**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 №2

по дисциплине: «Высокоуровневые языки программирования»

на тему: **«Работа с окнами. Обработка сигналов и событий»**

Выполнил: студент гр. ИТИ-21

ВажнаяЯ И

Принял: ассистент

Карась О.В.

Гомель 2024

# **Цель работы:** научиться создавать и выводить окна, управлять их размерами. Изучить обработку сигналов и событий.

**Задание:** Создать приложение, которое при запуске выводит окно с надписью (см. ниже) заданного размера (таблица 1) и заданного расположения (таблица 1). В заголовке окна указать свои имя, фамилию и группу. Поменять значок в заголовке на свой. При написании надписи использовать 5 различных стилей оформления текста (на выбор).

Под надписью должно находится 2 кнопки. При наведении мыши на кнопку должна выводится подсказка.

Кнопки

1 – открыть модальное окно заданной формы с любым фоновым рисунком и заданной прозрачностью (таблица 1).

2 – открыть модальное окно и котором организовано выполнение заданной функции (таблица 1). Надпись в окне:

**Лабораторная работа №2**

**Работа с окнами. Обработка сигналов и событий**

Выполнил студент группы ИТИ-21

Фамилия Имя Отчество

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Размер окна W H | Расположение окна X Y | Форма окна | Прозрачность | Функции |
| 4 | 400x400 | 500, 600 | параллелограмм | 40 | При нажатии клавиши на клавиатуре выводится какая клавиша была нажата |

**Ход работы:**

1. Импорт необходимых модулей

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

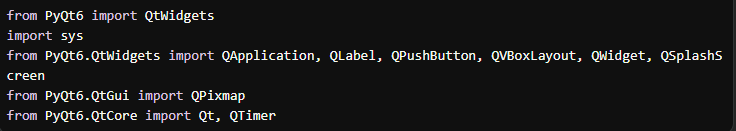


Рисунок 1 – Импорт необходимых модулей

* *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. Функции для создания главного окна

На рисунке 2 представлен фрагмент кода для создания главного окна.

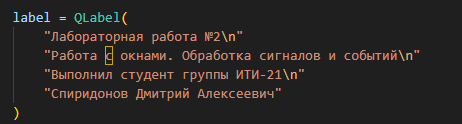


Рисунок 2 – Фрагмент кода для создания главного окна

* Создание окна: main\_window = *QWidget*() создает экземпляр главного окна.
* Установка заголовка: *setWindowTitle*("Имя Фамилия ИТИ-21") задает заголовок окна.

1. Создание макета и добавление виджетов.

На рисунке 3 представлен фрагмент кода для создания макета и добавления виджетов.

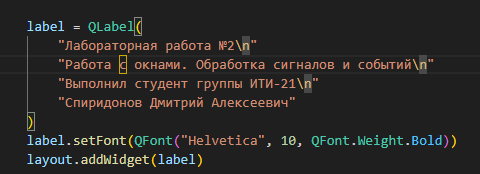


Рисунок 3 – Фрагмент кода для создания макета и добавления виджетов

* 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. Создание кнопки и добавление её в макет

На рисунке 4 представлен результат выполнения программы для четвертого задания.

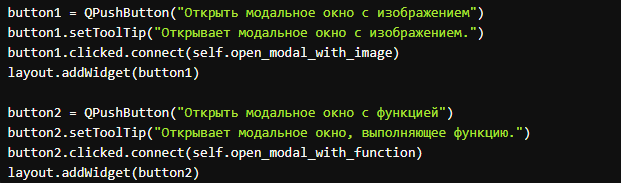


Рисунок 4 – Результат выполнения программы для четвертого задания

* Создание кнопок: *QPushButton* создает кнопки с текстом.
* Установка подсказок: *setToolTip* добавляет всплывающие подсказки.
* Соединение сигналов с методами: *clicked.connect*связывает нажатие кнопки с функцией.
* Добавление кнопок в макет: *addWidget* добавляет кнопки в макет.

