Министерство образования и науки России

Рыбинский государственный авиационный технический университет

имени П.А. Соловьева

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ДИСЦИПЛИНА:

Объектно-ориентированное программирование

ОТЧЁТ   
ПО ЛАБАРАТОРНОЙ РАБОТЕ № 1

Выполнил Мухин Д.С.

(фамилия, имя, отчество)

студент гр. ЗИС-23

Преподаватель Пруктишина В.А.

(фамилия, имя, отчество)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Рыбинск

2024 г.

**Содержание**

[1. Постановка задачи 3](#_Toc184225809)

[2. Спецификация 4](#_Toc184225810)

[3. Теоретические сведения 7](#_Toc184225811)

[4. Организация данных 10](#_Toc184225812)

[5. Текст программы 14](#_Toc184225813)

1. **Постановка задачи**

На языке программирования Python разработать графический редактор Требования:

* должны использоваться визуальные компоненты, с кнопками и формами ввода;
* должны соблюдаться основные принципы ООП, элементы точка и линии должны быть классами, рисование — методы;
* код выкладывать на GitHub.

1. **Спецификация**

Этот графический редактор создан с использованием библиотеки **Tkinter** для Python, предоставляя пользователю возможность рисовать основные графические элементы, такие как точки, линии, круги, квадраты, треугольники, а также произвольно рисовать с помощью мыши. Интерфейс программы интуитивно понятен и включает простые инструменты для рисования, изменения толщины линий и выбора цвета.

## **Ключевые функции программы**

1. **Инструмент «Точка»:**
   * Пользователь может выбрать «Точку» и кликать на холст для ее размещения.
   * Цвет и размер точки можно настроить через панель инструментов.
2. **Инструмент «Линия»:**
   * Линия создается путем выбора двух точек: первая — при клике ЛКМ, вторая — при отпускании кнопки.
   * Толщина и цвет линии настраиваются через панель инструментов.
3. **Инструмент «Круг»:**
   * Круг рисуется путем выбора двух точек, определяющих диаметр.
   * Толщина и цвет линии круга настраиваются в панели инструментов.
4. **Инструмент «Квадрат»:**
   * Квадрат рисуется через два щелчка, задающих противоположные углы.
   * Цвет и толщина линий настраиваются через панель инструментов.
5. **Инструмент «Треугольник»:**
   * Треугольник рисуется путем выбора двух точек, определяющих его основание.
   * Автоматически создается равнобедренный треугольник, настройка цвета и толщины доступна через панель инструментов.
6. **Свободное рисование:**
   * Пользователь может рисовать произвольные линии, удерживая левую кнопку мыши.
   * Настройка толщины и цвета линии доступна в панели инструментов.
7. **Выбор цвета:**
   * Пользователь может выбрать цвет для всех инструментов рисования через окно выбора цвета.
8. **Настройка толщины линий:**
   * Пользователь может задать толщину всех линий через соответствующее поле в панели инструментов.
   * По умолчанию толщина установлена на значение 3.
9. **Функция отмены действия:**
   * Программа сохраняет историю всех действий пользователя, и по нажатию кнопки «Отменить» можно удалить последнее нарисованное.

**Принцип управления программой**

Запуск программы:

1. Исполняемый файл

Для запуска программы необходимо найти файл с именем graph\_red.exe и дважды кликнуть по этому файлу, чтобы открыть программу.

1. Окно программы

После запуска программы откроется главное окно, содержащее:

Рабочую область (холст), на которой будут рисоваться элементы; Панель инструментов, позволяющую выбирать цвет, толщину линий, тип рисуемых элементов.

Завершение работы:

После завершения работы просто закройте окно программы. Программа не сохраняет результаты работы.

**Основные элементы интерфейса**

* **Холст (Canvas):** Рабочая область для рисования, где отображаются все элементы.
* **Панель инструментов (Toolbar):**
  + Кнопка для выбора инструмента «Точка».
  + Кнопка для выбора инструмента «Линия».
  + Кнопка для выбора инструмента «Круг».
  + Кнопка для выбора инструмента «Квадрат».
  + Кнопка для выбора инструмента «Треугольник».
  + Кнопка для свободного рисования.
  + Кнопка для выбора цвета рисуемых объектов.
  + Поле для установки толщины линий.
  + Кнопка «Отменить» для возврата на одно действие назад.
  + Кнопка «Очистить» для возвращения холста к исходному состоянию.

