

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе № 4.8
«Обработка событий и рисование в Tkinter»**

по дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н.В. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2022

Цель работы: приобретение навыков улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

Ход работы:

1. Клонировал общедоступный репозиторий GitHub на свой локальный сервер.
2. Изучил теоретический материал и приступил к выполнению заданий.

```
task1.py
14 # Создаем функцию для добавления продуктов в список
15 def add_item():
16     product = []
17     select = list(lbox_first.curselection())
18     select.reverse()
19     for item in select:
20         op = lbox_first.get(item)
21         product.append(op)
22     for val in product:
23         lbox_second.insert(0, val)
24     for k in select:
25         lbox_first.delete(k)
26
27
28 # Создаем функцию для удаления из списка
29 def delete_item():
30     product = []
31     select = list(lbox_second.curselection())
32     select.reverse()
33     for item in select:
34         op = lbox_second.get(item)
35         product.append(op)
36     for val in product:
37         lbox_first.insert(0, val)
38     for k in select:
39         lbox_second.delete(k)
```

Рисунок 1 – Код программы первого задания

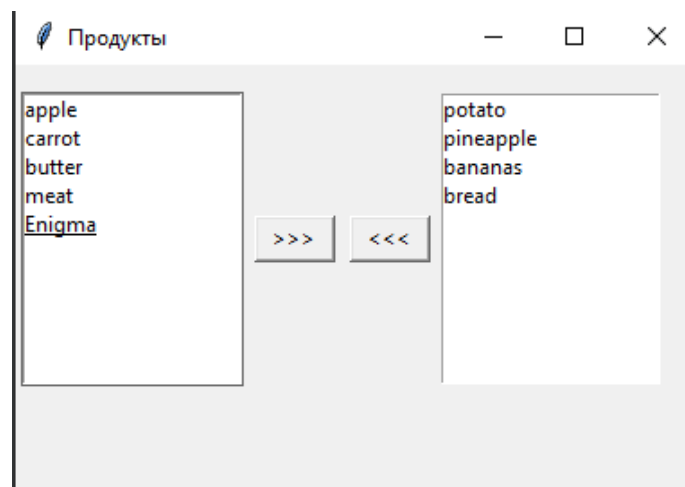


Рисунок 2 – Графический интерфейс программы

3. Выполнил второе задание.

```
12 ▶ if __name__ == "__main__":
13     root = Tk()
14     root.geometry("200x200+90+90")
15     root.title("Task2")
16
17     ent1 = Entry()
18     lb1 = Listbox()
19
20     ent1.pack()
21     lb1.pack()
22     ent1.bind('<Return>', lambda e: lb1.insert(0, ent1.get()))
23     lb1.bind('<Double-Button-1>', lambda e: ent1.insert(0, lb1.get(lb1.curselection())))
24
25     root.mainloop()
```

Рисунок 3 – Код программы второго задания

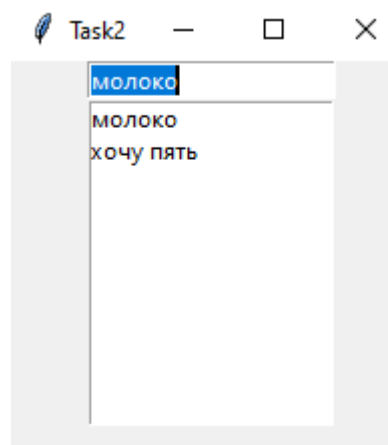


Рисунок 4 – Графический интерфейс программы

4. Выполнил третью задачу.

```
16 def button_click(event):
17     txt1['width'] = ent1.get()
18     txt1['height'] = ent2.get()
19
20
21 def focus_change(event, color):
22     txt1['bg'] = color
23
24
25 ▶ if __name__ == "__main__":
26     root = Tk()
27     root.geometry("500x300+300+300")
28     root.title("Задание 3")
29
30     f = Frame()
31     f.pack()
32     ent1 = Entry(f, width=3)
33     ent2 = Entry(f, width=3)
34     bt1 = Button(f, text='Изменить')
35     txt1 = Text(width=25, height=25, bg='lightgrey')
36
37     bt1.bind('<Button-1>', button_click)
38     ent1.bind('<Return>', button_click)
39     ent2.bind('<Return>', button_click)
40     bt1.bind('<Return>', button_click)
41     root.bind('<Return>', button_click)
```