1. Функции для открытия модальных окон

На рисунке 5 представлен фрагмент кода функций для открытия модальных окон.

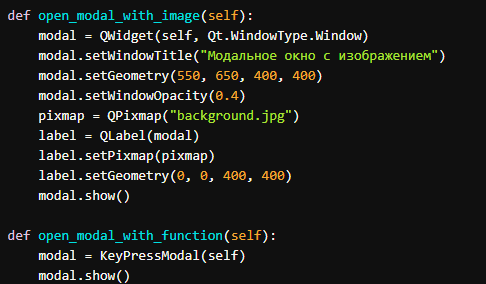


Рисунок 5 – Фрагмент кода для установки размеров и центрирования окна

* Создание модального окна: *QWidget* создает новое окно.
* Настройка прозрачности: *setWindowOpacity* задает уровень прозрачности.
* Добавление изображения: *QPixmap* и *QLabel* для отображения изображения.
* Открытие окна с функцией: *KeyPressModal* обрабатывает нажатия клавиш.

1. Обработка событий клавиш

На рисунке 6 представлен фрагмент кода для обработки событий клавиш.

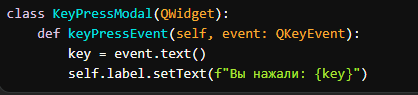


Рисунок 6 – Фрагмент кода для обработки событий клавиш

* Обработка нажатий клавиш*: keyPressEvent* выводит информацию о нажатой клавише.

1. Основной блок запуска приложения

На рисунке 7 представлен фрагмент кода для запуска приложения.

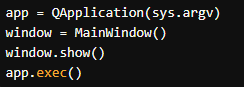


Рисунок 7 – Фрагмент кода для запуска приложения

* 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 8, 9, 10 представлен результат работы программы.

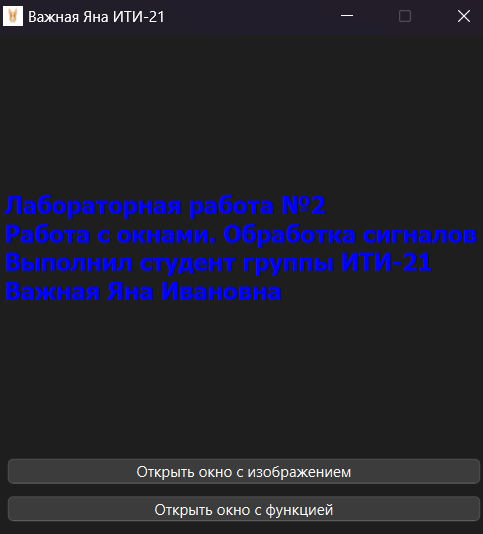


Рисунок 8 – Главное окно приложения

После запуска приложения сразу же появляется главное окно, показанное на рисунке 8. На рисунке 9 показано модальное окно с полупрозрачным фоном.

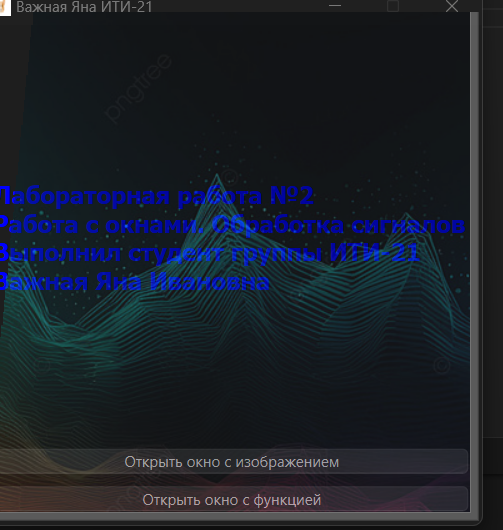


Рисунок 9 – Модальное полупрозрачное окно в форме параллелограмма

На рисунке 10 показан результат вывода в окно последней нажатой пользователем клавиши.