## **Реализация и структура**

* **Модульность и ООП**: Каждый элемент (точка, линия, круг, квадрат, треугольник) реализован как отдельный класс с методами для создания и отображения объекта на холсте. Это позволяет легко расширять программу и добавлять новые инструменты рисования.
* **История действий**: Все действия пользователя сохраняются в список, что позволяет реализовать функцию «Отменить». Это обеспечивается хранением каждого созданного объекта в списке, который служит основой для отмены действий.
* **Обработка событий**: Основная логика программы использует методы обработки кликов мыши и движения мыши для рисования объектов на холсте. Используется обработка нажатий кнопок для выбора инструментов и изменения параметров рисования.

## **Ограничения**

* **Отсутствие сохранения изображений**: Программа не поддерживает экспорт или сохранение изображений, созданных пользователем.
* **Ограниченные возможности редактирования**: Программа фокусируется на базовых функциях рисования и не включает сложных инструментов редактирования, таких как изменение уже нарисованных объектов.

## **Технические детали**

* Все объекты рисования реализованы через классы в Python с использованием библиотеки **Tkinter**.
* Для управления цветом используется диалоговое окно выбора цвета.
* История действий сохраняется в памяти программы для возможности отмены.
* Основной цикл программы работает с обработчиками событий, привязанными к элементам интерфейса.

Этот редактор служит хорошим инструментом для простых графических задач и может быть расширен для реализации более сложных функций, таких как экспорт изображений или создание многоуровневых объектов.

# **Организация данных**

Программа построена с использованием объектно-ориентированного подхода, что позволяет эффективно управлять рисуемыми объектами и взаимодействием с пользователем. Основные части программы включают в себя классы для графических элементов, обработку событий и управление состоянием редактора.

## **Основные компоненты программы**

* 1. Классы для рисуемых объектов – эти классы отвечают за представление и отрисовку графических элементов, таких как точка, линия, круг, квадрат и треугольник. Каждый объект имеет свои характеристики, такие как цвет, толщина и координаты, и реализует метод отрисовки.
  2. Класс редактора – управляет логикой работы программы: взаимодействием с пользователем, обработкой событий (например, нажатие на кнопки и рисование на холсте), а также хранением истории действий для возможности их отмены.

## **Описание классов и методов**

Абстрактный класс Drawable

Drawable является базовым для всех классов, которые представляют рисуемые объекты. Этот класс описывает основные атрибуты объектов, такие как цвет, толщина и холст, на котором они будут отображаться.

Атрибуты:

* canvas: объект холста, на котором будет рисоваться фигура.
* color: цвет объекта.
* thickness: толщина линии или размер объекта.

Методы:

* draw(): абстрактный метод, который должен быть реализован в каждом наследующем классе для рисования конкретного объекта.

Подклассы Drawable

Point — класс для рисования точки:

* Сохраняет координаты x и y.
* Метод draw() реализует отрисовку точки с использованием метода canvas.create\_oval.

Line — класс для рисования линии:

* Хранит координаты начальной и конечной точек (x1, y1, x2, y2).
* Метод draw() рисует линию с помощью canvas.create\_line.

Circle, Square, Triangle — классы для рисования круга, квадрата и треугольника:

* Каждый класс использует координаты для определения размеров объектов.
* Реализует метод draw() с использованием таких функций, как canvas.create\_oval, canvas.create\_rectangle, и canvas.create\_polygon.

Класс DrawingEditor

Это главный класс программы, который координирует взаимодействие между пользователем и графическим интерфейсом. Он управляет рисованием объектов и обработкой событий.

Атрибуты:

* canvas: рабочая область для рисования.
* current\_color: выбранный пользователем цвет для объектов.
* current\_thickness: текущая толщина линии или размера объекта.
* history: список, в котором сохраняются все нарисованные объекты для отмены действий.
* drawing\_mode: текущий режим рисования (точка, линия, фигура и т. д.).
* temp\_shape: временный объект, который отображается при рисовании объектов.

