# Computer Graphics Lab

**1. Point**

import turtle

turtle.title("Point")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

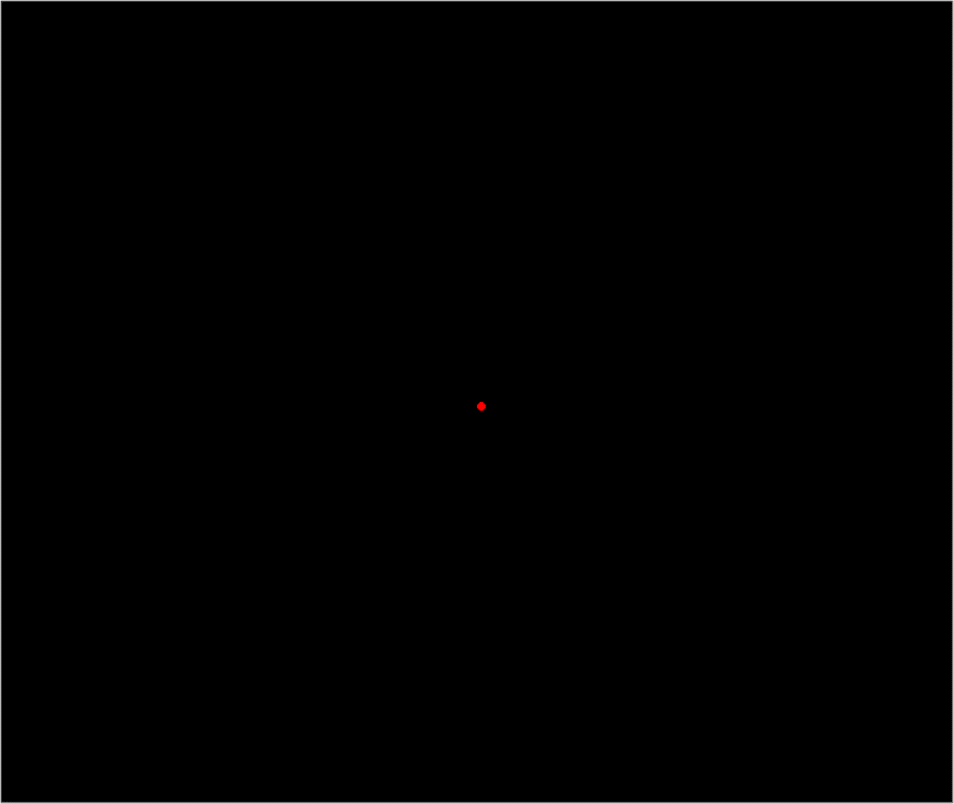
t.speed("fastest")

t.width(3)

t.color("red")

t.dot()

t.ht()



**2. Line**

import turtle

turtle.title("Line")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

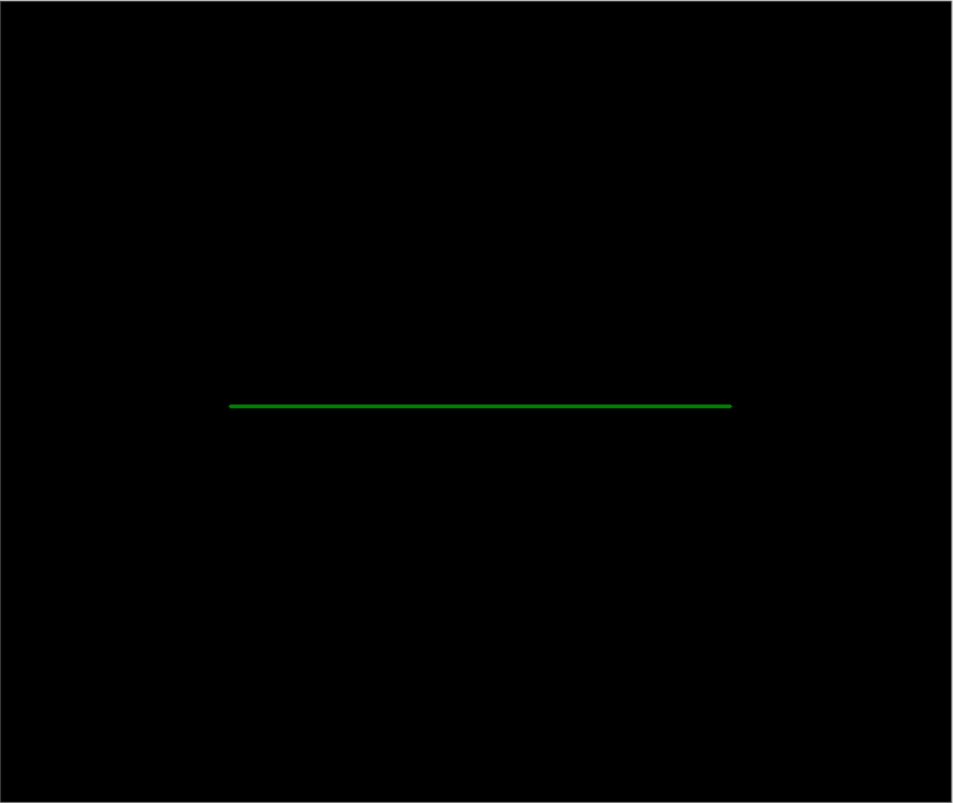
t.width(3)

t.color("green")

t.backward(200)

t.forward(400)

t.ht()



