**Ссылка на GitHub:** [**https://github.com/DenisNyux/computer\_graphics**](https://github.com/DenisNyux/computer_graphics)

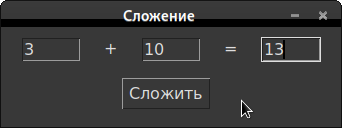
**Язык программирования:** Python 3.7 + библиотека tkinter

Лабораторная работа №1

**Тема:** Знакомство со средой разработки

**Условие:** Написать процедуру обработки события - щелчок по кнопке - сложение двух чисел. Для этого необходимо создать процедуру обработки события (двойным щелчком по объекту кнопка), объявить переменные, ввести значение переменных a и b, вычислить значение переменной с и вывести значение с на экран.

**Скриншот:**



**Код в процедуре:**

**def adder\_click():**

**a = int(entering\_number1.get())**

**b = int(entering\_number2.get())**

**res = str(a+b)**

**result.delete(0, END)**

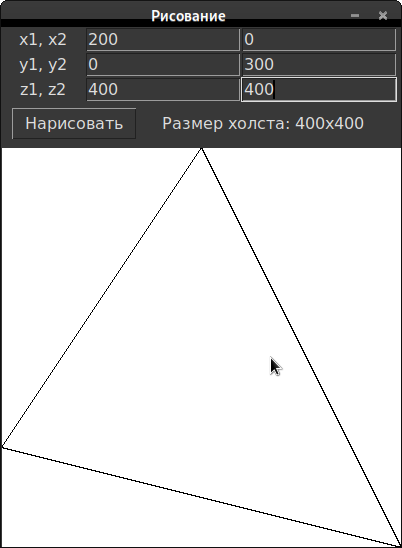
**result.insert(0, res)**

Лабораторная работа №2

**Тема:** Работа с графикой

**Условие:** Создать программу рисующую треугольник, координаты вершин которого можно изменять через пользовательский интерфейс.

**Скриншот:**

****

**Код процедуры:**

**def draw():**

**c.delete("all")**

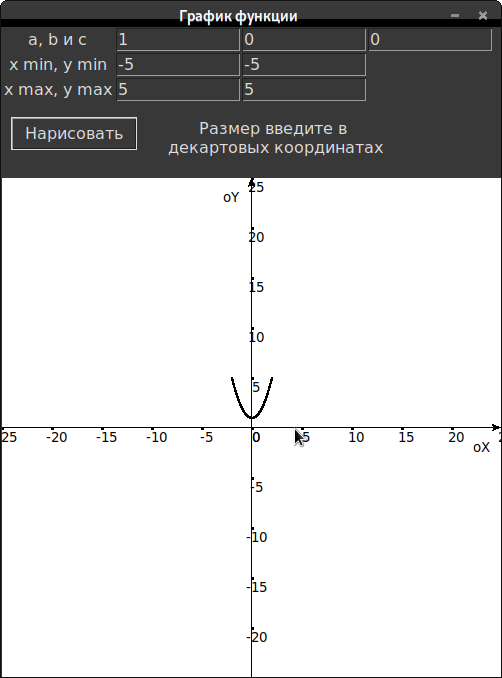
**c.create\_polygon(int(x1.get()), int(x2.get()), int(y1.get()), int(y2.get()), int(z1.get()), int(z2.get()), fill='white', outline='black')**

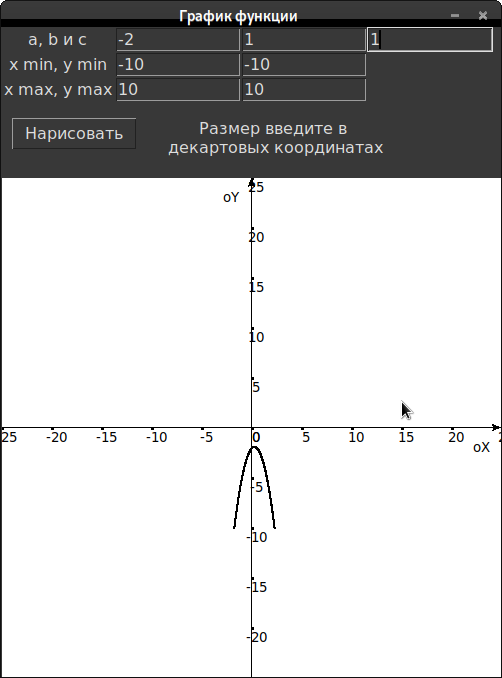
Лабораторная работа №3

**Тема:** Построение графика заданной функции в указанном диапазоне

**Условие:** Разработать алгоритм и программу решения прикладной задачи – построение графика заданной функции. Научиться переводить декартовые координаты в экранные. Построить график функции y = Ax^2 + Bx + C в диапазоне ( x min , y min ) – ( x max , y max ). Постоянные А, В, С, x min , y min , x max , y max вводятся через пользовательский интерфейс.

