# Самостоятельная работа № 2. Координатный метод

## Задача

1. Разработать математическую модель и программную реализацию задачи по вычерчиванию произвольного числа треугольников, располагаемых внутри друг друга.
2. Разработайте программу для вычерчивания следующей последовательности отрезков прямых линий:

* из точки (1.0, 6.0) в точку (1.0, 1.0);
* из точки (1.0, 5.8) в точку (1.2, 1.0);
* из точки (1.0, 5.6) в точку (1.4, 1.0);
* ...
* из точки (1.0, 1.0) в точку (6.0, 1.0);

1. Разработать математическую модель и программную реализацию задачи по вычерчиванию 30 стрелок на окружности с центром в (x,y).
2. Решение задачи на отображение.
   1. Дан треугольник с координатами вершин. Смоделируйте вращение вокруг линии .
   2. Дан треугольник с координатами вершин. Смоделируйте вращение вокруг линии .

## Код программы

WD = 500  
ZR = WD / 2  
KF = 10  
MAIN\_TR = [(-15, -10), (0, 15), (15, -10)]  
  
  
def dec\_to\_scr(cords: tuple):  
 return cords[0] \* KF + ZR, ZR - cords[1] \* KF  
  
  
def axis():  
 cnvs.create\_line(0, 250, 500, 250, arrow=LAST)  
 cnvs.create\_line(250, 500, 250, 0, arrow=LAST)  
 cnvs.create\_text(WD - 20, ZR + 20, text='X')  
 cnvs.create\_text(ZR - 20, 20, text='Y', )  
 vals = [i for i in range(-50, 51, 5)]  
 cords = [int(ZR + i\*KF) for i in vals]  
 for i in range(len(cords)):  
 cnvs.create\_oval(cords[i], ZR, cords[i] + 2, ZR + 2, fill='black')  
 cnvs.create\_text(cords[i] + 5, ZR + 10, text=str(vals[i]))  
 cnvs.create\_oval(ZR, cords[i], ZR + 2, cords[i] + 2, fill='black')  
 cnvs.create\_text(ZR + 5, cords[i] + 10, text=str(-vals[i]))  
  
  
def divide(lst\_of\_tp, i):  
 new\_lst = list()  
 for each in lst\_of\_tp:  
 new\_tp = (each[0] / i, each[1] / i)  
 new\_lst.append(new\_tp)  
 return new\_lst  
  
  
def draw():  
 cnvs.delete('all')  
 n = int(amount.get())  
 for i in range(1, n + 1):  
 new\_triangle = divide(MAIN\_TR, i)  
 new\_triangle\_scr = [dec\_to\_scr(i) for i in new\_triangle]  
 cnvs.create\_polygon(new\_triangle\_scr, fill='white', outline='black')  
 axis()

WD = 500  
ZR = WD / 2  
KF = 25  
  
  
def dec\_to\_scr(cords):  
 return cords[0] \* KF + ZR, ZR - cords[1] \* KF  
  
  
def axis():  
 canv.create\_line(0, 250, 500, 250, arrow=LAST)  
 canv.create\_line(250, 500, 250, 0, arrow=LAST)  
 canv.create\_text(WD-20, ZR+20, text='X')  
 canv.create\_text(ZR-20, 20, text='Y', )  
 vals = [i for i in range(-10, 10)]  
 cords = [int(ZR + i\*KF) for i in vals]  
 for i in range(len(cords)):  
 canv.create\_oval(cords[i], ZR, cords[i]+2, ZR+2, fill='black')  
 canv.create\_text(cords[i]+5, ZR+10, text=str(vals[i]))  
 canv.create\_oval(ZR, cords[i], ZR+2, cords[i]+2, fill='black')  
 canv.create\_text(ZR+5, cords[i]+10, text=str(-vals[i]))  
  
  
def draw():  
 p1 = [1.0, 6.0]  
 p2 = [1.0, 1.0]  
 while p1[1] > 1.0:  
 canv.create\_line(dec\_to\_scr(p1), dec\_to\_scr(p2))  
 p1[1] -= 0.2  
 p2[0] += 0.2

WD = 500  
ZR = WD / 2  
KF = 25  
DIAM = 10  
RAD = DIAM/2  
LENGTH = 3  
  
  
def dec\_to\_scr(cords):  
 return round(cords[0] \* KF + ZR), round(ZR - cords[1] \* KF)  
  
  
def atrd(x):  
 return x \* math.pi / 180  
  
  
def axis():  
 cnvs.create\_line(0, 250, 500, 250, arrow=LAST)  
 cnvs.create\_line(250, 500, 250, 0, arrow=LAST)  
 cnvs.create\_text(WD - 20, ZR + 20, text='X')  
 cnvs.create\_text(ZR - 20, 20, text='Y', )  
 vals = [i for i in range(-10, 10)]  
 cords = [int(ZR + i\*KF) for i in vals]  
 for i in range(len(cords)):  
 cnvs.create\_oval(cords[i], ZR, cords[i] + 2, ZR + 2, fill='black')  
 cnvs.create\_text(cords[i] + 5, ZR + 10, text=str(vals[i]))  
 cnvs.create\_oval(ZR, cords[i], ZR + 2, cords[i] + 2, fill='black')  
 cnvs.create\_text(ZR + 5, cords[i] + 10, text=str(-vals[i]))  
  
