УДК 004.624

Создание простого графического редактора на языке программирования Python

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания простого графического редактора способного рисовать простые герметические фигуры. Для создания используется язык программирования Python и встроенная библиотека Tkinter. В созданном графическом редакторе можно рисовать простые фигуры, такие как: линии, дуги, прямоугольники, овалы, а также вставлять текст и рисовать карандашом.
Ключевые слова: Python, Tkinter, графический редактор

Creating a simple graphical editor in the Python programming language

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating a simple graphic editor capable of drawing simple hermetic shapes. It uses the Python programming language and the built-in Tkinter library. In the created graphical editor it is possible to draw simple shapes, such as: lines, arcs, rectangles, ovals, as well as to insert text and draw with a pencil.
Keywords: Python, Tkinter, graphic editor

При обучении детей программированию многие темы бывает сложно объяснить без наглядно примера. Материал лучше усваивается при визуальном контакте с результатом работы написанной программы. Таким примером может выступать и простой графический редактор. Отсутствие большого количества кода и упрощённой структура программы лучше понять основа работы с элементом canvas библиотеки Tkinter. Чтобы рисовать нестандартные фигуры можно воспользоваться карандашом, а простые элемента такие как прямые, прямоугольники и т.д. уже встроенные в модуль Tkinter.

Цель исследования – создать графического редактора на базе языка программирования Python.

Ранее этим вопросом интересовались И.Е. Юков, П.Б. Крылов, Д.В. Мельников, М.Г. Роминов, М.В. Степанов развивали тему «Трехмерное объектно-ориентированное ядро для разработки графических приложений» [1] в которой была проведена функциональная структуризация объектно-ориентированного ядра. Выделены компоненты, необходимые при реализации трехмерных приложений в различных прикладных областях: твердотельное и геометрическое ядро, графическое ядро и компонента прямого манипулирования в пространстве. Была разработана структура базового трехмерного приложения и объектно- -ориентированная технология порождения специализированных трехмерных приложений, заключающаяся в переопределении базовых механизмов и реализации специализированных методов. Разработано и реализовано объектно-ориентированное ядро для построения высокоинтерактивных трехмерных графических приложений в различных прикладных областях информатики, представляющее собой набор функционально независимых иерархий классов (программно реализованных на языке C++).Г.А. Ломакин, А.С. Гусейнова с темой «Общие подходы к разработке интегрированной среды для построения графических приложений» [2], а подробнее про подходы к созданию интерактивной веб-среды для построения 3D-приложений. При современном развитии графических систем и технологий актуальным является перевод визуализации на новый уровень, значительно упрощающий разработку 3D-объектов. Научная новизна подхода состоит в создании интегрированной системы для реализации графических приложений, которая абстрагирует конечного пользователя от промежуточных уровней создания графики и дает возможность напрямую строить графические сцены посредством графического интерфейса на таких современных платформах, как Windows 7 и Android. Основная концепция, лежащая в основе предлагаемого решения, связана с созданием набора классов и утилит, которые реализуют более общие подходы к визуализации графических объектов и скрывают от пользователя непосредственное применение тех или иных математических моделей и графических алгоритмов. Это позволит более широкому кругу заинтересованных лиц создавать требуемые 3D-объекты и сцены, затрачивая минимальное время на разработку необходимого решения. А.Д. Филинских, А.А. Иванова опубликовали статью «Разработка api приложений для графических пакетов» [3] описали вопрос использования прикладного программирования в графических пакетах для решения задач, связанных с разработкой, модернизацией и оптимизацией геометрических моделей, а также доработка пользовательского интерфейса под нужды конкретного пользователя. В пособии представлен практикум с примерами программного кода для реализации конкретных примеров.

На языке программирования Python есть встроенная библиотека Tkinter[4], это библиотека позволяет создавать графические интерфейсы для программ на Python. Библиотека обладает большим функционалом, в частности и графическими инструментами. Модуль canvas из этой

библиотеки позволяет внутри графического окна рисовать геометрические фигуры и обладает функцией карандаша.