**Скриншот:**





**Код процедуры рисования графика и осей:**

WD = 500

HG = 500

ZR = 250

KF = 10

def func(x):

a = int(a\_const.get())

b = int(b\_const.get())

c = int(a\_const.get())

return a\*x\*x + b\*x + c

def axis\_descr():

canv.create\_text(WD-20, ZR+20, text='oX')

canv.create\_text(ZR-20, 20, text='oY', )

vals = [i for i in range(-50, 51, 5)]

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():

canv.delete("all")

canv.create\_line(0, 250, 500, 250, arrow=LAST)

canv.create\_line(250, 500, 250, 0, arrow=LAST)

axis\_descr()

x\_mn = int(x\_min.get())

x\_mx = int(x\_max.get())

y\_mn = int(y\_min.get())

y\_mx = int(y\_max.get())

x = x\_mn

while x <= x\_mx:

xc = round(ZR + x\*KF)

yc = round(ZR - func(x)\*KF)

if ZR - y\_mn\*KF >= yc >= ZR - y\_mx\*KF:

canv.create\_oval(xc, yc, xc+1, yc+1)

x += 1/100

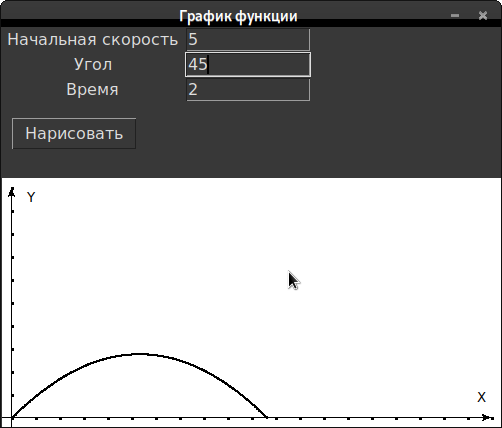
Лабораторная работа №4

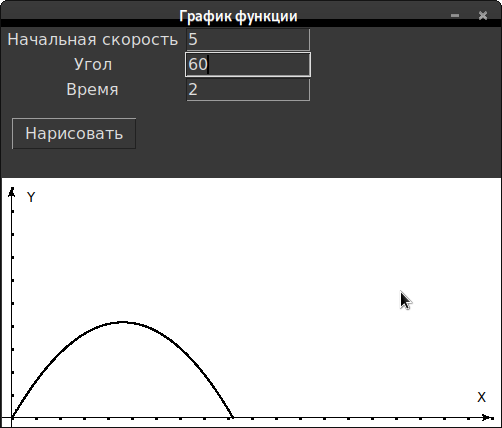
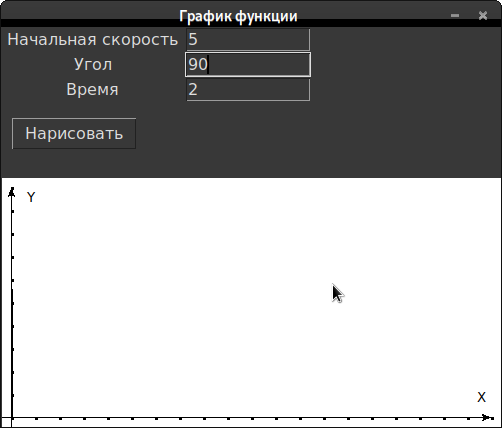
**Тема:** Движение тела брошенного под углом к горизонту

**Условие:** Разработать алгоритм и программу решения прикладной задачи – построение траектории движения тела брошенного под углом к горизонту. Научиться переводить мировые координаты в экранные.

