**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

специальности 6-05-0611-01 Информационные системы

и технологии (в игровой индустрии)

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине: «Высокоуровневые языки программирования»

на тему: **«Работа с графикой. Графическая сцена.»**

Выполнил: студент гр. ИТИ-21

Важная Я. И.

Принял: ассистент

Карась О.В.

Гомель 2024

# **Цель работы:** изучить класс QPainter, содержащий инструменты для рисования произвольной графики; изучить компонент «графическое представление».

**Задание:** Создать приложение, которое при запуске выводит окно с надписью (см. ниже). Поменять значок в заголовке на свой. В заголовке окна указать свои имя, фамилию и группу.

Под надписью должно находится 3 кнопки. При наведении мыши на кнопку должна выводится подсказка.

Кнопки

1 – Открыть окно с перерисовкой любой знаменитой картины и подписи к ней. При отрисовке картины использовать классы

• Класс QСоlог. цвет

• Класс QPen: перо

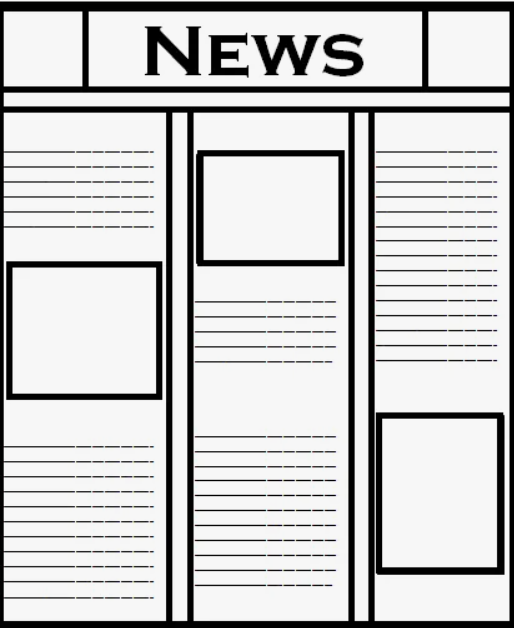
• Класс QBrush: кисть

• Класс QLine: линия

• Класс QPolygon: многоугольник

• Класс QFont шрифт

• Класс QPainter (рисование линий и фигур, вывод текста)



2 – Открыть окно, в котором отображен газетный лист. Обязательно должен присутствовать заголовок, 3(или больше) рисунка, текст должен быть осмысленным и подходящим под название вашей газеты.

При реализации обязательно использовать

– метод drawТext () класса QPainter

– методы drawPixmap () и drawImage () класса QPainter

– класс QPixmap

– класс QBitmap

– класс QImage

Пример газеты

– 3 – Открыть окно, содержащее графическую сцену. В сцену поместить 10 разных объектов разного цвета и формы (треугольник, круг, прямоугольник, текст, рисунок и т.д.).

Передвижение по сцене осуществить с помощью линеек прокрутки.

Часть объектов должны быть зафиксированы на сцене, для оставшихся у пользователя должна быть возможность их перемещения.

При щелчке мыши на зафиксированном объекте, он должен изменяться (например, изменить цвет границы).

Если объект можно переместить, то при наведении на него курсора, вид курсора должен измениться (например, со «стрелки» на «руку»)

Окно должно содержать кнопки

• Изменить размеры сцены

• Показать элемент(т.е. переместиться к этому элементу в сцене)

• Повернуть элемент

• Изменить размер элемента

• Сделать элемент видимым/не видимым

• Переместить элемент

• Запретить/разрешить перемещение для элементов

• Сгруппировать/разгруппировать элементы (для их дальнейшего перемещения пользователем)

• Добавить/убрать эффект 1

• Добавить/убрать эффект 2

Элементы, с которыми осуществляются преобразования, и используемые эффекты выбрать самостоятельно

При реализации обязательно использовать

• QGraphicsScene:

• QGraphicsView

• QGraphicsItem

• QGraphicsLineItem

• QAbstractGraphicsShapeItem

• QGraphicsRectItem

• QGraphicsPolygonItem

• QGraphicsEllipseItem

• QGraphicsPixrnapItem

• QGraphicsSimpleTextItem

• QGraphicsItemGroup

• QGraphicsEffect

**Ход работы:**

1. Импорт необходимых модулей

Все необходимые модули, нужные для создания окна представлены на рисунке 1.

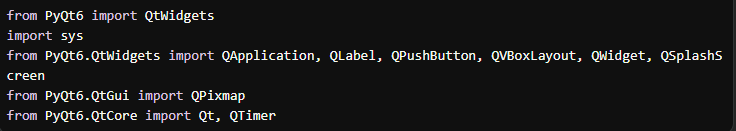


Рисунок 1 – Импорт необходимых модулей

* *PyQt6.QtWidgets*: Содержит все виджеты для создания графического интерфейса.
* *sys*: Используется для работы с системными аргументами и выхода из приложения.
* *QApplication*: Основной класс для управления потоком приложения.
* *QLabel*: Виджет для отображения текстовой информации.
* *QPushButton*: Виджет кнопки.
* *QVBoxLayout*: Макет для вертикального размещения виджетов.
* *QWidget*: Базовый класс для всех виджетов.
* *QPixmap*: Класс для работы с изображениями.

