Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

# МОДИФИКАЦИЯ ГОТОВЫХ КЛАССОВ

Отчёт о лабораторной работе № 1

по дисциплине «Современные средства программирования»

Выполнил: студент гр. 434-М1

Колпаков Н.А.

« » 2024 г.

Проверил: доцент каф. АСУ

Алферов С. М.

« » 2024 г.

Томск 2024

# Оглавление

1. [Цели и задачи 3](#_bookmark0)
2. [Ход работы 4](#_bookmark0)

[Вывод 5](#_bookmark1)

## 1 ЦЕЛИ И ЗАДАЧИ

* 1. Цель работы:

Получить навык использования имеющихся классов для создания новых классов.

* 1. Задания:

Взять готовый класс и создать на основе его класс-наследник с дополнительными возможностями. В качестве класса выбран класс анимации изображений.

## 2 ХОД РАБОТЫ

Листинг программы:

import sys

import os

from PyQt5.QtCore import Qt, QTimer

from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QHBoxLayout, QSlider

from PyQt5.QtGui import QPixmap

class ImageAnimationWidget(QWidget):

    def \_\_init\_\_(self, image\_folder):

        super().\_\_init\_\_()

        self.images = self.load\_images(image\_folder)

        self.current\_image\_index = 0

        self.direction = 1

        self.image\_label = QLabel(self)

        self.image\_label.setAlignment(Qt.AlignCenter)

        self.image\_label.setFixedSize(600, 600)

        self.update\_image()

        self.slider = QSlider(Qt.Horizontal, self)

        self.slider.setRange(-100, 100)

        self.slider.setValue(0)

        self.slider.valueChanged.connect(self.update\_animation\_speed)

        self.timer = QTimer(self)

        self.timer.timeout.connect(self.animate)

        self.update\_animation\_speed(0)

        self.timer.start()

        layout = QVBoxLayout()

        layout.addWidget(self.image\_label)

        layout.addWidget(self.slider)

        self.setLayout(layout)

    def load\_images(self, image\_folder):

        return sorted([os.path.join(image\_folder, img) for img in os.listdir(image\_folder) if img.endswith('jpg')])

    def update\_image(self):

        pixmap = QPixmap(self.images[self.current\_image\_index])

        self.image\_label.setPixmap(pixmap.scaled(self.image\_label.size(), Qt.KeepAspectRatio, Qt.SmoothTransformation))

    def animate(self):

        self.current\_image\_index += self.direction

        if self.current\_image\_index >= len(self.images):

            self.current\_image\_index = 0

        elif self.current\_image\_index < 0:

            self.current\_image\_index = len(self.images) - 1

        self.update\_image()

    def update\_animation\_speed(self, value):

        if value == 0:

            self.timer.stop()

        else:

            self.direction = 1 if value > 0 else -1

            speed = max(16, 100 - abs(value))

            self.timer.start(speed)

class MainWindow(QWidget):

    def \_\_init\_\_(self, image\_folders, titles):

        super().\_\_init\_\_()

        self.setWindowTitle("Мульти-Анимация")

        self.setGeometry(100, 100, 600, 1000)

        main\_layout = QVBoxLayout()

        for image\_folder, title in zip(image\_folders, titles):

            animation\_block = ImageAnimationWidget(image\_folder)

            labeled\_block = self.create\_labeled\_block(animation\_block, title)

            main\_layout.addLayout(labeled\_block)

        self.setLayout(main\_layout)

    def create\_labeled\_block(self, animation\_block, title):

        block\_layout = QVBoxLayout()

        label = QLabel(title)

        label.setAlignment(Qt.AlignCenter)

        block\_layout.addWidget(label)

        block\_layout.addWidget(animation\_block)

        return block\_layout

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

    app = QApplication(sys.argv)

    image\_folders = [

        'C:/Users/Kolpachkov/Downloads/qiwi',

        'C:/Users/Kolpachkov/Downloads/2',

        'C:/Users/Kolpachkov/Downloads/3'

    ]

    titles = ["Омериканский АрЁль", "КИВИ", "ХАСБИК"]

    window = MainWindow(image\_folders, titles)

    window.show()

    sys.exit(app.exec\_())

Результат выполнения работы представлен на рисунке 1.1.

