Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5
По дисциплине: «ОСиСП»
Тема: «Многопоточность»

Выполнил:
Студент 3 курса
Группы ПО-7
Комиссаров А.Е.
Проверил:
Булей Е.В.

2022

Цель: познакомиться с возможностями, предлагаемыми фреймворком Qt, для разработки многопоточных приложений.

Общее задание:

Задание

1) Основное задание заключается в доработке функционала обновления, разработка которого производилась в ЛР №4. Нужно интегрировать указанную функцию в само приложение, без использования стороннего клиента. При этом серверная часть приложения остается без изменений (возможны некоторые доработки сервера, без изменения общей клиент-серверной архитектуры);

2) Проверка обновления должна осуществляться автоматически по таймеру (QTimer) либо по непосредственному запросу пользователя. Предусмотреть выбор из меню политики обновления (с пользовательским подтверждением, без подтверждения/автоматически);

3) Сам процесс обновления должен осуществляться с использованием отдельного потока (QThread) с минимальной вовлечённостью пользователя;

4) Необходимо отображать прогресс обновления (для этого можно использовать строку состояния – QStatusBar);

5) Для демонстрации процесса обновления и независимой работы основного и вспомогательного потоков приложения осуществлять передачу с сервера обновления помимо основных обновляемых компонентов (в соответствии с вариантом задания) одного-двух крупных файлов с произвольным содержимым (например, видео).

6 DLL, дополнительная фигурка

Ход работы:

Изменения файла tetris_launcher.py

```
from PyQt5 import uic, QtTest
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from github import Github
import os
from threading import *
import sys
import tetris
import time as time_module

class Thread(QThread):
    _signal = pyqtSignal(int)
    def __init__(self,contents,repo):
        super(Thread, self).__init__()
        self.contents = contents
        self.repo = repo

    def __del__(self):
        self.wait()

    def run(self):
        len1 = len(self.contents) +1
        while self.contents:
            percentage = round((len(self.contents)/len1)*100)
            print("percentage", percentage)
            file_content = self.contents.pop(0)
            self._signal.emit(100-percentage)
```

```
            if file_content.type == "dir":
                self.contents.extend(self.repo.get_contents(file_content.path))
            else:
                if(len(file_content.path) > len(file_content.name)):
                    print("Checking " + file_content.path)
                    if not os.path.isdir(file_content.path[:(len(file_content.path) -
len(file_content.name))]):
                        print("\"" + file_content.path + "\" does not exist, creating...")
                        os.makedirs(file_content.path[:(len(file_content.path) -
len(file_content.name))])
                print("Writing " + file_content.name)
                binary_file = open(file_content.path, "wb")
                if file_content.encoding != 'base64':
                    binary_file.write(b'encoding unsupported')
                    binary_file.close()
                else:
                    binary_file.write(file_content.decoded_content)
                    binary_file.close()
        self._signal.emit(999)
...
```

Результат работы программы:

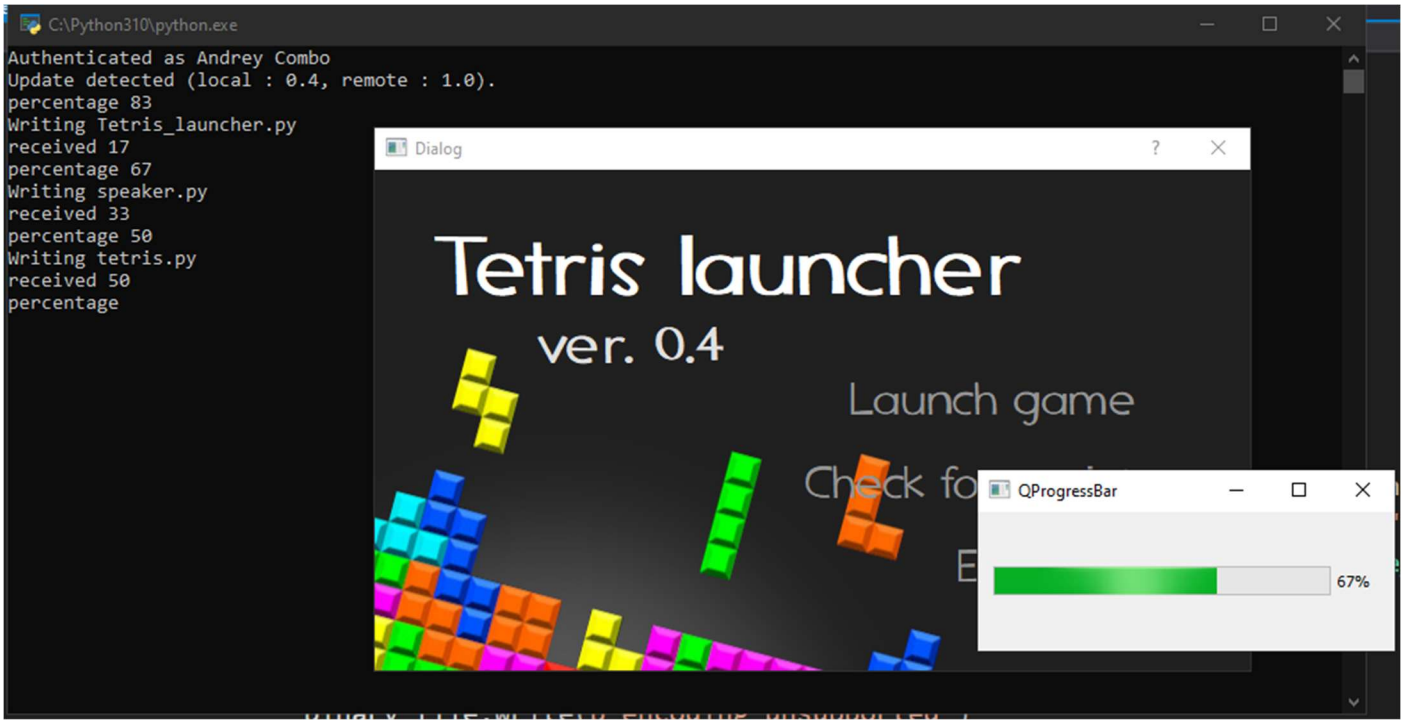

Рис. 1,2 – Результат работы программы

Вывод: я познакомился с возможностями, предлагаемыми фреймворком Qt, для разработки многопоточных приложений.