1. Создание главного окна с кнопкой.

На рисунке 2 представлен фрагмент кода для создания макета и добавления виджетов.

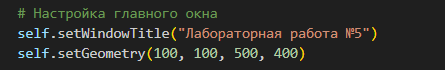


Рисунок 2 – Фрагмент кода для создания макета и добавления виджетов

* Создание макета: layout = *QVBoxLayout*() создает вертикальный макет для размещения виджетов.
* Создание метки: label = *QLabel*(...) создает текстовую метку.
* Добавление метки в макет: *layout.addWidget(label)* добавляет метку в макет.

1. Основной блок запуска приложения

На рисунке 3 представлен фрагмент кода для запуска приложения.

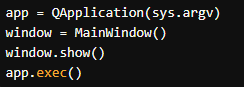


Рисунок 3 – Фрагмент кода для запуска приложения

* Инициализация приложения: *QApplication* запускает приложение.
* Создание главного окна: Создается экземпляр *MainWindow*.
* Запуск основного цикла: *app.exec*() обрабатывает события приложения до его завершения.

1. Результат работы программы

На рисунках 4 представлен результат работы программы (запуск приложения и появление главного окна приложения).

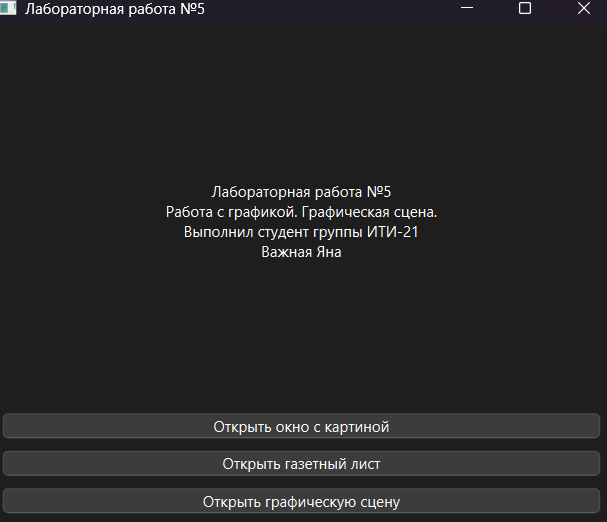


Рисунок 4 – Главное окно приложения

На рисунке 5 показано окно, которое выводит картину «Звездная ночь Ван Гога».



Рисунок 5 – Картина

На рисунке 6 показано окно с газетой, оформленной по примеру задания.



Рисунок 6 – Газета

На рисунке 7 показано графическое окно с фигурами и изображением, а также полный функционал приложения, представленного кнопками, несущий в себе определенную функцию.

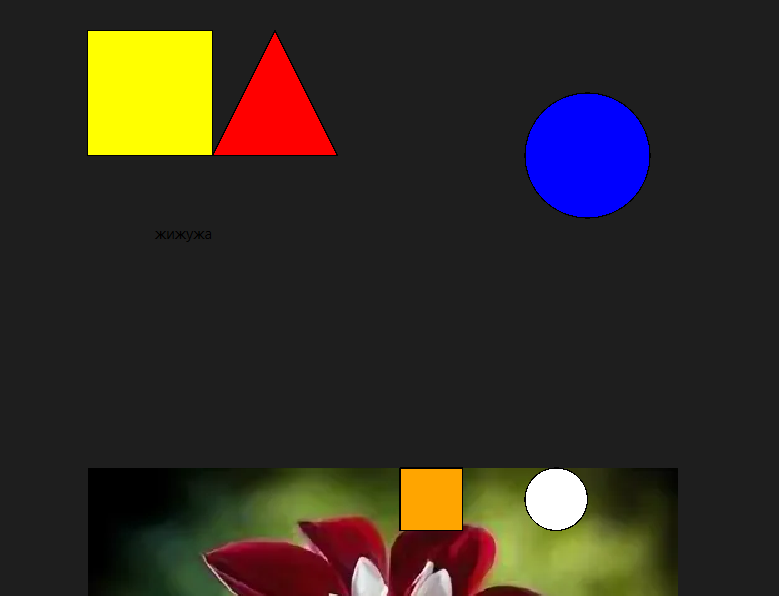


Рисунок 7 – Поле

Листинг кода представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы были изучены классы, реализующие различные виды списков (раскрывающийся, обычный и иерархический) и таблицу, определенные в модуле QtWidgets.