Виджет Canvas предоставляет графические средства для Tkinter. Среди этих графических объектов есть линии, круги, изображения и даже другие виджеты. С помощью этого виджета можно рисовать графики и графики, создавать графические редакторы и реализовывать различные виды пользовательских виджетов.

Пример кода взаимодействия с библиотекой представлен ниже.

```
import tkinter.font
from tkinter import ARC, Canvas, Menu, Tk


class PaintApp:
    drawing = "pencil"
    left_button = "up"
    x_position, y_position = (
        None,
        None,
    )
    x1_line, y1_line, x2_line, y2_line = (
        None,
        None,
        None,
        None,
    )

    @staticmethod
    def quit_app(event: None = None) -> None:
        root.quit()

    def __init__(self: "PaintApp", root: None) -> None:
        drawing_area = Canvas(root)
        drawing_area.pack()
        drawing_area.bind("<Motion>", self.motion)
        drawing_area.bind("<ButtonPress-1>", self.l_button_down)
        drawing_area.bind("<ButtonRelease-1>", self.l_button_up)
        the_menu = Menu(root)
        file_menu = Menu(the_menu, tearoff=0)
        file_menu.add_command(label="Линия",
command=self.line_drawing)
        file_menu.add_command(label="Карандаш",
command=self.pencil_drawing)
        file_menu.add_command(label="Дуга", command=self.arc_drawing)
        file_menu.add_command(
```

```
            label="Прямоугольник",
            command=self.rectangle_drawing,
        )
        file_menu.add_command(label="Овал", command=self.oval_drawing)
        file_menu.add_command(label="Текст", command=self.text_drawing)
        file_menu.add_separator()
        file_menu.add_command(label="Выход", command=self.quit_app)
        the_menu.add_cascade(label="Опции", menu=file_menu)
        root.config(menu=the_menu)

    def line_drawing(self: "PaintApp") -> None:
        self.drawing = "line"

    def pencil_drawing(self: "PaintApp") -> None:
        self.drawing = "pencil"

    def arc_drawing(self: "PaintApp") -> None:
        self.drawing = "arc"

    def rectangle_drawing(self: "PaintApp") -> None:
        self.drawing = "rectangle"

    def oval_drawing(self: "PaintApp") -> None:
        self.drawing = "oval"

    def text_drawing(self: "PaintApp") -> None:
        self.drawing = "text"

    def l_button_down(self: "PaintApp", event: None = None) -> None:
        self.left_button = "down"
        self.x1_line = event.x
        self.y1_line = event.y

    def l_button_up(self: "PaintApp", event: None = None) -> None:
        self.left_button = "up"
        self.x_position = None
        self.y_position = None
        self.x2_line = event.x
        self.y2_line = event.y
        if self.drawing == "line":
            self.line_draw(event)
        if self.drawing == "pencil":
            self.pencil_draw(event)
        if self.drawing == "arc":
            self.arc_draw(event)
```

```
    if self.drawing == "oval":
        self.oval_draw(event)
    if self.drawing == "rectangle":
        self.rect_draw(event)
    if self.drawing == "text":
        self.text_draw(event)

def motion(self: "PaintApp", event: None = None) -> None:
    if self.drawing == "pencil":
        self.pencil_draw(event)
        self.x_position = event.x
        self.y_position = event.y

def pencil_draw(self: "PaintApp", event: None = None) -> None:
    if (
        self.left_button == "down"
        and self.x_position is not None
        and self.y_position is not None
    ):
        event.widget.create_line(
            self.x_position,
            self.y_position,
            event.x,
            event.y,
            smooth=True,
        )

def line_draw(self: "PaintApp", event: None = None) -> None:
    if None not in (
        self.x1_line,
        self.x2_line,
        self.y1_line,
        self.y2_line,
    ):
        event.widget.create_line(
            self.x1_line,
            self.x2_line,
            self.y1_line,
            self.y2_line,
            smooth=True,
            fill="green",
        )

def arc_draw(self: "PaintApp", event: None = None) -> None:
    if None not in (
```

```
            self.x1_line,
            self.x2_line,
            self.y1_line,
            self.y2_line,
        ):
            coords = self.x1_line, self.x2_line, self.y1_line, self.y2_line
            event.widget.create_arc(
                coords,
                start=0,
                extent=150,
                style=ARC,
                fill="blue",
            )

    def oval_draw(self: "PaintApp", event: None = None) -> None:
        if None not in (
            self.x1_line,
            self.x2_line,
            self.y1_line,
            self.y2_line,
        ):
            event.widget.create_oval(
                self.x1_line,
                self.x2_line,
                self.y1_line,
                self.y2_line,
                fill="midnight blue",
                outline="yellow",
                width=2,
            )

    def rect_draw(self: "PaintApp", event: None = None) -> None:
        if None not in (
            self.x1_line,
            self.x2_line,
            self.y1_line,
            self.y2_line,
        ):
            event.widget.create_rectangle(
                self.x1_line,
                self.x2_line,
                self.y1_line,
                self.y2_line,
                fill="red",
                outline="pink",
```

```
            width=2,
        )

    def text_draw(self: "PaintApp", event: None = None) -> None:
        if None not in (self.x1_line, self.y1_line):
            text_font = tkinter.font.Font(
                family="Helvetica",
                size=20,
                weight="bold",
                slant="italic",
            )
            event.widget.create_text(
                self.x1_line,
                self.y1_line,
                fill="lightblue",
                font=text_font,
                text="Привет!",
            )


root = Tk()
paint_app = PaintApp(root)
root.mainloop()
```

