Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №4  
По дисциплине: «Современные системы программирования»

Выполнил:  
студент 3 курса  
группы ИИ-21(I)  
Кирилович А. А.

Проверила:  
Якимук А.В.

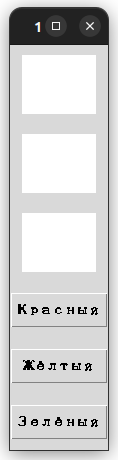
Брест 2023

**Цель:** научиться создавать приложения с использованием апплетов.

**Ход работы**

**Задание 1**

Смоделировать работу световой колонны, управ­ляемой с помощью кнопок состояния системы.

import tkinter as tk  
  
  
class TrafficLight:  
 def \_\_init\_\_(self, master):  
 self.master = master  
 master.title("1")  
  
 self.column\_frame = tk.Frame(master)  
 self.column\_frame.pack()  
  
 self.red\_light = tk.Label(self.column\_frame, width=10, height=5, bg="white")  
 self.yellow\_light = tk.Label(self.column\_frame,width=10, height=5, bg="white")  
 self.green\_light = tk.Label(self.column\_frame, width=10, height=5, bg="white")  
  
 self.red\_light.grid(row=0, column=0, padx=10, pady=10)  
 self.yellow\_light.grid(row=1, column=0, padx=10, pady=10)  
 self.green\_light.grid(row=2, column=0, padx=10, pady=10)  
  
 self.button\_frame = tk.Frame(master)  
 self.button\_frame.pack()  
  
 button\_width = 10  
 button\_height = 2  
  
 self.red\_button = tk.Button(self.button\_frame, text="Красный", command=lambda: self.change\_color("red"), width=button\_width, height=button\_height)  
 self.yellow\_button = tk.Button(self.button\_frame, text="Жёлтый", command=lambda: self.change\_color("yellow"), width=button\_width, height=button\_height)  
 self.green\_button = tk.Button(self.button\_frame, text="Зелёный", command=lambda: self.change\_color("green"), width=button\_width, height=button\_height)  
  
 self.red\_button.grid(row=0, column=0, pady=10)  
 self.yellow\_button.grid(row=1, column=0, pady=10)  
 self.green\_button.grid(row=2, column=0, pady=10)  
  
 def change\_color(self, color):  
 self.red\_light.config(bg="white")  
 self.yellow\_light.config(bg="white")  
 self.green\_light.config(bg="white")  
  
 if color == "red":  
 self.red\_light.config(bg="red")  
 elif color == "yellow":  
 self.yellow\_light.config(bg="yellow")  
 elif color == "green":  
 self.green\_light.config(bg="green")  
  
  
root = tk.Tk()  
traffic\_light\_app = TrafficLight(root)  
root.mainloop()

**Задание 2**

Смоделировать работу любого из светодиодов, управляемой с помощью одной кнопки «Вкл/Выкл»

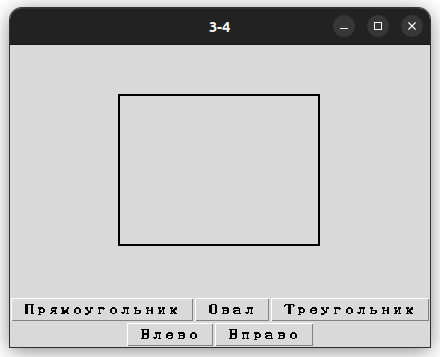
import tkinter as tkclass LEDApp: def \_\_init\_\_(self, master): self.master = master master.title("2") self.led\_status = tk.IntVar() self.led\_status.set(0) self.led\_canvas = tk.Canvas(master, width=50, height=50) self.led\_canvas.pack() self.led = self.led\_canvas.create\_oval(10, 10, 40, 40, fill="gray") self.toggle\_button = tk.Button(master, text="Вкл/Выкл", command=self.toggle\_led) self.toggle\_button.pack() def toggle\_led(self): if self.led\_status.get() == 0: self.led\_canvas.itemconfig(self.led, fill="blue") self.led\_status.set(1) else:  
 self.led\_canvas.itemconfig(self.led, fill="gray")  
 self.led\_status.set(0)  
  
  
root = tk.Tk()  
led\_app = LEDApp(root)  
root.mainloop()

**Задание 3**

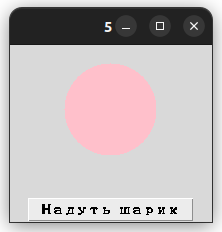
Смена формы фигуры. Создать фигуру и три кнопки. Нажатие на каждую из них кнопок должно приводить к смене формы фигуры. Надписи на кнопках должны соответствовать выполняемым действиям.