**3. Triangle**

import turtle

turtle.title("Traingle")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(3)

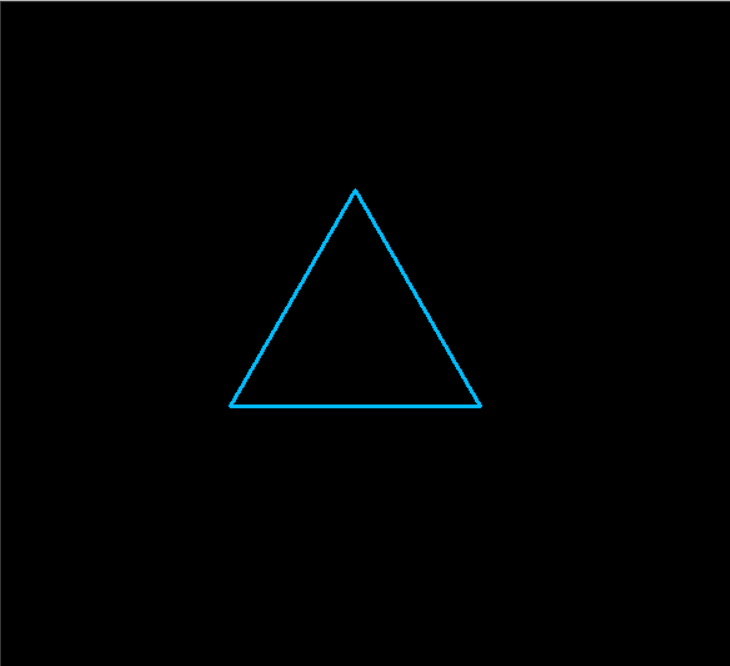
t.color("deepskyblue")

for i in range(3):

t.left(120)

t.forward(200)

t.ht()



**4. Rectangle**

import turtle

turtle.title("Rectangle")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(2)

t.color("gold")

for i in range(2):

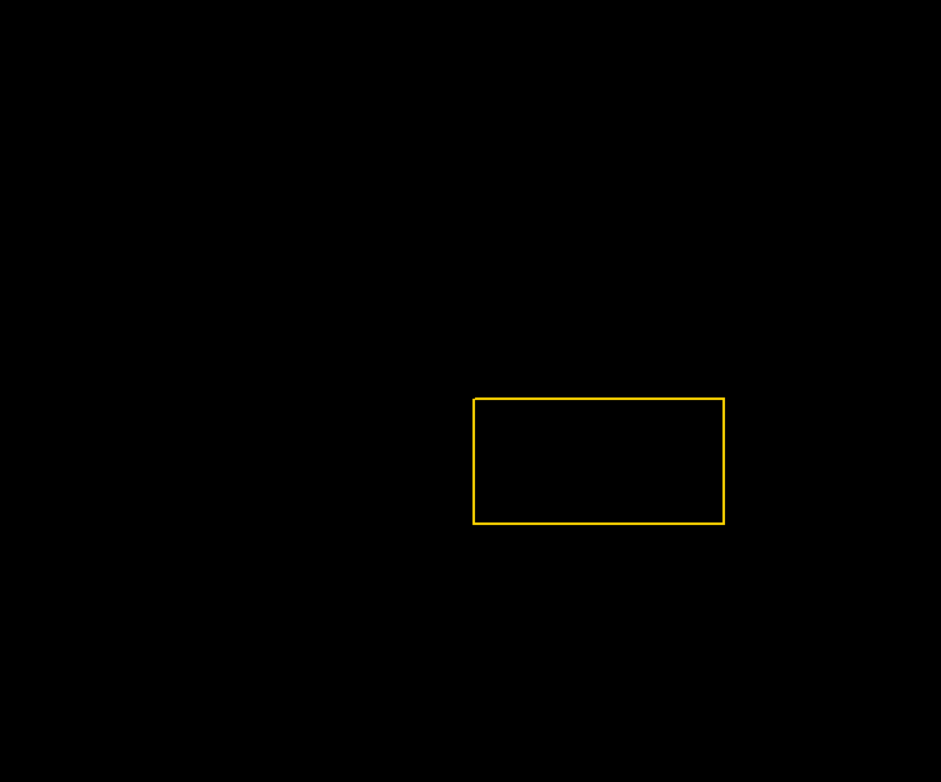
t.forward(200)

t.right(90)

t.forward(100)

t.right(90)

t.ht()