**ПРИЛОЖЕНИЕ А  
(обязательное)  
Листинг кода**

*laba4.py*

import sys

from PyQt6.QtWidgets import (

    QApplication, QMainWindow, QPushButton, QMessageBox, QComboBox,

    QVBoxLayout, QWidget, QTableView, QTreeView, QHeaderView, QLabel

)

from PyQt6.QtGui import QIcon, QStandardItemModel, QStandardItem

from PyQt6.QtCore import Qt

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        # Настройка главного окна

        self.setWindowTitle("Спиридонов Дмитрий Алексеевич, ИТИ-21")

        self.setGeometry(100, 100, 400, 300)

        self.setWindowIcon(QIcon('1.png'))  # Путь к вашему значку

        # Создание кнопки

        self.button = QPushButton("Открыть статью", self)

        self.button.setToolTip("Нажмите для открытия окна со статьей")

        self.button.clicked.connect(self.open\_article\_window)

        self.button.setGeometry(100, 150, 200, 40)

    def open\_article\_window(self):

        self.article\_window = ArticleWindow()

        self.article\_window.show()

class ArticleWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setWindowTitle("Лабораторная работа №4 - Списки и таблицы")

        self.setGeometry(150, 150, 600, 600)

        # Создание основного виджета и макета

        self.central\_widget = QWidget()

        self.setCentralWidget(self.central\_widget)

        self.layout = QVBoxLayout(self.central\_widget)

        # Выпадающий список (QComboBox)

        self.combo\_box = QComboBox(self)

        self.combo\_box.addItems(["Введение в ИИ", "Применение ИИ", "Этические аспекты ИИ"])

        self.layout.addWidget(self.combo\_box)

        # Текстовое содержание статьи

        self.article\_text = QLabel(self)

        self.article\_text.setWordWrap(True)

        self.article\_text.setAlignment(Qt.AlignmentFlag.AlignTop)

        article\_content = """

<h2>Искусственный интеллект и его влияние на общество</h2>

<h3>Введение в искусственный интеллект</h3>

Искусственный интеллект (ИИ) - это область компьютерных наук, которая стремится создать системы, способные выполнять задачи, требующие человеческого интеллекта, такие как восприятие, обучение, принятие решений и решение проблем. ИИ стремится имитировать человеческое мышление и поведение с помощью алгоритмов и вычислительных систем.

<h3>Применение искусственного интеллекта</h3>

ИИ находит широкое применение в различных сферах, включая медицину, финансы, транспорт, образование и развлечения. Например, алгоритмы ИИ используются для ранней диагностики заболеваний, оптимизации финансовых операций, управления автономными транспортными средствами и персонализации развлекательного контента.

<h3>Этические аспекты искусственного интеллекта</h3>

Развитие ИИ поднимает важные этические вопросы, связанные с конфиденциальностью, безопасностью, ответственностью и справедливостью. Необходимо тщательно рассматривать потенциальные негативные последствия применения ИИ, такие как потеря рабочих мест, усиление существующих предубеждений и нарушение неприкосновенности частной жизни. Важно разработать этические принципы и нормативные акты, регулирующие развитие и использование ИИ.

"""

        self.article\_text.setText(article\_content)

        self.layout.addWidget(self.article\_text)

        # Обычный список (QListView через QStandardItemModel)

        self.list\_model = QStandardItemModel()

        self.list\_model.appendRow(QStandardItem("Пункт списка 1"))

        self.list\_model.appendRow(QStandardItem("Пункт списка 2"))

        self.list\_model.appendRow(QStandardItem("Пункт списка 3"))

        self.list\_view = QTreeView(self)

        self.list\_view.setModel(self.list\_model)

        self.layout.addWidget(self.list\_view)

        # Иерархический список (QTreeView)

        self.tree\_model = QStandardItemModel()

        parent\_item = self.tree\_model.invisibleRootItem()

        for i in range(3):

            parent = QStandardItem(f"Раздел {i+1}")

            for j in range(3):

                child = QStandardItem(f"Подраздел {i+1}.{j+1}")

                parent.appendRow(child)

            parent\_item.appendRow(parent)

        self.tree\_view = QTreeView(self)

        self.tree\_view.setModel(self.tree\_model)

        self.tree\_view.expandAll()  # Развернем все элементы

        self.layout.addWidget(self.tree\_view)

        # Таблица 1 (QTableView)

        self.table1\_model = QStandardItemModel(3, 3)

        self.table1\_model.setHorizontalHeaderLabels(['Колонка 1', 'Колонка 2', 'Колонка 3'])

        for row in range(3):

            for column in range(3):

                item = QStandardItem(f'Ячейка {row+1},{column+1}')

                self.table1\_model.setItem(row, column, item)

        self.table1\_view = QTableView(self)

        self.table1\_view.setModel(self.table1\_model)

        self.table1\_view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeMode.Stretch)

        self.layout.addWidget(self.table1\_view)

        # Таблица 2 (QTableView)

        self.table2\_model = QStandardItemModel(2, 2)

        self.table2\_model.setHorizontalHeaderLabels(['A', 'B'])

        for row in range(2):

            for column in range(2):

                item = QStandardItem(f'Data {row+1},{column+1}')

                self.table2\_model.setItem(row, column, item)

        self.table2\_view = QTableView(self)

        self.table2\_view.setModel(self.table2\_model)

        self.table2\_view.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeMode.Stretch)

        self.layout.addWidget(self.table2\_view)

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    main\_window = MainWindow()

    main\_window.show()

    sys.exit(app.exec())