**Задание 4**

Смоделируйте движение фигуры влево/вправо при нажатии кнопок.

import tkinter as tk  
  
  
class ShapeManager:  
 def \_\_init\_\_(self, master):  
 self.master = master  
 master.title("3-4")  
  
 self.x\_position = 100  
 self.current\_shape = "rectangle"  
  
 self.canvas = tk.Canvas(master, width=400, height=250)  
 self.canvas.pack()  
  
 self.shape = self.create\_shape(self.x\_position, 50, self.current\_shape)  
  
 shape\_buttons\_frame = tk.Frame(master)  
 shape\_buttons\_frame.pack()  
  
 self.rectangle\_button = tk.Button(shape\_buttons\_frame, text="Прямоугольник",  
 command=lambda: self.change\_shape("rectangle"))  
 self.oval\_button = tk.Button(shape\_buttons\_frame, text="Овал",  
 command=lambda: self.change\_shape("oval"))  
 self.triangle\_button = tk.Button(shape\_buttons\_frame, text="Треугольник",  
 command=lambda: self.change\_shape("triangle"))  
  
 self.rectangle\_button.pack(side=tk.LEFT)  
 self.oval\_button.pack(side=tk.LEFT)  
 self.triangle\_button.pack(side=tk.LEFT)  
  
 move\_buttons\_frame = tk.Frame(master)  
 move\_buttons\_frame.pack()  
  
 self.move\_left\_button = tk.Button(move\_buttons\_frame, text="Влево",  
 command=lambda: self.move\_shape("left"))  
 self.move\_right\_button = tk.Button(move\_buttons\_frame, text="Вправо",  
 command=lambda: self.move\_shape("right"))  
  
 self.move\_left\_button.pack(side=tk.LEFT)  
 self.move\_right\_button.pack(side=tk.LEFT)  
  
 def clear\_canvas(self):  
 self.canvas.delete("all")  
  
 def create\_shape(self, x, y, shape\_type):  
 if shape\_type == "rectangle":  
 return self.canvas.create\_rectangle(x, y, x + 200, y + 150,  
 outline="black", fill="", width=2)  
 elif shape\_type == "oval":  
 return self.canvas.create\_oval(x, y, x + 200, y + 150,  
 outline="black", fill="", width=2)  
 elif shape\_type == "triangle":  
 return self.canvas.create\_polygon(x + 100, y, x + 200, y + 150, x, y + 150,  
 outline="black", fill="", width=2)  
  
 def move\_shape(self, direction):  
 self.clear\_canvas()  
  
 if direction == "left":  
 self.x\_position -= 20  
 elif direction == "right":  
 self.x\_position += 20  
  
 self.shape = self.create\_shape(self.x\_position, 50, self.current\_shape)  
  
 def change\_shape(self, new\_shape):  
 self.clear\_canvas()  
 self.current\_shape = new\_shape  
 self.shape = self.create\_shape(self.x\_position, 50, new\_shape)  
  
  
root = tk.Tk()  
shape\_manager = ShapeManager(root)  
root.mainloop()

**Задание 5**

Смоделируйте постепенное надувание шарика, сохраняя нижнюю точку неподвижной.

import tkinter as tk

class InflatableBallApp:

def \_\_init\_\_(self, master):

self.master = master

master.title("5")

self.radius = 10

self.centerY = 100

self.canvas = tk.Canvas(master, width=200, height=150)

self.canvas.pack()

self.ball = self.canvas.create\_oval(100 - self.radius, self.centerY - self.radius,

100 + self.radius, self.centerY + self.radius,

fill="#FFC0CB", outline="")

self.inflate\_button = tk.Button(master, text="Надуть шарик", command=self.inflate\_ball)

self.inflate\_button.pack()

def inflate\_ball(self):

if self.radius < 50:

self.radius += 1

self.centerY -= 1

self.canvas.delete(self.ball)

self.ball = self.canvas.create\_oval(100 - self.radius, self.centerY - self.radius,

100 + self.radius, self.centerY + self.radius,

fill="#FFC0CB", outline="")

self.canvas.update()

self.canvas.after(100, self.inflate\_ball)

root = tk.Tk()

app = InflatableBallApp(root)

root.mainloop()

**Вывод:** в ходе лабораторной работы я научился создавать приложения с графическим интерфейсом.