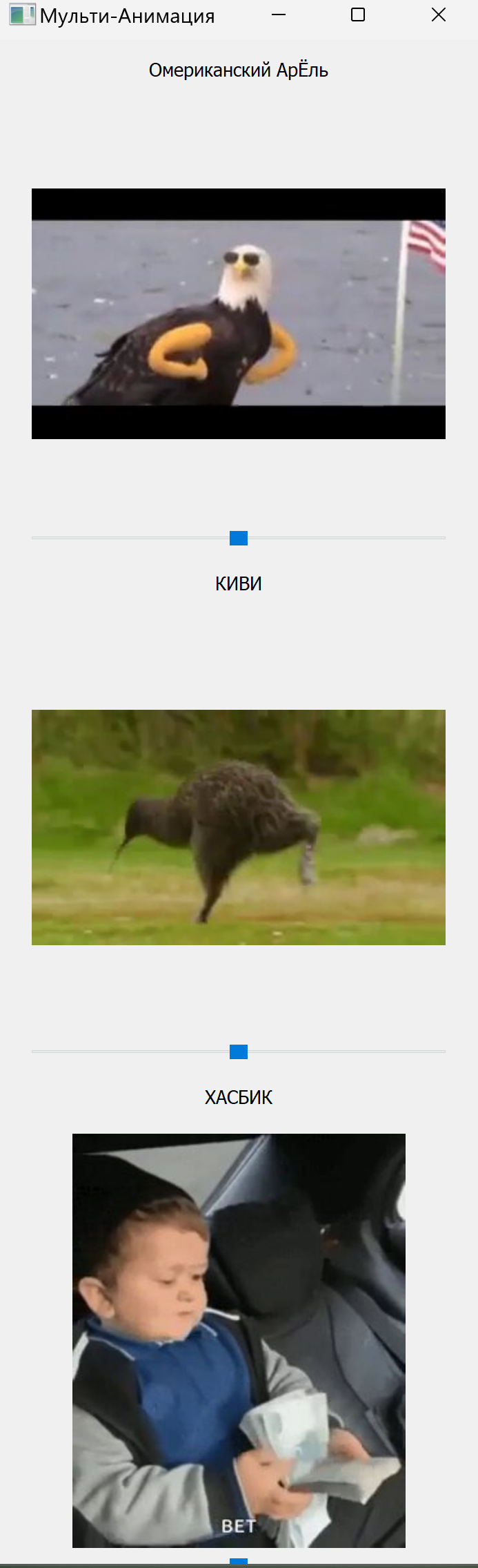


Рисунок 1.1 — Результат работы программы

UML диаграмма представлена на рисунке 1.2.



Рисунок 1.2 — UML диаграмма классов

* MainWindow создаёт объекты типа ImageAnimationWidget и управляет их жизненным циклом.
* Каждый экземпляр ImageAnimationWidget добавляется в интерфейс MainWindow как часть пользовательского интерфейса.
* Когда объект MainWindow уничтожается, все связанные объекты ImageAnimationWidget также будут уничтожены.
* Таким образом, объекты ImageAnimationWidget не могут существовать без объекта MainWindow, что делает их частью композиции.
* ImageAnimationWidget содержит объекты QLabel, QSlider и QTimer.
* Эти объекты создаются и управляются непосредственно внутри ImageAnimationWidget. Они являются частью его внутренней структуры.
* Когда объект ImageAnimationWidget уничтожается, связанные с ним объекты QLabel, QSlider и QTimer также будут уничтожены.
* Таким образом, существует композиция между ImageAnimationWidget и его внутренними элементами интерфейса.

## Вывод

В ходе работы были реализованы следующие классы:

* ImageAnimationWidget: Наследуется от базового класса QWidget.

Этот класс отвечает за отображение и анимацию изображений. Он включает в себя компоненты для отображения изображений, управления скоростью анимации и создания таймера, который обновляет изображение через определенные интервалы времени.

* MainWindow: Наследуется от класса QWidget.

Этот класс создаёт главное окно программы, в котором размещаются несколько объектов типа ImageAnimationWidget. Он организует структуру интерфейса и обеспечивает правильное расположение анимационных виджетов в окне.

Оба класса наследуют функциональность от QWidget, что позволяет им интегрироваться в графический интерфейс программы.