***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования***

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

**Факультет ИУ**

**Кафедра ИУ10**

**Отчет**

**по лабораторной работе № 7**

**Дисциплина: Информатика**

**Тема: «ООП»**

Работу выполнил: Андреев Александр Вадимович ИУ10-23

Проверил: Буркацкий Кирилл Александрович

Москва, 2023

**Цель работы:** провести модернизацию и рефакторинг кода из лабораторной работы №6.

# **Ход работы**

## Задача №1

**Цель**: модернизация кода

### Ход решения задачи

1. Написание программы:

import math

import time

from random import randrange as rnd, choice

from tkinter import \*

root = Tk()

fr = Frame(root)

root.geometry('800x600')

canv = Canvas(root, bg='white')

canv.pack(fill=BOTH, expand=1)

class ball():

    def \_\_init\_\_(self, x=40, y=450):

        self.x = x

        self.y = y

        self.r = 10

        self.vx = 0

        self.vy = 0

        self.color = choice(['blue', 'green', 'red', 'brown'])

        if k % 2 == 0:

            self.id = canv.create\_oval(self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r,

                                       fill=self.color)

        else:

            self.id = canv.create\_rectangle(self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r,

                                            fill=self.color)

        self.live = 30

    def set\_coords(self):

        canv.coords(self.id, self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r)

    def move(self):

        if self.y <= 500:

            self.vy -= 1.2

            self.y -= self.vy

            self.x += self.vx

            self.vx \*= 0.99

            self.set\_coords()

        else:

            if self.vx \*\* 2 + self.vy \*\* 2 > 10:

                self.vy = -self.vy / 2

                self.vx = self.vx / 2

                self.y = 499

            if self.live < 0:

                balls.pop(balls.index(self))

                canv.delete(self.id)

            else:

                self.live -= 1

        if self.x > 780:

            self.vx = -self.vx / 2

            self.x = 779

    def hittest(self, ob):

        if abs(ob.x - self.x) <= (self.r + ob.r) and abs(ob.y - self.y) <= (self.r + ob.r):

            return True

        else:

            return False

class gun():

    def \_\_init\_\_(self, x):

        self.f2\_power = 10

        self.f2\_on = 0

        self.an = 1

        self.x = x

        self.speed = 5

        self.id = canv.create\_line(self.x, 450, self.x + 30, 420, width=7)

    def fire2\_start(self, event):

        self.f2\_on = 1

    def fire2\_end(self, event):

        global balls, bullet, k

        bullet += 1

        new\_ball = ball(self.x + 20, 450)

        new\_ball.r += 5

        self.an = math.atan((event.y - new\_ball.y) / (event.x - new\_ball.x))

        if event.x < new\_ball.x:

            self.an += math.pi

        new\_ball.vx = self.f2\_power \* math.cos(self.an)

        new\_ball.vy = -self.f2\_power \* math.sin(self.an)

        balls += [new\_ball]

        self.f2\_on = 0

        self.f2\_power = 10

        canv.bind('<Motion>', '')

        k += 1

    def targetting(self, event=0):

        if event:

            self.an = math.atan((event.y - 450) / (event.x - self.x))

            if event.x < self.x:

                self.an += math.pi

            print(self.an)

        if self.f2\_on:

            canv.itemconfig(self.id, fill='orange')

        else:

            canv.itemconfig(self.id, fill='black')

        canv.coords(self.id, self.x, 450, self.x + max(self.f2\_power, 20) \* math.cos(self.an),

                    450 + max(self.f2\_power, 20) \* math.sin(self.an))

    def moving(self):

        if self.x < 0:

            self.speed = 5

        elif self.x > 800:

            self.speed = -5

        self.x += self.speed

    def power\_up(self):

        if self.f2\_on:

            if self.f2\_power < 100:

                self.f2\_power += 1

            canv.itemconfig(self.id, fill='orange')

        else:

            canv.itemconfig(self.id, fill='black')

class target():

    def \_\_init\_\_(self):

        self.points = 0

        self.speed = 5

        self.sp\_x = rnd(1, 5)

        self.sp\_y = (25 - self.sp\_x \*\* 2) \*\* 0.5

        self.id = canv.create\_oval(0, 0, 0, 0)