  
def draw():  
 cnvs.delete('all')  
 x = int(x\_dec.get())  
 y = -int(y\_dec.get())  
 angle = 0  
 x0 = RAD \* math.cos(angle) + x  
 y0 = RAD \* math.sin(angle) - y  
 p1 = (x0, y0)  
 for i in range(31):  
 ang = angle + 2 \* math.pi \* i / 30  
 xi = RAD \* math.cos(ang) + x  
 yi = RAD \* math.sin(ang) - y  
 p2 = (xi, yi)  
 print(p1, p2)  
 cnvs.create\_line(dec\_to\_scr(p1), dec\_to\_scr(p2), arrow=LAST)  
 p1 = p2  
 axis()

WD = 500  
ZR = WD / 2  
KF = 10  
TRIANG = [(-5, 5), (0, 15), (5, 5)]  
DIAM = 10  
RAD = DIAM/2  
  
  
def dec\_to\_scr(cords: tuple):  
 return round(cords[0] \* KF + ZR), round(ZR - cords[1] \* KF)  
  
  
def axis():  
 cnvs.create\_line(0, 250, 500, 250, arrow=LAST)  
 cnvs.create\_line(250, 500, 250, 0, arrow=LAST)  
 cnvs.create\_line(0, 500, 500, 0, arrow=LAST)  
 cnvs.create\_text(WD - 20, ZR + 20, text='X')  
 cnvs.create\_text(ZR - 20, 20, text='Y', )  
 vals = [i for i in range(-50, 51, 5)]  
 cords = [int(ZR + i\*KF) for i in vals]  
 for i in range(len(cords)):  
 cnvs.create\_oval(cords[i], ZR, cords[i] + 2, ZR + 2, fill='black')  
 cnvs.create\_text(cords[i] + 5, ZR + 10, text=str(vals[i]))  
 cnvs.create\_oval(ZR, cords[i], ZR + 2, cords[i] + 2, fill='black')  
 cnvs.create\_text(ZR + 5, cords[i] + 10, text=str(-vals[i]))  
  
  
def transl\_to\_rad(x):  
 return x \* math.pi / 180  
  
  
def turner(lst\_of\_tp, ang):  
 new\_lst = list()  
 for each in lst\_of\_tp:  
 new\_tp = (each[0]\*math.cos(ang), each[1]\*math.sin(ang))  
 new\_lst.append(new\_tp)  
 return new\_lst  
  
  
def draw1():  
 cnvs.delete('all')  
 angle = transl\_to\_rad(int(angle1.get()))  
 matrix = ((math.cos(angle), math.sin(angle)),  
 (-math.sin(angle), math.cos(angle)))  
  
 new\_triangle = list()  
 for each in TRIANG:  
 new\_cords = (each[0]\*matrix[0][0] + each[1]\*matrix[1][0],  
 each[0]\*matrix[0][1] + each[1]\*matrix[1][1])  
 new\_triangle.append(new\_cords)  
 new\_triangle\_scr = [dec\_to\_scr(i) for i in new\_triangle]  
 cnvs.create\_polygon(\*new\_triangle\_scr, fill='white', outline='black')  
 axis()  
  
  
def draw2():  
 cnvs.delete('all')  
 angle = transl\_to\_rad(int(angle2.get()))  
 angle45 = transl\_to\_rad(45)  
 matrix = ((math.cos(angle), math.sin(angle)),  
 (-math.sin(angle), math.cos(angle)))  
 matrix45 = ((math.cos(angle45), math.sin(angle45)),  
 (-math.sin(angle45), math.cos(angle45)))  
 new\_triangle1 = list()  
 for each in TRIANG:  
 new\_cords1 = (each[0]\*matrix[0][0] + each[1]\*matrix[1][0],  
 each[0]\*matrix[0][1] + each[1]\*matrix[1][1])  
 new\_triangle1.append(new\_cords1)  
 new\_triangle45 = list()  
 for each in new\_triangle1:  
 new\_cords45 = (each[0] \* matrix45[0][0] + each[1] \* matrix45[1][0],  
 each[0] \* matrix45[0][1] + each[1] \* matrix45[1][1])  
 new\_triangle45.append(new\_cords45)  
 new\_triangle45\_scr = [dec\_to\_scr(i) for i in new\_triangle45]  
 cnvs.create\_polygon(\*new\_triangle45\_scr, fill='white', outline='black')  
 axis()

## Результаты

 



 

 

 