Методы:

* create\_toolbar(): создает панель инструментов для выбора цвета, толщины и типа объекта.
* bind\_canvas\_events(): связывает события мыши с соответствующими обработчиками (например, нажатие, движение, отпускание кнопки).
* choose\_color(): открывает диалоговое окно для выбора цвета.
* get\_thickness(): возвращает текущую толщину линии или объекта.

## **Режимы рисования**

Для управления различными режимами рисования (например, для рисования точек, линий, кругов и т. д.) в классе DrawingEditor реализованы методы:

* set\_point\_mode(): включает режим рисования точек.
* set\_line\_mode(): включает режим рисования линии.
* set\_circle\_mode(): включает режим рисования круга.
* set\_square\_mode(): включает режим рисования квадрата.
* set\_triangle\_mode(): включает режим рисования треугольника.
* set\_draw\_mode(): включает режим произвольного рисования.

## **Обработка событий**

Программа реагирует на действия пользователя с помощью следующих методов обработки событий:

* on\_click(event) — вызывается при нажатии кнопки мыши. Здесь происходит создание объекта выбранного типа (точка, линия, фигура) и начало рисования.
* on\_drag(event) — вызывается при движении мыши с зажатой кнопкой. Используется для изменения формы или размеров объектов в процессе рисования.
* on\_release(event) — вызывается при отпускании кнопки мыши. Завершается рисование и фиксируется объект.
* undo() — метод для отмены последнего действия, который удаляет последний объект из холста и списка истории.

## **Логика работы программы**

1. Запуск и интерфейс: При запуске программы открывается главное окно с холстом и панелью инструментов. Панель инструментов позволяет пользователю выбрать цвет, толщину и тип рисуемого объекта.
2. Выбор инструмента: Пользователь выбирает инструмент (например, точку, линию, круг, квадрат) через панель инструментов. Программа переключается в нужный режим рисования, и пользователь может начать рисовать.
3. Рисование: При нажатии на холст создается объект выбранного класса (например, точка или линия). Метод draw() этого объекта отвечает за его отображение на холсте. В случае сложных объектов (таких как линия или круг) создается временный объект, который обновляется по мере перемещения мыши.
4. Отмена действий: Все объекты, нарисованные пользователем, сохраняются в списке history. Кнопка «Отменить» позволяет удалить последний элемент из списка и с холста, отменяя последнюю операцию.
5. Цвет и толщина: Пользователь может выбрать цвет для объектов через диалоговое окно, которое открывается кнопкой «Цвет». Толщину линий и фигур можно задать в текстовом поле, что влияет на все нарисованные элементы.
6. **Текст программы**

import tkinter as tk

from tkinter import colorchooser

# Абстрактный класс, представляющий общую основу для фигур.

class Shape:

    def \_\_init\_\_(self, color, thickness):

        self.color = color

        self.thickness = thickness

# Класс точки, наследует базовые свойства из Shape.

class Point(Shape):

    def \_\_init\_\_(self, x, y, color, thickness):

        super().\_\_init\_\_(color, thickness) # Инициализируем базовые свойства

        self.x = x # Координата x

        self.y = y # Координата y

    # Метод для рисования точки на холсте

    def draw(self, canvas):

        radius = self.thickness // 2

        return canvas.create\_oval(

            self.x - radius, self.y - radius,

            self.x + radius, self.y + radius,

            fill=self.color, outline=self.color

        )

# Класс линии, наследует Shape.

class Line(Shape):

    def \_\_init\_\_(self, start\_point, end\_point, color, thickness):

        super().\_\_init\_\_(color, thickness)

        self.start\_point = start\_point # Начальная точка

        self.end\_point = end\_point # Конечная точка

    # Метод для рисования линии на холсте

    def draw(self, canvas):

        return canvas.create\_line(

            self.start\_point.x, self.start\_point.y,

            self.end\_point.x, self.end\_point.y,

            fill=self.color, width=self.thickness

        )

# Класс круга.

class Circle(Shape):

    def \_\_init\_\_(self, start\_point, end\_point, color, thickness):

        super().\_\_init\_\_(color, thickness)

        self.start\_point = start\_point

        self.end\_point = end\_point

    # Метод для рисования круга

    def draw(self, canvas):

        x1, y1 = self.start\_point.x, self.start\_point.y

        x2, y2 = self.end\_point.x, self.end\_point.y

        return canvas.create\_oval(x1, y1, x2, y2, outline=self.color, width=self.thickness)