Изобразить местонахождение тела, брошенного с земли с начальной скорость U0 под углом к горизонту α, через равные интервалы времени ( N – число равных временных интервалов)

**Скриншот:**

****

****

**Код процедур:**

WD = 500

HG = 250

ZR\_X = 10

ZR\_Y = 240

g = 9.8

def draw\_axis():

canv.create\_text(WD - 20, ZR\_Y - 20, text='X')

canv.create\_text(ZR\_X + 20, 20, text='Y')

x\_vals = [i/10 for i in range(21)]

y\_vals = [i/10 for i in range(11)]

x\_cords = [ZR\_X+x\_vals[i]\*240 for i in range(len(x\_vals))]

y\_cords = [ZR\_Y-y\_vals[i]\*230 for i in range(len(y\_vals))]

for i in range(len(x\_cords)):

canv.create\_oval(x\_cords[i], ZR\_Y, x\_cords[i] + 2, ZR\_Y + 2, fill='black')

for j in range(len(y\_cords)):

canv.create\_oval(ZR\_X, y\_cords[j], ZR\_X + 2, y\_cords[j] + 2, fill='black')

def x\_func(t):

v = int(v0.get())

ang = int(angle.get()) \* math.pi/180

return v \* t \* math.cos(ang)

def y\_func(t):

v = int(v0.get())

ang = int(angle.get()) \* math.pi/180

return v \* t \* math.sin(ang) - g \* t \* t / 2

def draw():

canv.delete("all")

canv.create\_line(0, 240, 490, 240, arrow=LAST)

canv.create\_line(10, 250, 10, 10, arrow=LAST)

draw\_axis()

tm = float(time.get())

i = 0

xc = 0

yc = 0

while i <= tm:

xc = round(ZR\_X + x\_func(i)\*100)

yc = round(ZR\_Y - y\_func(i)\*100)

if yc <= 240:

canv.create\_oval(xc, yc, xc + 1, yc + 1)

i += tm/1000

print(xc)

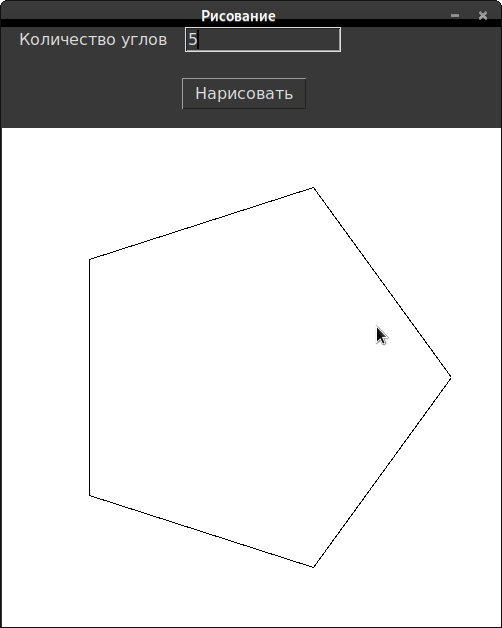
print(yc)

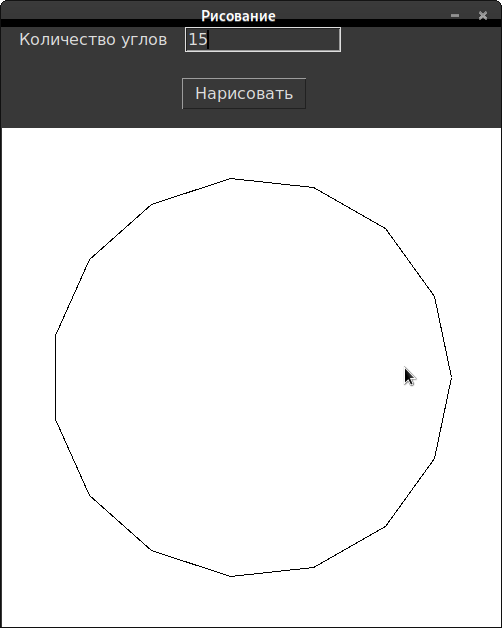
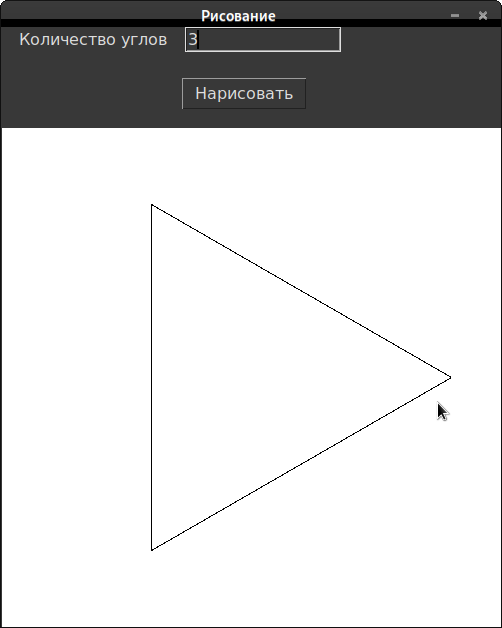
Лабораторная работа №5

**Тема:** Построение правильного n-угольника

**Условие:** Построить правильный n -угольник, количество углов которого задано и вводиться через пользовательский интерфейс.