Первое окно выглядит как на рисунке 1.

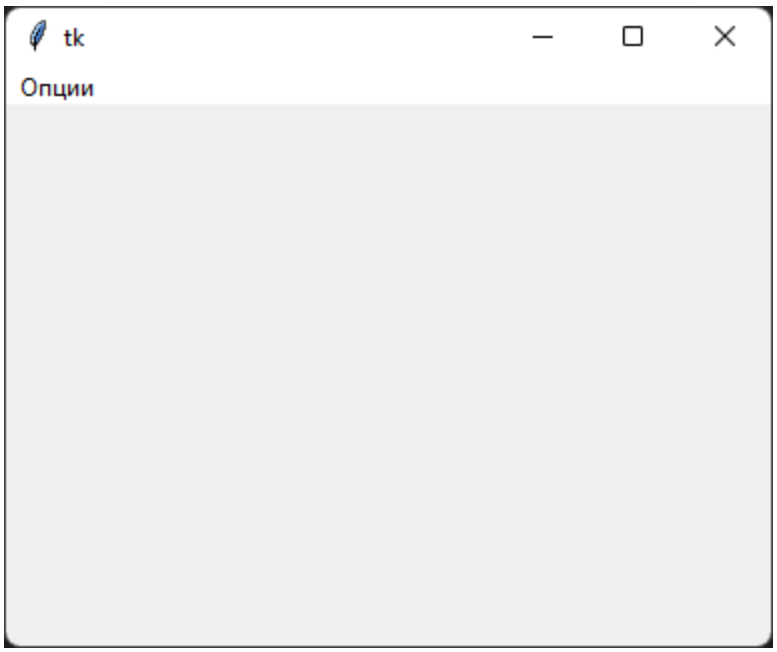

Рисунок 1. Главное окно редактора

В панели меню в левом верхнем углу можно найти список функций, таких как:

- Линия
- Карандаш
- Дуга
- Прямоугольник
- Овал
- Текст

Пример рисования линий представлен на рисунке 2.

Рисунок 2. Рисование линий

Пример рисования карандашом представлен на рисунке 3.