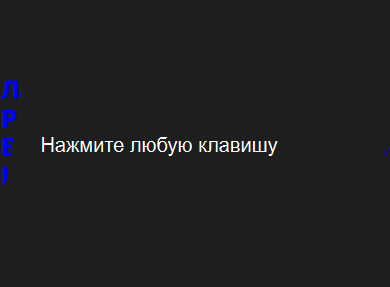


Рисунок 10 – Модальное окно в форме параллелограмма с функцией вывода нажатой клавиши

Листинг кода представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы были созданы окна, а также редактировались их размеры, была изучена обработка сигналов и событий.

**ПРИЛОЖЕНИЕ А  
(обязательное)  
Листинг кода**

*lab2.py*

import sys

from PyQt6.QtWidgets import QApplication, QVBoxLayout, QLabel, QPushButton, QMainWindow, QDialog, QWidget

from PyQt6.QtGui import QFont, QPixmap, QIcon, QPolygon, QRegion

from PyQt6.QtCore import Qt, QPoint

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

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

        self.setFixedSize(500, 500)

        self.setWindowIcon(QIcon('icon.png'))  # Убедитесь, что файл существует

        layout = QVBoxLayout()

        label = QLabel(

            "Лабораторная работа №2\n"

            "Работа с окнами. Обработка сигналов и событий\n"

            "Выполнил студент группы ИТИ-21\n"

            "Спиридонов Дмитрий Алексеевич"

        )

        label.setFont(QFont("Helvetica", 12, QFont.Weight.Bold))

        layout.addWidget(label)

        button1 = QPushButton("Открыть окно с изображением")

        button1.setToolTip("Открывает окно с изображением.")

        button1.clicked.connect(self.open\_window\_with\_image)

        layout.addWidget(button1)

        button2 = QPushButton("Открыть окно с функцией")

        button2.setToolTip("Открывает окно, выполняющее функцию.")

        button2.clicked.connect(self.open\_window\_with\_function)

        layout.addWidget(button2)

        container = QWidget()

        container.setLayout(layout)

        self.setCentralWidget(container)

        self.center\_window()

    def center\_window(self):

        screen = QApplication.primaryScreen().availableGeometry()

        x = (screen.width() - self.width()) // 2

        y = (screen.height() - self.height()) // 2

        self.move(x, y)

    def open\_window\_with\_image(self):

        window = ParallelogramWindow(self)

        window.setWindowTitle("Окно с изображением")

        window.setFixedSize(500, 500)

        window.setWindowOpacity(0.9)

        window.show()  # Используем show вместо exec

    def open\_window\_with\_function(self):

        window = KeyPressWindow(self)

        window.show()  # Используем show вместо exec

class ParallelogramWindow(QDialog):  # Наследуем от QDialog

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setFixedSize(500, 500)

        self.apply\_parallelogram\_mask()

        pixmap = QPixmap("fonn.jpg")  # Убедитесь, что файл существует

        label = QLabel(self)

        label.setPixmap(pixmap)

        label.setGeometry(0, 0, 500, 500)

    def apply\_parallelogram\_mask(self):

        points = [

            QPoint(50, 0), QPoint(450, 0),

            QPoint(400, 500), QPoint(0, 500)

        ]

        polygon = QPolygon(points)

        self.setMask(QRegion(polygon))

class KeyPressWindow(QDialog):  # Наследуем от QDialog

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("Окно с функцией")

        self.setFixedSize(400, 300)

        self.apply\_parallelogram\_mask()

        self.label = QLabel("Нажмите любую клавишу", self)

        self.label.setFont(QFont("Arial", 12))

        self.label.setGeometry(50, 130, 300, 20)

    def apply\_parallelogram\_mask(self):

        points = [

            QPoint(50, 0), QPoint(350, 0),

            QPoint(300, 300), QPoint(0, 300)

        ]

        polygon = QPolygon(points)

        self.setMask(QRegion(polygon))

    def keyPressEvent(self, event):

        key = event.text()

        self.label.setText(f"Вы нажали: {key}")

app = QApplication(sys.argv)

window = MainWindow()

window.show()

app.exec()