        self.id\_points = canv.create\_text(30, 30, text=self.points, font='28')

        self.new\_target()

        self.live = 1

    def new\_target(self):

        x = self.x = rnd(600, 780)

        y = self.y = rnd(300, 550)

        r = self.r = rnd(2, 50)

        color = self.color = 'red'

        canv.coords(self.id, x - r, y - r, x + r, y + r)

        canv.itemconfig(self.id, fill=color)

    def move(self):

        if self.live != 0:

            if self.x < 0 or self.x > 800:

                self.sp\_x = -self.sp\_x

            if self.y < 0 or self.y > 600:

                self.sp\_y = -self.sp\_y

            self.x += self.sp\_x

            self.y += self.sp\_y

            canv.coords(self.id, self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r)

            canv.itemconfig(self.id, fill=self.color)

    def hit(self, points=1):

        canv.coords(self.id, -10, -10, -10, -10)

        self.points += points

        canv.itemconfig(self.id\_points, text=self.points)

class n\_target():

    def \_\_init\_\_(self):

        self.points = 0

        self.sp\_x = 5

        self.id = canv.create\_rectangle(0, 0, 0, 0)

        self.id\_points = canv.create\_text(70, 30, text=self.points, font='28')

        self.new\_target()

        self.live = 1

    def new\_target(self):

        x = self.x = rnd(600, 780)

        y = self.y = rnd(300, 550)

        r = self.r = rnd(2, 50)

        self.live = 1

        color = self.color = 'red'

        canv.coords(self.id, x - r, y - r, x + r, y + r)

        canv.itemconfig(self.id, fill=color)

    def move(self):

        if self.live != 0:

            if 0 < self.x < 800:

                self.x += self.sp\_x

            else:

                self.sp\_x = -self.sp\_x

                self.x += self.sp\_x

            self.y += math.sin(self.x) \* 20

            canv.coords(self.id, self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r)

            canv.itemconfig(self.id, fill=self.color)

    def hit(self, points=1):

        canv.coords(self.id, -10, -10, -10, -10)

        self.points += points

        canv.itemconfig(self.id\_points, text=self.points)

class BomberTarget:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

        self.sp\_x = 0

        self.sp\_y = 5

        self.r = 2

        self.live = 1

        self.id = canv.create\_rectangle(0, 0, 0, 0)

        self.color = 'blue'

    def move(self):

        if self.live != 0:

            self.x += self.sp\_x

            self.y += self.sp\_y

            canv.coords(self.id, self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r)

            canv.itemconfig(self.id, fill=self.color)

    def hit(self):

        canv.coords(self.id, -10, -10, -10, -10)

class Bomber():

    def \_\_init\_\_(self):

        self.x = 20

        self.sp = 5

        self.color = 'blue'

        self.drop\_time = 0

        self.id = canv.create\_rectangle(0, 0, 0, 0)

        canv.coords(self.id, self.x, 50, self.x + 70, 70)

        canv.itemconfig(self.id, fill=self.color)

    def drop(self):

        self.drop\_time = time.time()

        bomb = BomberTarget(self.x + 20, 50)

        btargets.append(bomb)

        bomb.r = 5

        bomb.vx = 0

        bomb.vy = 5

    def moving(self):

        if self.x < 0:

            self.sp = 5

        elif self.x > 800:

            self.sp = -5

        self.x += self.sp

        canv.coords(self.id, self.x, 50, self.x + 70, 70)

        canv.itemconfig(self.id, fill=self.color)

        if time.time() - self.drop\_time > 1:

            self.drop()

t1 = target()

t2 = n\_target()

b1 = Bomber()

screen1 = canv.create\_text(400, 300, text='', font='28')

g1 = gun(500)

g2 = gun(200)

bullet = 0

balls = []

targets = []

btargets = []

k = 1

def new\_game(event=''):

    global gun, t1, t2, screen1, balls, bullet, targets

    bullet = 0

    balls = []

    targets = []

    t1.new\_target()

    t2.new\_target()

    targets.append(t1)

    targets.append(t2)

    canv.bind('<Button-1>', g1.fire2\_start)

    canv.bind('<ButtonRelease-1>', g1.fire2\_end)

    canv.bind('<Motion>', g1.targetting)

    z = 0.03

    t1.live = 1

    while t1.live or t2.live or balls or targets:

        for t in targets:

            t.move()

        for bt in btargets:

            bt.move()

        for b in balls:

            b.move()

            for t in targets:

                if b.hittest(t) and t.live:

                    t.live = 0

                    t.hit()

                    targets.remove(t)

                    canv.bind('<Button-1>', '')

                    canv.bind('<ButtonRelease-1>', '')

                    canv.itemconfig(screen1,

                                    text='Вы уничтожили цель за ' + str(bullet) + ' выстрелов')

            for bt in btargets:

                if b.hittest(bt) and bt.live:

                    bt.live = 0

                    bt.hit()

        canv.update()

        time.sleep(0.03)

        b1.moving()

        g1.power\_up()

        g1.moving()

        g2.power\_up()

        g2.moving()

        g1.targetting()

        g2.targetting()

        t1.move()

        t2.move()

        if k % 2 == 0:

            canv.bind('<Button-1>', g1.fire2\_start)

            canv.bind('<ButtonRelease-1>', g1.fire2\_end)

            canv.bind('<Motion>', g1.targetting)

        else:

            canv.bind('<Button-1>', g2.fire2\_start)

            canv.bind('<ButtonRelease-1>', g2.fire2\_end)

            canv.bind('<Motion>', g2.targetting)

    canv.itemconfig(screen1, text='')

    canv.delete(gun)

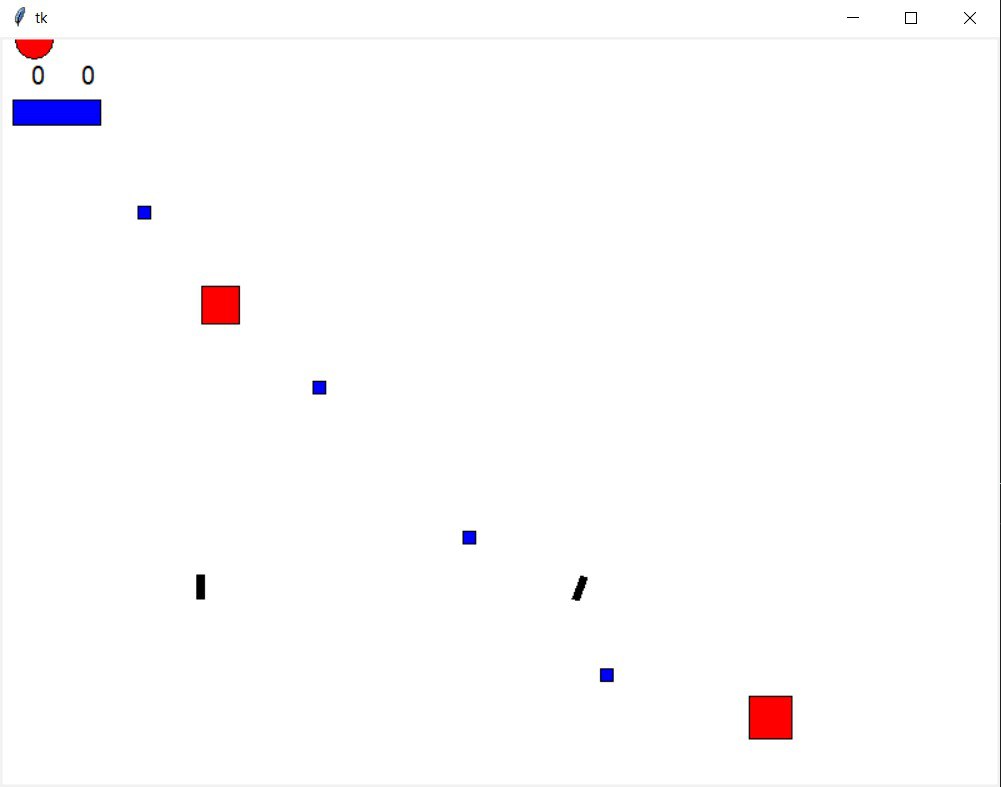
    root.after(750, new\_game)

new\_game()

mainloop()

Задание 2.

Тестирование:



Программа работает

## Задание 3.

Составить UML диаграмму классов, соответствующую коду:

## 

**Вывод**: была выполнена модернизация своей программы.