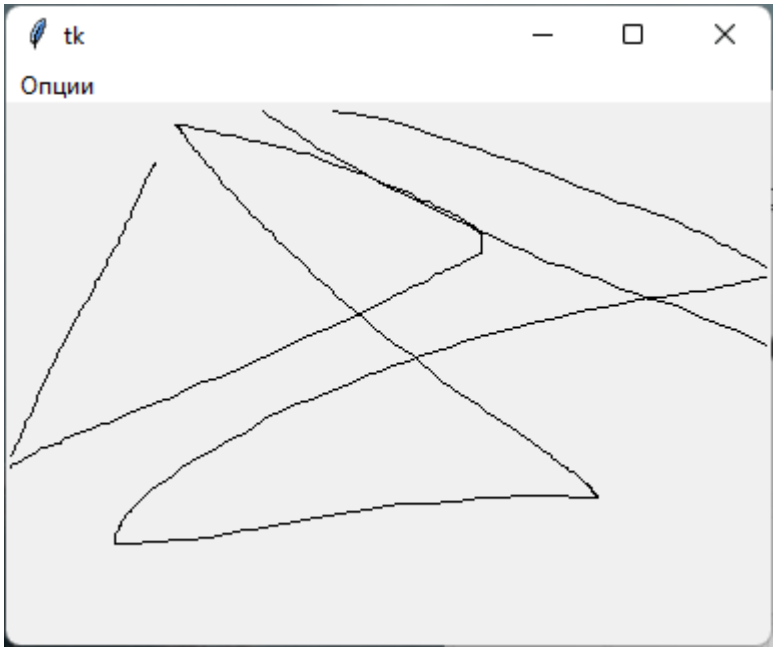

Рисунок 3. Рисование карандашом

Пример рисования дуг представлен на рисунке 4.

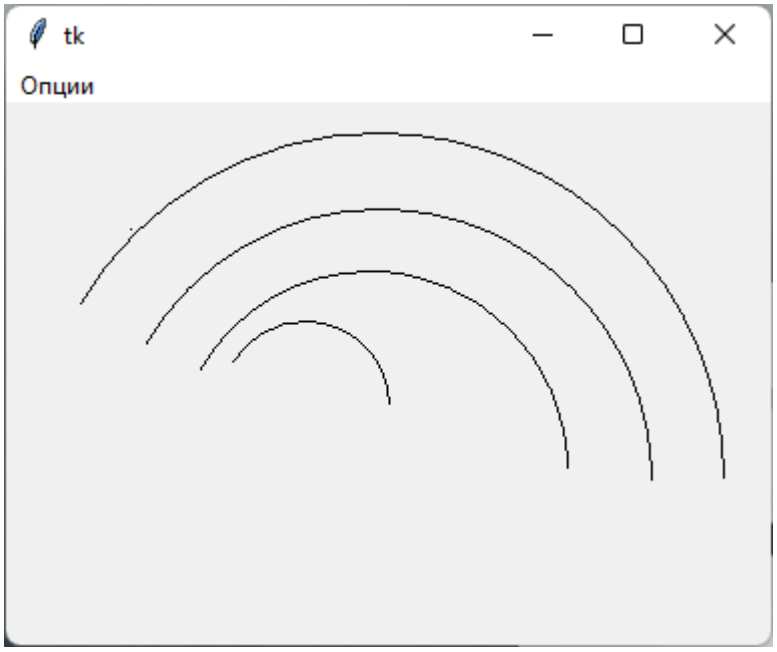

Рисунок 4. Рисование дуг

Пример рисования прямоугольников представлен на рисунке 5.