**5. Circle**

import turtle

turtle.title("Circle")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

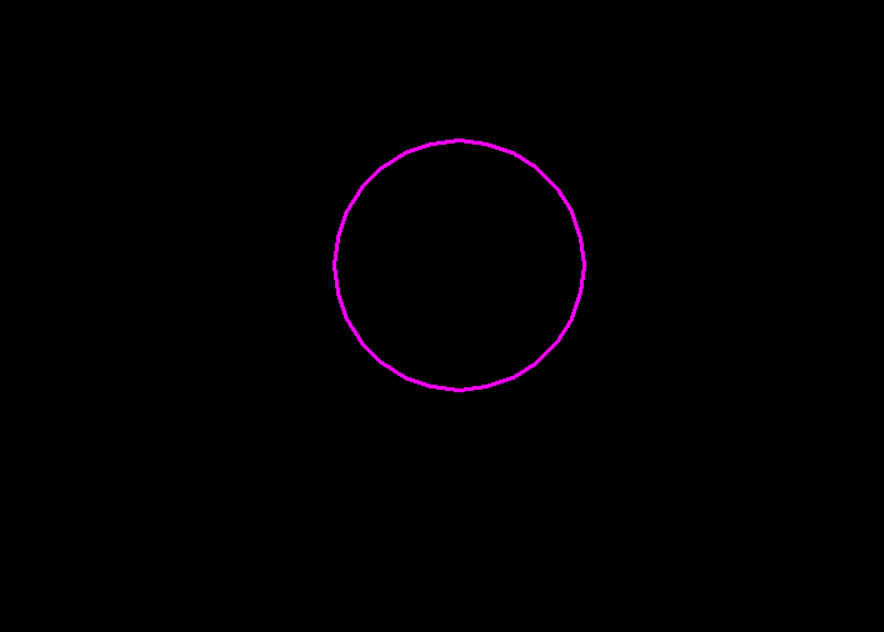
t.speed("fastest")

t.width(3)

t.color("magenta")

t.circle(100)

t.ht()



**6. Ellipse**

import turtle

turtle.title("Ellipse")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(3)

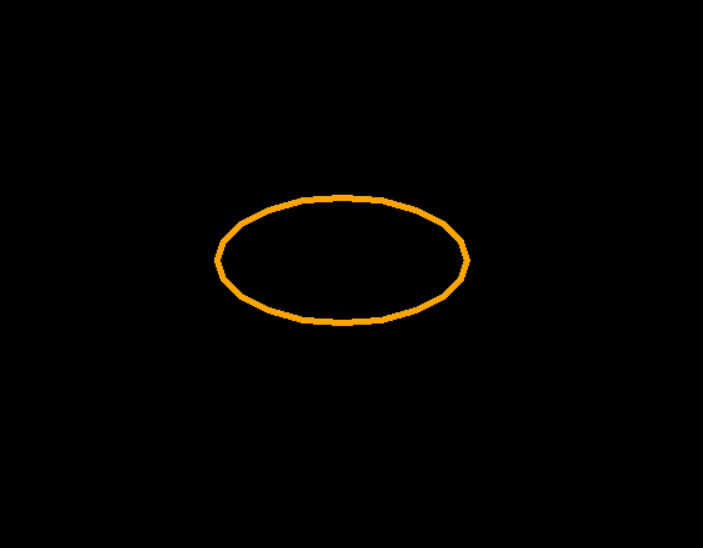
t.color("orange","black")

t.shape("circle")

t.shapesize(5,10,5)

t.stamp()

t.ht()



**7. Rotation**

import turtle

turtle.title("Rotating Object")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(10)

t.color("green")

def drawObject():

for i in range(10):

t.forward(143)

t.right(144)

def rotate(deg):

t.right(deg)

while True:

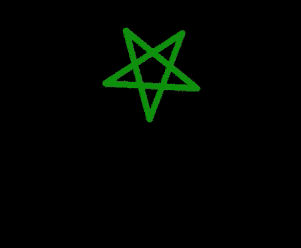
t.hideturtle()

t.clear()

drawObject()

rotate(35)

screen.delay(1)



**8. 2D Object Filling**

import turtle

turtle.title("2D Object Filling")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.width(3)

t.color("chocolate")

t.begin\_fill()