Рисунок 5 – Код программы третьего задания

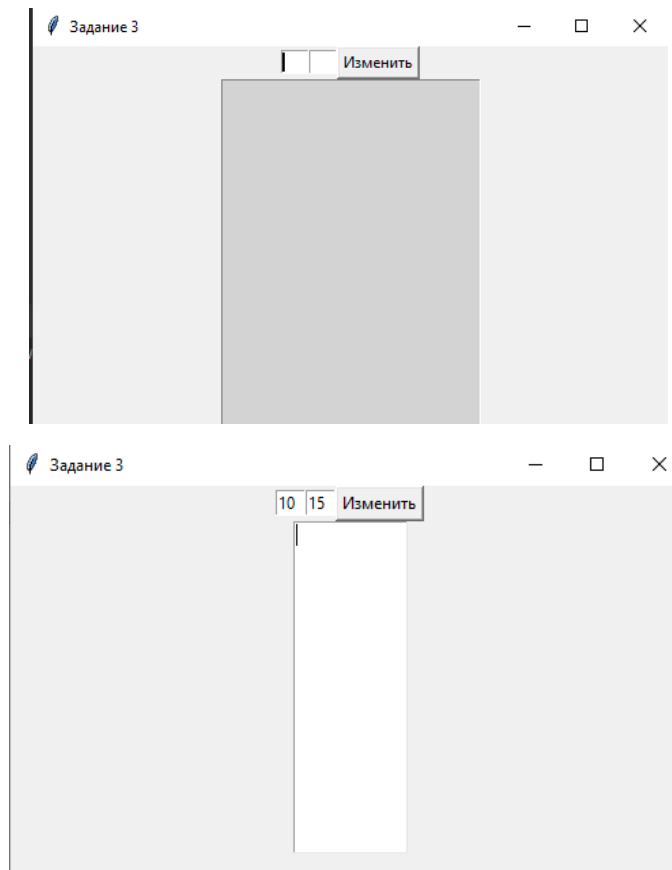


Рисунок 6 – Результат работы программы

5. Выполнил четвертую задачу.

```

10 ▶ if __name__ == '__main__':
11     root = Tk()
12     root.geometry("500x300+300+300")
13     root.title("Task4")
14     c = Canvas(root, width=200, height=200, bg='white')
15     c.pack()
16     c.create_rectangle(50, 90, 140, 175,
17                       fill='#7fc7ff', outline='#7fc7ff')
18     c.create_polygon(30, 90, 160, 90, 95, 40,
19                     fill='#7fc7ff', outline='#7fc7ff')
20     c.create_oval(150, 10, 190, 50,
21                  fill='yellow', outline='yellow')
22     x = -20
23     while x < 200:
24         c.create_arc(x, 170, x + 40, 250, outline='green',
25                     style=ARC, start=200, extent=-80,
26                     width=2)
27         x += 10
28     root.mainloop()

```

Рисунок 7 – Код программы

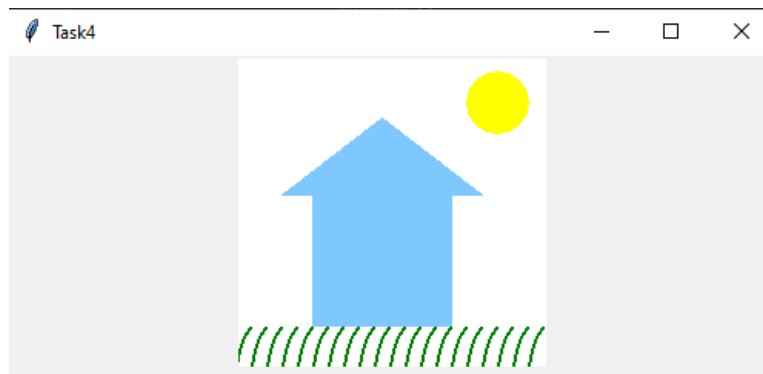


Рисунок 8 – Результат работы программы

6. Выполнил пятую задачу.

```

13  def click(event):
14      c.x = event.x + c.radius
15      c.y = event.y + c.radius
16      motion()
17
18
19  def motion():
20      x = c.coords(c.ball)[2]
21      y = c.coords(c.ball)[3]
22      if c.x == x and c.y == y:
23          return
24      if c.x < x:
25          c.move(c.ball, -1, 0)
26      if c.x > x:
27          c.move(c.ball, 1, 0)
28      if c.y < y:
29          c.move(c.ball, 0, -1)
30      if c.y > y:
31          c.move(c.ball, 0, 1)
32      root.after(10, motion)
33
  
