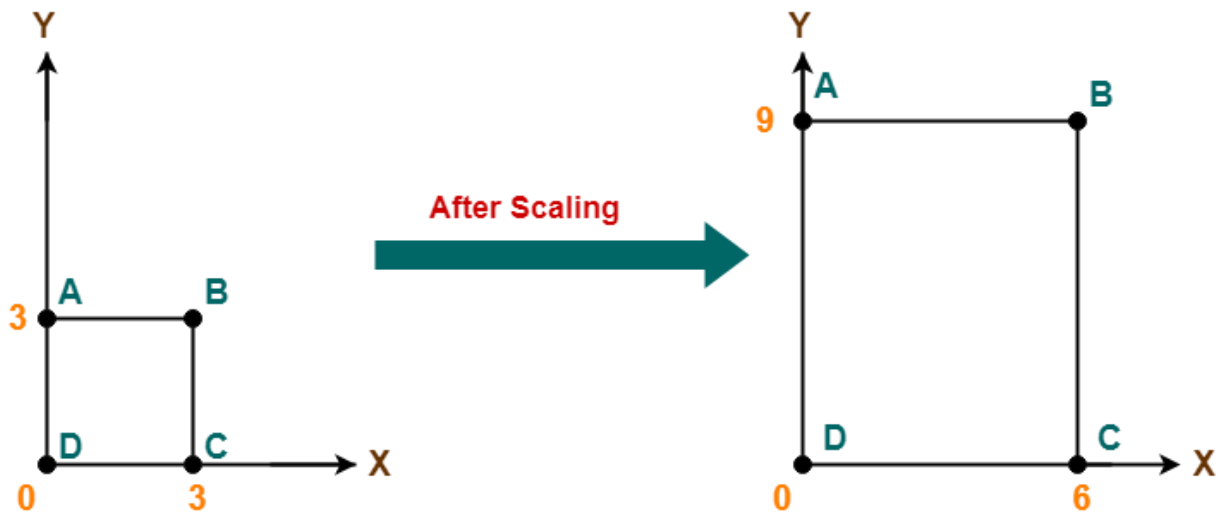
    x1 = tcx - (width / 2)
    y1 = tcy - (height / 2)
    x2 = tcx + (width / 2)
    y2 = tcy - (height / 2)
    x3 = tcx + (width / 2)
    y3 = tcy + (height / 2)
    x4 = tcx - (width / 2)
    y4 = tcy + (height / 2)
    cnvs.delete("p1")
    cnvs.create_polygon(x1, y1, x2, y2, x3, y3, x4, y4, tags="p1")
    cnvs.after(100)
    cnvs.update()

    if tx < endx:
        tx += 5
    if ty < endy:
        ty += 5

window.mainloop()
```

Scaling

- Scaling is a process of modifying or altering the size of objects.
 - Scaling may be used to increase or reduce the size of object.
- Scaling factor determines whether the object size is to be increased or reduced.
 - If scaling factor > 1 , then the object size is increased.
 - If scaling factor < 1 , then the object size is reduced.



- Let-
 - Initial coordinates of the object $O = (X_{old}, Y_{old})$
 - Scaling factor for X-axis = S_x
 - Scaling factor for Y-axis = S_y
 - New coordinates of the object O after scaling = (X_{new}, Y_{new})
- Scaling is achieved using the following equation

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

$$x_{new} = x_{old} * S_x$$

$$y_{new} = y_{old} * S_y$$

Code

```
from tkinter import *

def scale(lt, rt, rb, lb, S):
    sx1 = lt[0] * S[0]
    sy1 = lt[1] * S[1]
    sx2 = rt[0] * S[0]
    sy2 = rt[1] * S[1]
    sx3 = rb[0] * S[0]
    sy3 = rb[1] * S[1]
    sx4 = lb[0] * S[0]
    sy4 = lb[1] * S[1]
    return sx1, sy1, sx2, sy2, sx3, sy3, sx4, sy4

window = Tk()
cnvs = Canvas(window, width=500, height=500)
cnvs.pack()
cx = 250
cy = 250
width = 100
height = 200
x1 = cx - (width / 2)
y1 = cy - (height / 2)
x2 = cx + (width / 2)
y2 = cy - (height / 2)
x3 = cx + (width / 2)
y3 = cy + (height / 2)
x4 = cx - (width / 2)
y4 = cy + (height / 2)
cnvs.create_polygon(x1, y1, x2, y2, x3, y3, x4, y4, tags="p1")

sx1, sy1, sx2, sy2, sx3, sy3, sx4, sy4 = scale((x1, y1), (x2, y2),
(x3, y3), (x4, y4), (1.01, 1.3))
endx = 1 + 3/100
endy = 1 + 3/100
sx = 1
sy = 1
```