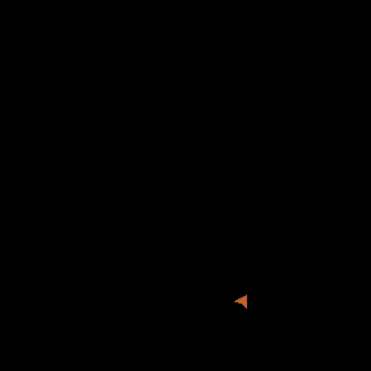
for i in range(6):

t.left(60)

t.forward(100)

t.end\_fill()

t.ht()



**9. 2D Object Transformations**

from time import sleep

import turtle

turtle.title("2D Object Transformations")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.color("red")

t.width(2)

t.goto(-400,0)

t.goto(400,0)

t.home()

t.goto(0,-400)

t.goto(0,400)

t.home()

t.color("blue")

pen = turtle.Turtle()

pen.color("blue")

pen.penup()

pen.ht()

pen.goto(0,250)

pen.write("Transformation on Square",align="center",font=("Verdana", 20, "underline"))

t.shape("square")

sleep(2.5)

pen.goto(-300,200)

pen.write("-> Rotation",font=("Verdana", 20, "normal"))

sleep(0.5)

t.tilt(45)

sleep(2)

pen.goto(-300,160)

pen.write("-> Translation",font=("Verdana", 20, "normal"))

t.penup()

t.goto(150,-150)

t.pendown()

sleep(2)

x = t.clone()

pen.goto(-300,120)

pen.write("-> Scaling",font=("Verdana", 20, "normal"))

t.shapesize(3,2)

sleep(2)

x.shapesize(3,2)

t.penup()

t.goto(150,150)

t.pendown()

pen.goto(-300,80)

pen.write("-> Reflection",font=("Verdana", 20, "normal"))

t.tilt(90)

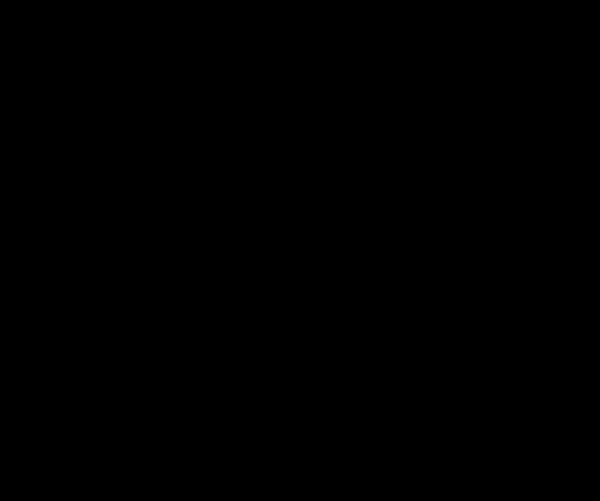
sleep(2)

x.ht()

pen.goto(-300,40)

pen.write("-> Shearing",font=("Verdana", 20, "normal"))

t.shearfactor(-4.5)



**10. Composition of Two Rotations**

import turtle

turtle.title("Composition of Two Rotations")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(10)

t.color("red")

tt = t.clone()

tt.color("blue")

def drawObject():

for i in range(10):

t.forward(143)

tt.forward(143)

t.right(144)

tt.right(144)

def rotate(deg):

t.right(deg)

tt.left(deg+10)

while True:

t.hideturtle()

tt.hideturtle()

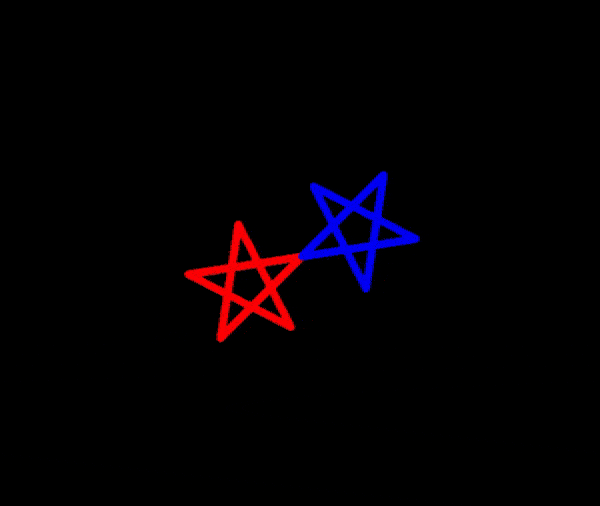
t.clear()

tt.clear()

drawObject()

rotate(45)

screen.delay(1)



from time import sleep

import turtle

turtle.title("Cohen-Sutherland Line Clipping")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

t.width(1)

t.ht()

def write(text):

t.color("deepskyblue")

t.penup()

t.goto(0,200)

t.write(text,align="center",font=("Verdana",20,"normal"))

def drawLine(x1, y1, x2, y2):

t.penup()

t.goto(x1,y1)

t.pendown()

t.goto(x2,y2)

write("Before Clipping")

w = t.clone()