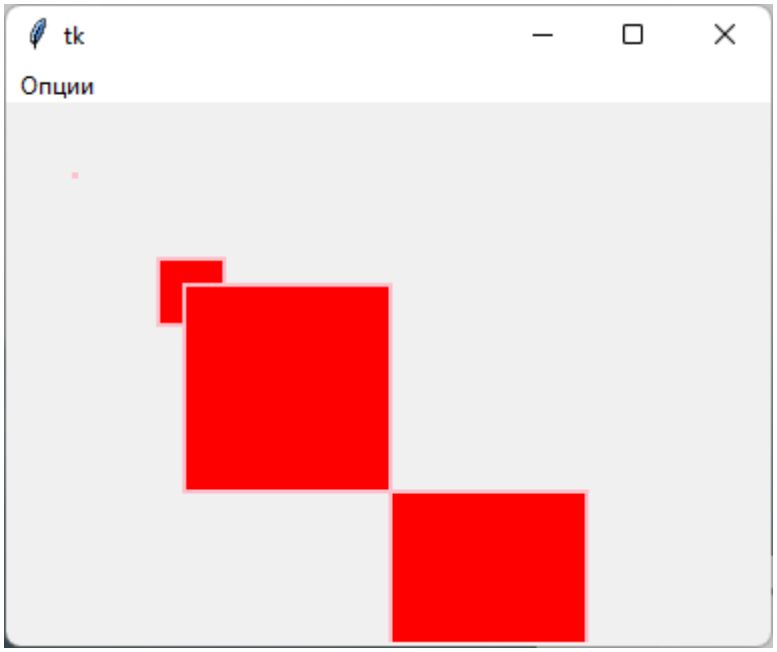

Рисунок 5. Рисование прямоугольников

Пример рисования овалов представлен на рисунке 6.

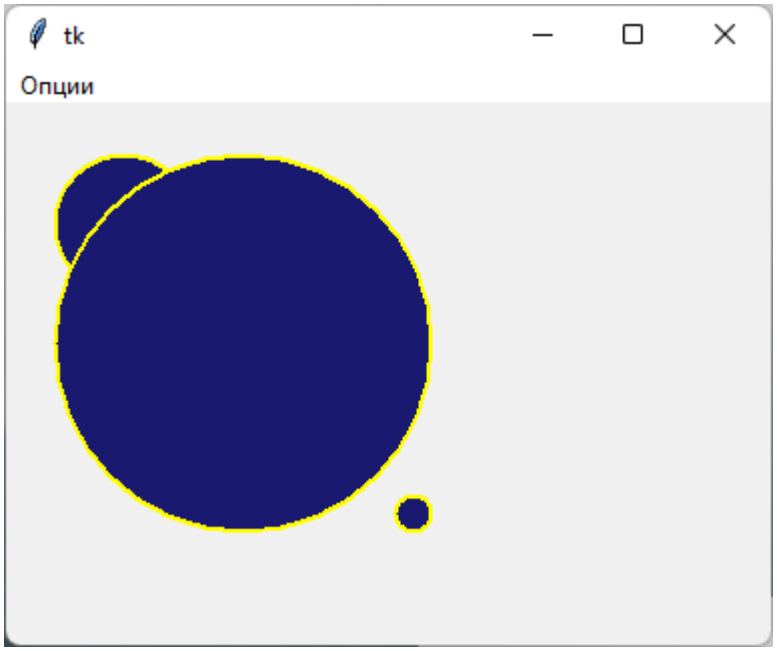

Рисунок 6. Рисование овалов.

Пример добавления текста представлен на рисунке 7.