**Скриншоты:**

****

****

Лабораторная работа №6

**Тема:** Преобразование объекта на плоскости

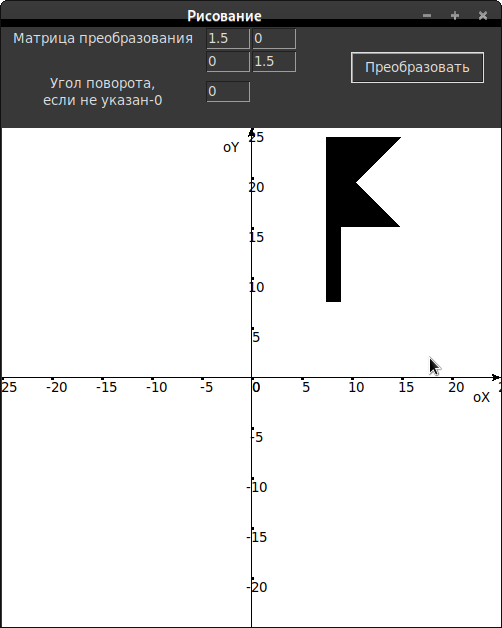
**Условие:** Построить произвольный объект в первой четверти системы координат (например, флажок. кол-во вершин объекта должно быть на меньше 7) и организовать следующие преобразования объекта:

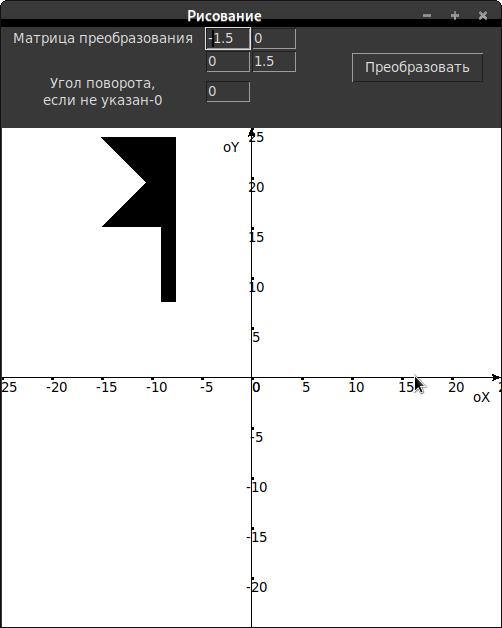
- общее преобразование (изменение масштаба, симметричное отражение относительно осей, сдвиг),

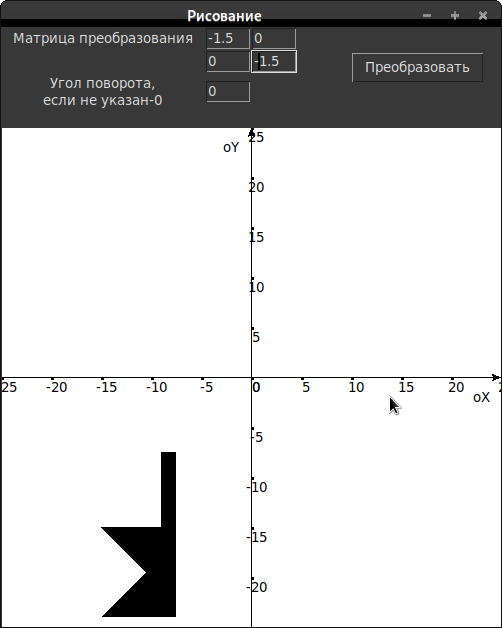
- вращение объекта относительно начала координат