INSIDE = 0 #0000

LEFT = 1 #0001

RIGHT = 2 #0010

BOTTOM = 4 #0100

TOP = 8 #1000

w.color("white")

w.penup()

w.goto(x\_min,y\_min)

**11. Cohen-Sutherland Line Clipping**

w.pendown()

w.goto(x\_min,y\_max)

w.goto(x\_max,y\_max)

w.goto(x\_max,y\_min)

w.goto(x\_min,y\_min)

def computeCode(x, y):

code = INSIDE

if x < x\_min:

code |= LEFT

elif x > x\_max:

code |= RIGHT

if y < y\_min:

code |= BOTTOM

elif y > y\_max:

code |= TOP

return code

def cohenSutherlandClip(x1, y1, x2, y2):

code1 = computeCode(x1, y1)

code2 = computeCode(x2, y2)

accept = False

while True:

if code1 == 0 and code2 == 0:

accept = True

break

elif (code1 & code2) != 0:

break

else:

x, y = 1.0, 1.0

if code1 != 0:

code\_out = code1

else:

code\_out = code2

if code\_out & TOP:

x = x1 + (x2 - x1) \* (y\_max - y1) / (y2 - y1)

y = y\_max

elif code\_out & BOTTOM:

x = x1 + (x2 - x1) \* (y\_min - y1) / (y2 - y1)

y = y\_min

elif code\_out & RIGHT:

y = y1 + (y2 - y1) \* (x\_max - x1) / (x2 - x1)

x = x\_max

elif code\_out & LEFT:

y = y1 + (y2 - y1) \* (x\_min - x1) / (x2 - x1)

x = x\_min

if code\_out == code1:

x1,y1 = x,y

code1 = computeCode(x1,y1)

else:

x2,y2 = x,y

code2 = computeCode(x2, y2)

if accept:

drawLine(x1,y1,x2,y2)

else:

pass

t.color("red")

coords = [ (-150,-130,-200,50),

(-150,-20,-50,180),

(0,-50,-50,50),

(50,20,100,-180),

(80,100,200,100) ]

for point in coords:

drawLine(point[0], point[1], point[2], point[3])

sleep(4)

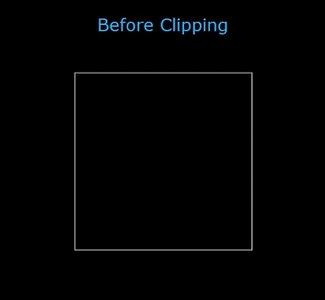
t.clear()

write("After Clipping")

t.color("red")

for point in coords:

cohenSutherlandClip(point[0], point[1], point[2], point[3])



from time import sleep

import turtle

turtle.title("Sutherland-Hodgman Polygon Clipping")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

def clip(subjectPolygon, clipPolygon):

def inside(p):

return(cp2[0]-cp1[0])\*(p[1]-cp1[1]) > (cp2[1]-cp1[1])\*(p[0]-cp1[0])

def computeIntersection():

dc = [ cp1[0] - cp2[0], cp1[1] - cp2[1] ]

dp = [ s[0] - e[0], s[1] - e[1] ]

n1 = cp1[0] \* cp2[1] - cp1[1] \* cp2[0]

n2 = s[0] \* e[1] - s[1] \* e[0]

n3 = 1.0 / (dc[0] \* dp[1] - dc[1] \* dp[0])

return [(n1\*dp[0] - n2\*dc[0]) \* n3, (n1\*dp[1] - n2\*dc[1]) \* n3]

outputList = subjectPolygon

cp1 = clipPolygon[-1]

for clipVertex in clipPolygon:

cp2 = clipVertex

inputList = outputList

outputList = []

s = inputList[-1]

**12. Sutherland-Hodgman Polygon Clipping**

for subjectVertex in inputList:

e = subjectVertex

if inside(e):

if not inside(s):

outputList.append(computeIntersection())

outputList.append(e)

elif inside(s):

outputList.append(computeIntersection())

s = e

cp1 = cp2

return(outputList)

def goto(p):

t.penup()

t.goto(p)

t.pendown()

def write(text):

t.color("deepskyblue")

t.penup()

t.goto(0,180)

t.write(text,align="center",font=("Verdana",20,"normal"))

def draw(points,col):

t.color(col)

goto(points[0])

for point in points:

t.goto(point)

t.goto(points[0])

polygon = [ (-150,0), (0,-120), (150,0),

(50,150), (0,50), (-50,150)]

clipWindow = [ (-100,-100), (100,-100), (100,100), (-100,100) ]

write("Before Clipping")

t.begin\_fill()

draw(polygon,"red")

t.end\_fill()

draw(clipWindow,"white")

clippedPoints = clip(polygon,clipWindow)

clippedResult = [tuple(p) for p in clippedPoints]

sleep(4)

t.clear()

write("After Clipping")