```

while sx < endx or sy < endy:
    sx1, sy1, sx2, sy2, sx3, sy3, sx4, sy4 = scale((x1, y1), (x2, y2),
    (x3, y3), (x4, y4), (sx, sy))

    width = abs(sx1 - sx2)
    height = abs(sy1 - sy4)
    x1 = cx - (width / 2)
    y1 = cy - (height / 2)
    x2 = cx + (width / 2)
    y2 = cy - (height / 2)
    x3 = cx + (width / 2)
    y3 = cy + (height / 2)
    x4 = cx - (width / 2)
    y4 = cy + (height / 2)
    cnvs.delete("p1")
    cnvs.create_polygon(x1, y1, x2, y2, x3, y3, x4, y4, tags="p1")
    cnvs.after(100)
    cnvs.update()

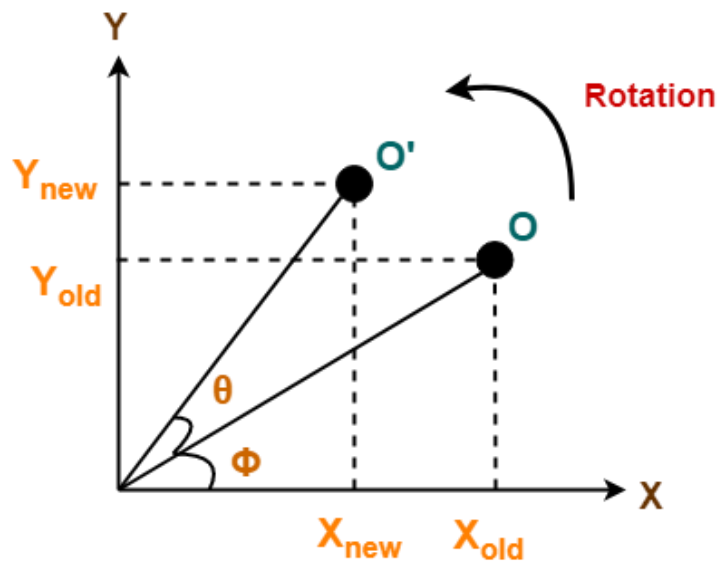
    if sx < endx:
        sx += 0.001
    if sy < endy:
        sy += 0.001

    print(sx, sy)
window.mainloop()

```

Rotation

- 2D Rotation is a process of rotating an object with respect to an angle in a two-dimensional plane.
- Let-
 - Initial coordinates of the object $O = (X_{old}, Y_{old})$
 - Rotation angle = θ
 - New coordinates of the object O after rotation = (X_{new}, Y_{new})



- This rotation is achieved by using the following rotation equations

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

$$x_{new} = x_{old} * \cos(\theta) - y_{old} * \sin(\theta)$$

$$y_{new} = x_{old} * \sin(\theta) + y_{old} * \cos(\theta)$$

Code

```
from tkinter import *
from math import *

def rotate(cx, cy, lt, rt, rb, lb, theta):
    # Convert degrees to radians
    angle = radians(theta)

    # shift the points to the origin (0,0)
    x1 = lt[0] - cx
    y1 = lt[1] - cy
    x2 = rt[0] - cx
    y2 = rt[1] - cy
    x3 = rb[0] - cx
    y3 = rb[1] - cy
    x4 = lb[0] - cx
    y4 = lb[1] - cy

    # Rotate the vertices
    rx1 = cx + ((x1 * cos(angle)) - (y1 * sin(angle)))
    ry1 = cy + ((x1 * sin(angle)) + (y1 * cos(angle)))

    rx2 = cx + ((x2 * cos(angle)) - (y2 * sin(angle)))
    ry2 = cy + ((x2 * sin(angle)) + (y2 * cos(angle)))

    rx3 = cx + ((x3 * cos(angle)) - (y3 * sin(angle)))
    ry3 = cy + ((x3 * sin(angle)) + (y3 * cos(angle)))

    rx4 = cx + ((x4 * cos(angle)) - (y4 * sin(angle)))
    ry4 = cy + ((x4 * sin(angle)) + (y4 * cos(angle)))

    print("%.1f" % (rx1), "%.1f" % (ry1), "%.1f" % (rx2), "%.1f" %
(ry2), "%.1f"
        % (rx3), "%.1f" % (ry3), "%.1f" % (rx4), "%.1f" % (ry4))

    return rx1, ry1, rx2, ry2, rx3, ry3, rx4, ry4
window = Tk()
cnvs = Canvas(window, width=500, height=500)
cnvs.pack()
cx = 250
cy = 250
width = 100
height = 200
```

```

cnvs.create_polygon(x1, y1, x2, y2, x3, y3, x4, y4, tags="p1")
ang = -5
while True:
    rx1, ry1, rx2, ry2, rx3, ry3, rx4, ry4 = rotate(cx, cy, (x1, y1),
(x2, y2), (x3, y3), (x4, y4), ang)

    cnvs.delete("p1")
    cnvs.create_polygon(rx1, ry1, rx2, ry2, rx3, ry3, rx4, ry4,
tags="p1")
    cnvs.after(50)
    cnvs.update()
    ang -= 5

window.mainloop()

```