# Класс прямоугольника.

class Rectangle(Shape):

    def \_\_init\_\_(self, start\_point, end\_point, color, thickness):

        super().\_\_init\_\_(color, thickness)

        self.start\_point = start\_point

        self.end\_point = end\_point

    # Метод для рисования прямоугольника

    def draw(self, canvas):

        x1, y1 = self.start\_point.x, self.start\_point.y

        x2, y2 = self.end\_point.x, self.end\_point.y

        return canvas.create\_rectangle(x1, y1, x2, y2, outline=self.color, width=self.thickness)

# Класс треугольника.

class Triangle(Shape):

    def \_\_init\_\_(self, start\_point, end\_point, color, thickness):

        super().\_\_init\_\_(color, thickness)

        self.start\_point = start\_point

        self.end\_point = end\_point

    # Метод для рисования треугольник

    def draw(self, canvas):

        x1, y1 = self.start\_point.x, self.start\_point.y

        x2, y2 = self.end\_point.x, self.end\_point.y

        x3 = (x1 + x2) / 2

        y3 = min(y1, y2) - abs(x2 - x1) / 2

        return canvas.create\_polygon(

            x1, y1, x2, y2, x3, y3,

            outline=self.color, width=self.thickness, fill=''

        )

# Основной класс графического редактора.

class GraphicEditor:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Графический редактор")

        self.canvas = tk.Canvas(self.root, width=800, height=600, bg="white")

        self.canvas.pack()

        # Инициализация параметров

        self.color = "black" # Цвет по умолчанию

        self.thickness = 2 # Толщина по умолчанию

        self.start\_point = None

        self.preview\_id = None

        self.shapes = [] # Список нарисованных фигур

        self.free\_draw\_shapes = [] # Временный список для свободного рисования

        self.current\_mode = "free\_draw" # Режим по умолчанию

        self.create\_ui() # Создаем элементы управления

    def create\_ui(self):

        # Кнопки и поля

        color\_button = tk.Button(self.root, text="Цвет", command=self.choose\_color)

        thickness\_label = tk.Label(self.root, text="Толщина:")

        self.thickness\_entry = tk.Entry(self.root, width=5)

        self.thickness\_entry.insert(0, "2")

        undo\_button = tk.Button(self.root, text="Отмена", command=self.undo)

        clear\_button = tk.Button(self.root, text="Очистить", command=self.clear\_canvas)

        free\_draw\_button = tk.Button(self.root, text="Рисование", command=lambda: self.set\_mode("free\_draw"))

        circle\_button = tk.Button(self.root, text="Круг", command=lambda: self.set\_mode("circle"))

        rectangle\_button = tk.Button(self.root, text="Прямоугольник", command=lambda: self.set\_mode("rectangle"))

        triangle\_button = tk.Button(self.root, text="Треугольник", command=lambda: self.set\_mode("triangle"))

        line\_button = tk.Button(self.root, text="Прямая линия", command=lambda: self.set\_mode("line"))

        point\_button = tk.Button(self.root, text="Точка", command=lambda: self.set\_mode("point"))

        # Размещение элементов

        color\_button.pack(side=tk.LEFT)

        thickness\_label.pack(side=tk.LEFT)

        self.thickness\_entry.pack(side=tk.LEFT)

        undo\_button.pack(side=tk.LEFT)

        clear\_button.pack(side=tk.LEFT)

        free\_draw\_button.pack(side=tk.LEFT)

        circle\_button.pack(side=tk.LEFT)

        rectangle\_button.pack(side=tk.LEFT)

        triangle\_button.pack(side=tk.LEFT)

        line\_button.pack(side=tk.LEFT)

        point\_button.pack(side=tk.LEFT)

        # Привязка событий к холсту

        self.canvas.bind("<Button-1>", self.on\_mouse\_down) # Нажатие кнопки мыши

        self.canvas.bind("<B1-Motion>", self.on\_mouse\_drag) # Перетаскивание

        self.canvas.bind("<ButtonRelease-1>", self.on\_mouse\_up) # Отпускание кнопки

    # Выбор цвета

    def choose\_color(self):

        color = colorchooser.askcolor()[1]

        if color:

            self.color = color

    # Установка режима

    def set\_mode(self, mode):

        self.current\_mode = mode

        self.start\_point = None

        self.free\_draw\_shapes.clear()

        self.root.title(f"Графический редактор - Режим: {mode.capitalize()}")