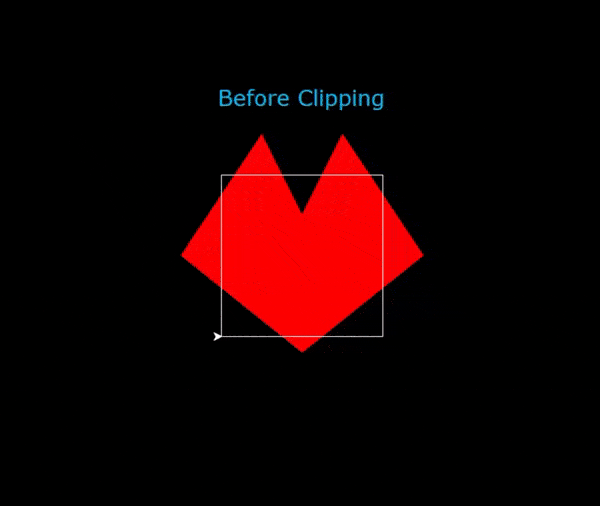
t.begin\_fill()

draw(clippedResult,"red")

t.end\_fill()

draw(clipWindow,"white")

t.ht()



**13. 3D Object**

import turtle

turtle.title("3D Object")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

top = (20,180)

# Right side

t.color("darkgoldenrod")

t.begin\_fill()

t.left(10)

t.forward(100)

t.goto(top)

t.home()

t.end\_fill()

# Left side

t.color("goldenrod")

t.begin\_fill()

t.left(160)

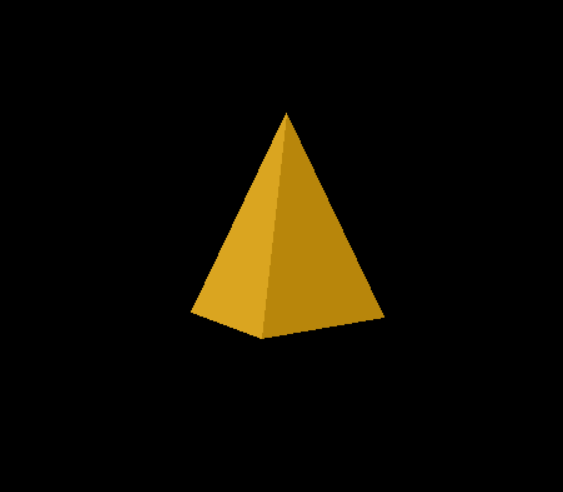
t.forward(60)

t.goto(top)

t.home()

t.end\_fill()

t.ht()



**14. 3D Object Filling**

import turtle

turtle.title("3D Object Filling")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.speed("fastest")

d = 100

def move():

t.forward(d)

# Right side

t.color("grey")

t.begin\_fill()

t.left(45)

move()

t.right(135)

move()

t.right(45)

move()

t.right(135)

move()

t.end\_fill()

# Left side

t.color("darkgrey")

t.begin\_fill()

t.left(45)

move()

t.left(135)

move()

t.left(45)

move()

t.left(135)

move()

t.end\_fill()

# Top side

t.color("lightgrey")

t.begin\_fill()

t.left(45)

move()

t.right(90)

move()

t.right(90)

move()

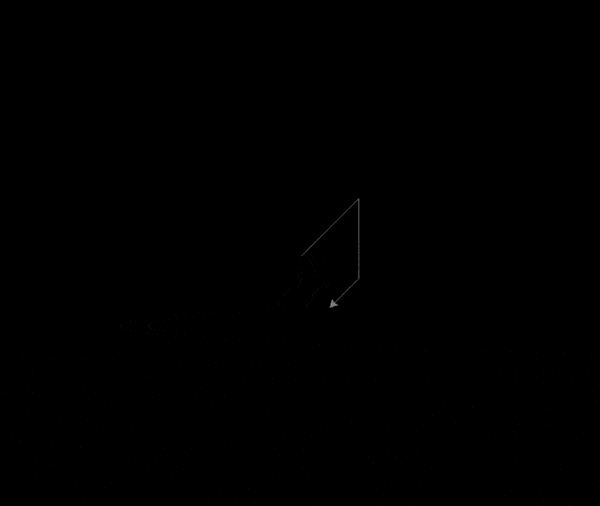
t.right(90)

move()

t.right(135)

move()

t.end\_fill()