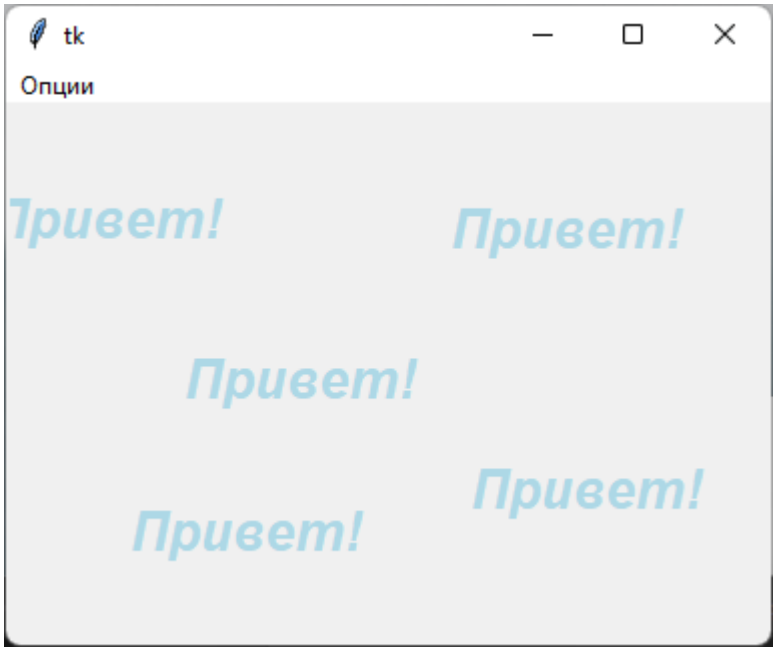

Рисунок 7. Добавление текста

Также все режимы можно объединять, как на рисунке 8.

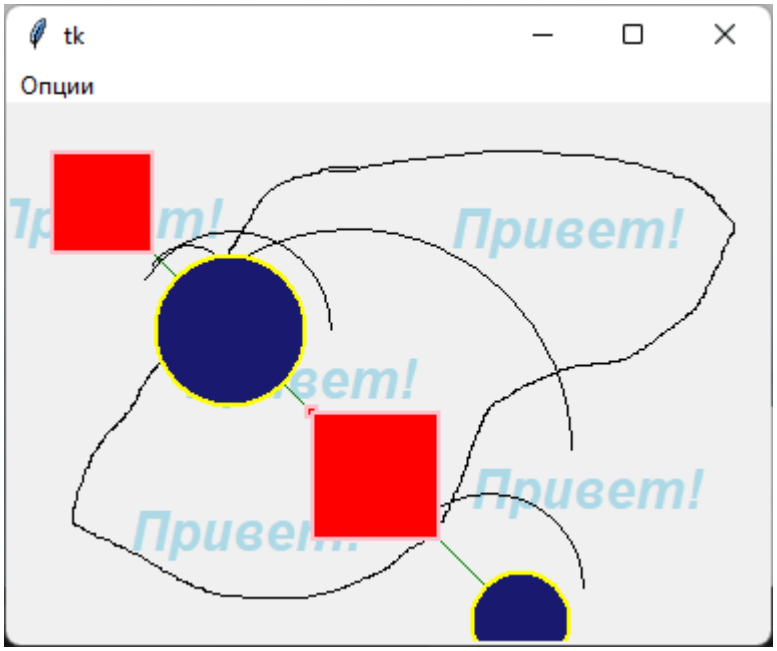

Рисунок 8. Все 6 режимов рисования

Вывод

В этой статье были описаны основы того, как работать с графической составляющей библиотеки tkinter языка программирования Python. Реализован простой графический редактор и продемонстрирован его функционал по некоторым встроенным функция модуля Tkinter.

Библиографический список

1. Юков И.Е., Крылов П.Б., Мельников Д.В., Роминов М.Г., Степанов М.В. Трехмерное объектно-ориентированное ядро для разработки графических приложений // Отчет о НИР № 95-01-01238 (Российский фонд фундаментальных исследований) URL: https://www.elibrary.ru/item.asp?id=222832 (Дата обращения: 15.06.2022)
2. Ломакин Г.А., Гусейнова А.С. Общие подходы к разработке интегрированной среды для построения графических приложений // Вестник БГУ. Серия 1, Физика. Математика. Информатика. 2013. № 1. С. 56-60. URL: https://www.elibrary.ru/item.asp?id=21116729 (Дата обращения: 15.06.2022)
3. Филинских А.Д., Иванова А.А. Разработка api приложений для графических пакетов. Нижний Новгород, 2017. URL: https://www.elibrary.ru/item.asp?id=41155978 (Дата обращения: 15.06.2022)
4. Tkinter https://docs.python.org/3/library/tkinter.html (Дата обращения: 15.06.2022)