    # Обработчик нажатия мыши

    def on\_mouse\_down(self, event):

        self.start\_point = Point(event.x, event.y, self.color, self.thickness)

        if self.current\_mode == "free\_draw":

            self.preview\_id = self.start\_point.draw(self.canvas)

            self.free\_draw\_shapes.append([self.preview\_id, self.start\_point])

        elif self.current\_mode == "point":

            self.start\_point.draw(self.canvas)

    # Обработчик перетаскивания

    def on\_mouse\_drag(self, event):

        self.thickness = int(self.thickness\_entry.get())  # Обновление толщины

        if self.current\_mode == "free\_draw":

            point = Point(event.x, event.y, self.color, self.thickness)

            line = Line(self.start\_point, point, self.color, self.thickness)

            self.start\_point = point

            shape\_id = line.draw(self.canvas)

            self.free\_draw\_shapes[-1].append(shape\_id)

        elif self.current\_mode == "circle" and self.start\_point:

            if self.preview\_id:

                self.canvas.delete(self.preview\_id)  # Удаляем старый предварительный вид

            end\_point = Point(event.x, event.y, self.color, self.thickness)

            shape = Circle(self.start\_point, end\_point, self.color, self.thickness)

            self.preview\_id = shape.draw(self.canvas)

        elif self.current\_mode == "rectangle" and self.start\_point:

            if self.preview\_id:

                self.canvas.delete(self.preview\_id)

            end\_point = Point(event.x, event.y, self.color, self.thickness)

            shape = Rectangle(self.start\_point, end\_point, self.color, self.thickness)

            self.preview\_id = shape.draw(self.canvas)

        elif self.current\_mode == "triangle" and self.start\_point:

            if self.preview\_id:

                self.canvas.delete(self.preview\_id)

            end\_point = Point(event.x, event.y, self.color, self.thickness)

            shape = Triangle(self.start\_point, end\_point, self.color, self.thickness)

            self.preview\_id = shape.draw(self.canvas)

        elif self.current\_mode == "line" and self.start\_point:

            if self.preview\_id:

                self.canvas.delete(self.preview\_id)

            end\_point = Point(event.x, event.y, self.color, self.thickness)

            line = Line(self.start\_point, end\_point, self.color, self.thickness)

            self.preview\_id = line.draw(self.canvas)

    # Завершение рисования фигуры

    def on\_mouse\_up(self, event):

        self.thickness = int(self.thickness\_entry.get())

        if self.current\_mode in ["circle", "rectangle", "triangle", "line"] and self.start\_point:

            end\_point = Point(event.x, event.y, self.color, self.thickness)

            if self.current\_mode == "circle":

                shape = Circle(self.start\_point, end\_point, self.color, self.thickness)

            elif self.current\_mode == "rectangle":

                shape = Rectangle(self.start\_point, end\_point, self.color, self.thickness)

            elif self.current\_mode == "triangle":

                shape = Triangle(self.start\_point, end\_point, self.color, self.thickness)

            elif self.current\_mode == "line":

                shape = Line(self.start\_point, end\_point, self.color, self.thickness)

            shape\_id = shape.draw(self.canvas)

            self.shapes.append([(shape\_id, shape)])

            self.start\_point = None

            if self.preview\_id:

                self.canvas.delete(self.preview\_id)

                self.preview\_id = None

    # Отмена последнего действия

    def undo(self):

        if self.free\_draw\_shapes:

            for shape in self.free\_draw\_shapes[-1]:

                self.canvas.delete(shape)

            self.free\_draw\_shapes.pop()

        elif self.shapes:

            last\_shape = self.shapes.pop()

            for shape\_id, \_ in last\_shape:

                self.canvas.delete(shape\_id)

     # Очистка холста

    def clear\_canvas(self):

        self.canvas.delete("all")

        self.shapes.clear()

        self.free\_draw\_shapes.clear()

root = tk.Tk()

editor = GraphicEditor(root)

root.mainloop()