**15. 3D Object Transformations**

from time import sleep

import turtle

turtle.title("3D Transformations")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.color("white","green")

t.width(3)

t.speed("fastest")

t.ht()

def cone(r):

t.pendown()

t.begin\_fill()

t.left(45)

t.circle(-r,90)

t.right(90)

t.circle(-r,90)

t.end\_fill()

t.right(67.5)

t.forward(3\*\*0.5\*r)

t.right(132.5)

t.forward(3\*\*0.5\*r)

t.penup()

def translate(x,y):

t.penup()

t.goto(x,y)

t.pendown()

def scale(val):

cone(r\*val)

def rotate(deg):

t.left(deg)

cone(r)

def write(x,y,text):

pen.penup()

pen.goto(x,y)

pen.pendown()

pen.write("-> "+text,font=("Verdana", 20, "normal"))

pen = t.clone()

pen.color("red")

r = 60

pen.penup()

pen.goto(0,250)

pen.pendown()

pen.write("Transformation on Cone",align="center",font=("Verdana", 20, "underline"))

t.penup()

t.goto(-r\*0.66,0)

t.pendown()

cone(r)

pen.penup()

pen.home()

pen.pendown()

pen.width(1)

pen.goto(-220,0)

pen.home()

pen.goto(0,220)

pen.home()

pen.goto(100,-100)

pen.home()

sleep(2)

t.home()

write(-300,200,"Translation")

translate(-210,70)

cone(r)

sleep(2)

t.home()

write(50,200,"Scaling")

translate(100,45)

scale(1.5)

sleep(2)

t.home()

write(50,-220,"Rotation")

translate(150,-150)

rotate(45)

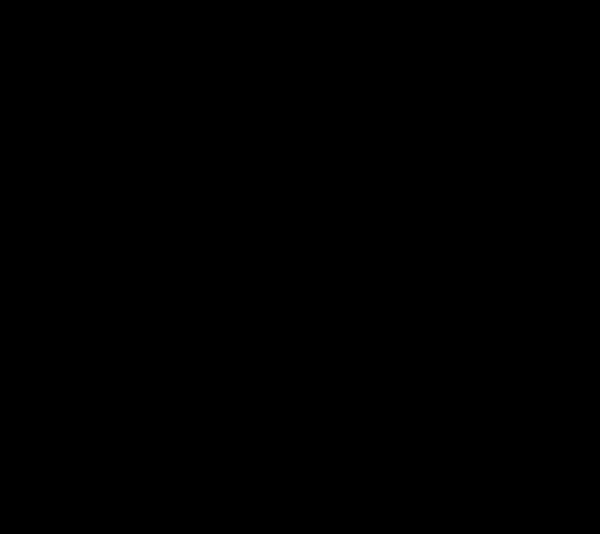
sleep(2)

t.home()

write(-300,-220,"Reflection")

translate(-125,-70)

rotate(180)



**16. Cubic Bezier Curve**

import turtle

turtle.title("Cubic Bezier Curve")

screen = turtle.Screen()

screen.bgcolor("black")

b = turtle.Turtle()

b.color("red")

b.penup()

b.ht()

def bx(t):

bxcor = ( c1x \* (1-t)\*\*3 + 3 \* c2x \* t \* (1-t)\*\*2

+ 3 \* c3x \* t\*\*2 \* (1-t) + c4x \* t\*\*3 )

return bxcor

def by(t):

bycor = ( c1y \* (1-t)\*\*3 + 3 \* c2y \* t \* (1-t)\*\*2

+ 3 \* c3y \* t\*\*2 \* (1-t) + c4y \* t\*\*3 )

return bycor

controlPoints = [ (-170,-150), (-40,130), (90,-150), (170,140) ]

for point in controlPoints:

b.goto(point)

b.dot()

c1x, c1y = controlPoints[0]

c2x, c2y = controlPoints[1]

c3x, c3y = controlPoints[2]

c4x, c4y = controlPoints[3]

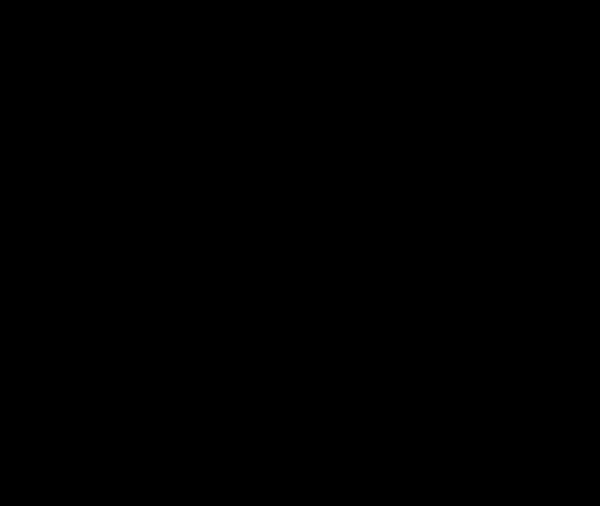
b.goto(c1x,c1y)

b.pendown()

for i in range(1001):

t = i/1000.0

b.goto(bx(t),by(t))