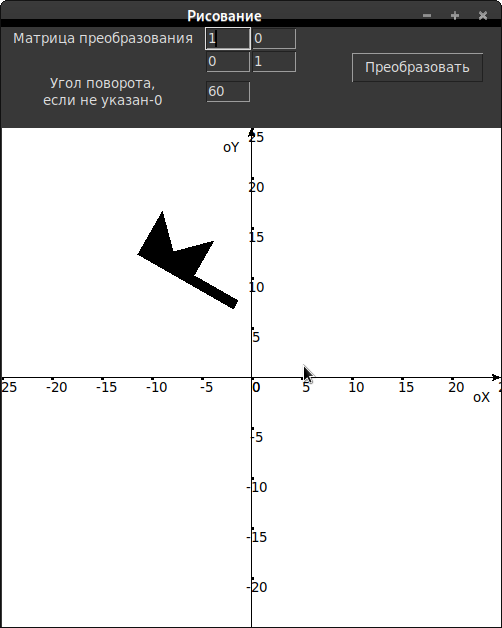
- комбинированное преобразование (преобразование общего вида + вращение на произвольный угол относительно начала координат)

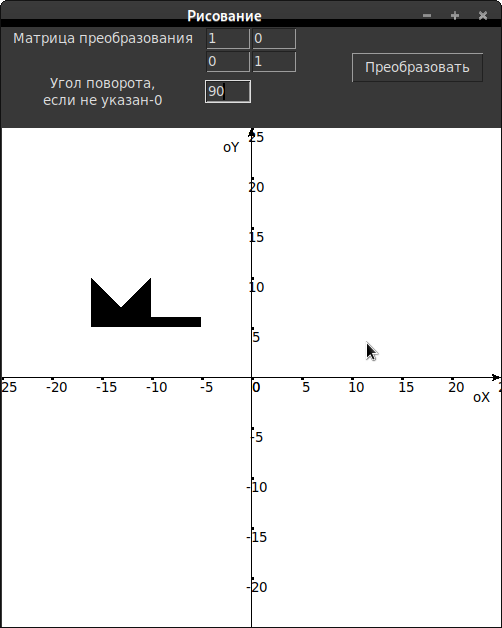
**Скриншоты:**

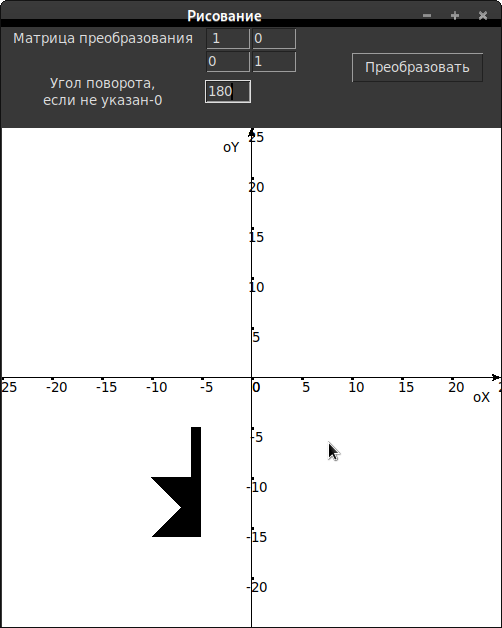
****

****

****

****

****



**Код программы:**

WD = 500

ZR = WD / 2

KF = 10

FLAG = [(5, 5), (5, 16), (10, 16), (7, 13), (10, 10), (6, 10), (6, 5)]

def dec\_to\_scr(cords: tuple):

return cords[0] \* KF + ZR, ZR - cords[1] \* KF

def axis():

canv.delete("all")

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='oX')

canv.create\_text(ZR-20, 20, text='oY', )

vals = [i for i in range(-50, 51, 5)]

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 atrd(x):

return x \* math.pi / 180

def transfer():

# Получаем матрицу преобразования

axis()

if int(angle.get()) == 0:

matrix = ((float(a11.get()), float(a12.get())),

(float(a21.get()), float(a22.get())))

else:

matrix = ((math.cos(atrd(int(angle.get()))), math.sin(atrd(int(angle.get())))),

(-math.sin(atrd(int(angle.get()))), math.cos(atrd(int(angle.get())))))

new\_flag = list()

for each in FLAG:

new\_cords = (each[0]\*matrix[0][0] + each[1]\*matrix[1][0],

each[0]\*matrix[0][1] + each[1]\*matrix[1][1])

new\_flag.append(new\_cords)

new\_flag\_scr = [dec\_to\_scr(i) for i in new\_flag]

canv.create\_polygon(new\_flag\_scr)