```

Рисунок 9 – Код программы

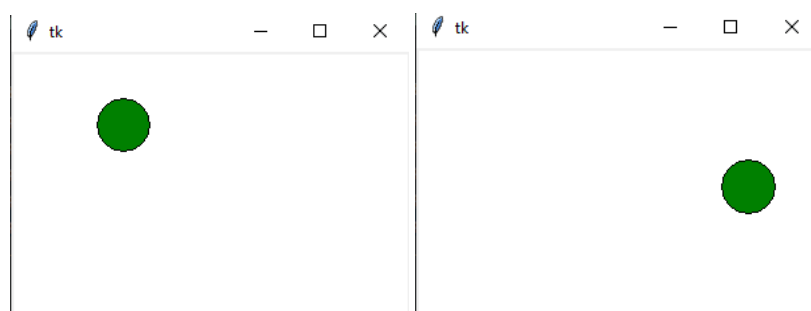


Рисунок 10 – Результат работы программы

Контрольные вопросы:

1. Каково назначение виджета ListBox?

От класса Listbox создаются списки – виджеты, внутри которых в столбик перечисляются элементы. При этом можно выбирать один или множество элементов списка

2. Каким образом осуществляется связывание событие или действие с виджетом Tkinter?

Events - события. В tkinter с помощью метода bind() между собой связываются виджет, событие и действие. Например, виджет – кнопка, событие – клик по ней левой кнопкой мыши, действие – отправка сообщения. Другой пример: виджет – текстовое поле, событие – нажатие Enter, действие – получение текста из поля методом get() для последующей обработки программой.

3. Какие существуют типы событий в Tkinter? Приведите примеры.

Можно выделить три основных типа событий: производимые мышью, нажатиями клавиш на клавиатуре, а также события, возникающие в результате изменения виджетов. Нередко обрабатываются сочетания. Например, клик мышью с зажатой клавишей на клавиатуре.

4. Как обрабатываются события в Tkinter?

Все события обрабатываются методом класса print_event(), который выводит тип события и положение мыши в консоли. Можете поэкспериментировать, нажимая на зеленую рамку мышью и двигая ею, пока она будет выводить сообщения события.

5. Как обрабатываются события мыши в Tkinter?

- <Button-1> – клик левой кнопкой мыши;
- <Button-2> – клик средней кнопкой мыши;
- <Button-3> – клик правой кнопкой мыши;
- <Double-Button-1> – двойной клик левой кнопкой мыши;
- <Motion> – движение мыши.

6. Каким образом можно отображать графические примитивы в Tkinter?

В tkinter от класса Canvas создаются объекты-холсты, на которых можно "рисовать", размещая различные фигуры и объекты. Делается это с помощью вызовов соответствующих методов. При создании экземпляра Canvas необходимо указать его ширину и высоту. При размещении геометрических примитивов и других объектов указываются их координаты на холсте. Точкой отсчета является верхний левый угол.

7. Перечислите основные методы для отображения графических примитивов в Tkinter.

Методом `create_polygon` рисуется произвольный многоугольник путем задания координат каждой его точки.

Метод `create_oval` создает эллипсы. При этом задаются координаты гипотетического прямоугольника, описывающего эллипс. Если нужно получить круг, то соответственно описываемый прямоугольник должен быть квадратом.

Более сложные для понимания фигуры получаются при использовании метода `create_arc`. В зависимости от значения опции `style` можно получить сектор (по умолчанию), сегмент (`CHORD`) или дугу (`ARC`). Также как в случае `create_oval` координаты задают прямоугольник, в который вписана окружность (или эллипс), из которой "вырезают" сектор, сегмент или дугу. Опции `start` присваивается градус начала фигуры, `extent` определяет угол поворота.

8. Каким образом можно обратиться к ранее созданным фигурам на холсте?

В Tkinter существует два способа "пометить" фигуры, размещенные на холсте, – это идентификаторы и теги. Первые всегда уникальны для каждого объекта. Два объекта не могут иметь одни и тот же идентификатор. Теги не уникальны. Группа объектов на холсте может иметь один и тот же тег. Это дает возможность менять свойства всей группы. Отдельно взятая фигура на Canvas может иметь как идентификатор, так и тег.

9. Каково назначение тэгов в Tkinter?

В отличие от идентификаторов, которые являются уникальными для каждого объекта, один и тот же тег может присваиваться разным объектам. Дальнейшее обращение к такому тегу позволит изменить все объекты, в которых он был указан. Метод `tag_bind` позволяет привязать событие (например, щелчок кнопкой мыши) к определенной фигуре на Canvas. Таким образом, можно реализовать обращение к различным областям холста с помощью одного и того же события.