**17. Cylinder**

import turtle

turtle.title("Cylinder")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.color("blue")

t.fillcolor("#000111")

t.width(5)

t.penup()

t.ht()

def drawEllipse():

t.right(45)

t.circle(-150,90)

t.right(90)

t.circle(-150,90)

def move():

t.right(90)

t.circle(-150,90)

t.right(45)

def drawLine():

t.forward(300)

def drawCylinder():

t.pendown()

t.begin\_fill()

drawLine()

drawEllipse()

move()

drawLine()

drawEllipse()

t.end\_fill()

t.forward(100)

t.left(90)

t.forward(150)

t.left(180)

drawCylinder()

t.color("grey")

t.width(1)

t.penup()

t.home()

t.goto(100,150)

t.left(180)

t.pendown()

t.forward(100)

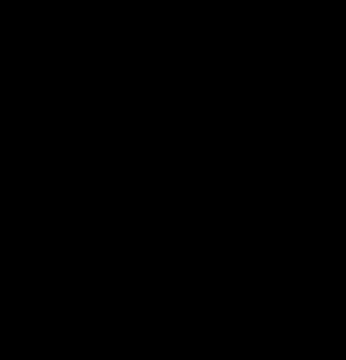
t.write(" r = 100 units")

t.left(90)

t.forward(150)

t.write(" h = 300 units")

t.forward(150)



**18. Parallel Projection**

import turtle

turtle.title("Parallel Projection")

screen = turtle.Screen()

screen.bgcolor("black")

t = turtle.Turtle()

t.width(3)

t.speed("fastest")

t.color("#121212","white")

def dot(tt):

tt.pendown()

tt.dot()

tt.penup()

def goto(tt,x,y):

tt.penup()

tt.goto(x,y)

tt.pendown()

def drawSquare(tt):

tt.begin\_fill()

for i in range(4):

tt.left(90)

tt.forward(50)

tt.end\_fill()

def drawArrow():

pen.left(135)

pen.forward(20)

pen.backward(20)

pen.left(90)

pen.forward(20)

t.left(45)

t.begin\_fill()

t.forward(50)

for i in range(3):

t.left(90)

t.forward(50)

t.right(45)

t.forward(50)

t.left(135)

t.forward(50)

t.left(45)

t.forward(50)

t.left(135)

t.forward(50)

t.right(90)

t.forward(50)

t.left(135)

t.forward(50)

t.left(45)

t.forward(50)

t.end\_fill()

t.left(112.5)

t.penup()

t.forward(47)

dot(t)

t.backward(47)

t.left(45)

for i in range(2):

t.forward(22.5)

dot(t)

t.forward(22.5)

t.backward(90)

t.right(22.5)

t.forward(50)

t.right(45)

t.forward(50)

t.left(135)

for i in range(3):

t.forward(18)

dot(t)

t.home()

t.ht()

pen = t.clone()

pen.width(4)

pen.color("white")

goto(pen,180,-220)

pen.write("Front View",align="center",font=("Verdana",20,"normal"))

front = t.clone()

goto(front,200,-160)

drawSquare(front)

front.penup()

front.left(135)

front.forward(35.36)

dot(front)

pen.left(135)

goto(pen,120,-120)

pen.goto(40,-40)

drawArrow()

goto(pen,0,240)

pen.write("Top View",align="center",font=("Verdana",20,"normal"))

top = t.clone()

goto(top,20,160)

drawSquare(top)

top.penup()

top.left(135)

for i in range(3):

top.forward(18)

dot(top)

pen.right(90)

goto(pen,0,140)

pen.goto(0,80)

drawArrow()

goto(pen,-180,-220)

pen.write("Side View",align="center",font=("Verdana",20,"normal"))

side = t.clone()

goto(side,-160,-160)

drawSquare(side)

side.penup()

side.left(135)

for i in range(2):

side.forward(18)

dot(side)

side.forward(18)

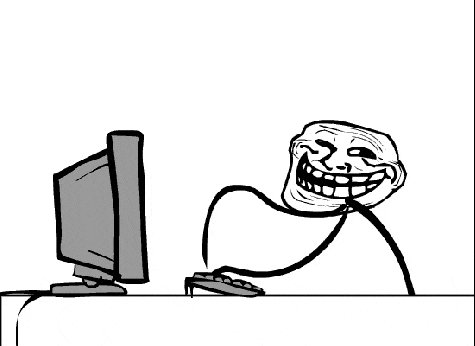
pen.right(90)

goto(pen,-120,-120)

pen.goto(-40,-40)

drawArrow()





[](https://dileepdodla.github.io/CG/)