Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
По дисциплине: «ОСиСП»
Тема: «Поддержка сетевого взаимодействия программ»

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

2022

Цель: ознакомиться с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ.

Общее задание:

1) Разработать сетевую утилиту для автоматического обновления приложения, разработанного в лабораторных работах 1-3. Утилита может иметь произвольный интерфейс, определяемый ее функциональными особенностями.

2) Программа должна состоять из двух взаимодействующих частей – клиентской, устанавливаемой на компьютере с обновляемым приложением и серверной, выполняющейся на любом компьютере в локальной либо глобальной сети.

3) Клиентская часть осуществляет соединение с сервером и проверку обновлений для приложения. При наличии обновлений, все необходимые файлы загружаются и копируются в директорию с целевым приложением. В противном случае выдается соответствующее сообщение. Обработать возможные исключительные ситуации (отсутствие соединения с сервером).

4) Внести изменения в исходный проект приложения с учетом специфики загружаемых обновлений (например, хранение структуры уровня для игрового приложения в отдельном файле). **То есть обновляемые ресурсы должны быть отделены от основного приложения.**

6 DLL, дополнительная фигурка

Ход работы:

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 LauncherWindow(QDialog):
    def __init__(self):
        super(LauncherWindow, self).__init__()
        uic.loadUi("UI/Launcher.ui", self)
        self.show()
        self.buttonLaunch = self.findChild(QPushButton, "LaunchButton")
        self.buttonLaunch.clicked.connect(self.LaunchPress)
        self.buttonUpdate = self.findChild(QPushButton, "CheckUpdatesButton")
        self.buttonUpdate.clicked.connect(self.UpdatePress)
        self.buttonExit = self.findChild(QPushButton, "ExitButton")
        self.buttonExit.clicked.connect(self.ExitPress)
        self.versionText = self.findChild(QLabel, "VersionText")
        self.versionText.setText("ver. " + self.getLocalVersion())
        self.keepSoundAwake = QTimer()
        self.keepSoundAwake.setInterval(2700)
        self.keepSoundAwake.start()

    def transformToSize(self, width, height):
        old_h = self.height()
        old_w = self.width()
        end_x = round(self.x() + ((self.width() - width)/2))
        end_y = round(self.y() + ((self.height() - height)/2))
```

```
        tick_duration = 10 #default = 10
        prevH = 9999
        prevW = 9999
        x = self.x()
        y = self.y()
        i = 0
        while not (self.height() == height and self.width() == width):
            diff_h = (height - self.height())/9
            diff_w = (width - self.width())/9
            i+=1
            offset_w = old_w - self.width()
            offset_h = old_h - self.height()
            new_x = round(x + (offset_w/2))
            new_y = round(y + (offset_h/2))
            self.move(new_x, new_y)

            if(prevW != diff_w or prevH != diff_h):
                prevW = diff_w
                prevH = diff_h
                self.setFixedHeight(round(self.height() + diff_h))
                self.setFixedWidth( round(self.width() + diff_w))
            else:
                self.move(end_x, end_y)
                self.setFixedHeight(height)
                self.setFixedWidth(width)
                break
            QtTest.QTest.qWait(tick_duration)


    def showLauncher(self):
        self.show()
        self.game.hide()
        self.transformToSize(630, 360)

    #Launch button is pressed
    def LaunchPress(self):
        self.transformToSize(360, 480)
        self.hide()
        self.game = tetris.launchGame()
        self.game.closed.connect(self.showLauncher)

    #Update button is pressed
    def UpdatePress(self):
        self.Update()

    #Exit button is pressed
    def ExitPress(self):
        app.quit()

    def getLocalVersion(self):
        try:
            version_file = open("version.txt")
            version = version_file.readline()
            return version
        except OSError:
            print("could not open file")
            return None

    def getRemoteVersion(self):
        repo = self.githubInstance.get_repo(self.repository)
        return repo.get_contents("version.txt").decoded_content.decode()


    def Authenticate(self,token):
        g = Github(token)
        print("Authenticated as " + g.get_user().name)
        return g

    def noUpdateNeeded(self):
```

```
        print("No update needed.")
        self.updateNotFound = QMessageBox(
            QMessageBox.Information, "No update needed",
            "You are already running the latest version, no need to update.",
            (QMessageBox.Ok)
            )
        self.updateNotFound.exec()

    def showUpdateConfirm(self,lcVersion, rmVersion):
        '''
        Asks the user if he wants to update.
        '''
        self.updateConfirm = QMessageBox(
            QMessageBox.Information, "Update detected",
            "A new update has been detected. Your local version is " + lcVersion + ". Are you
willing to update your game version to " + rmVersion + "?",
            (QMessageBox.Yes | QMessageBox.No)
            )
        return self.updateConfirm.exec()

    def Update(self):
        self.access_token = "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"
        self.repository = "combo-wombo/OSiSP_Lab4"
        self.githubInstance = self.Authenticate(self.access_token)
        self.repo = self.githubInstance.get_repo(self.repository)
        self.contents = self.repo.get_contents("")
        local = self.getLocalVersion()
        remote = self.getRemoteVersion()
        if(local < remote):
            print("Update detected (local : " + local + ", remote : " + remote + ").")
            if(self.showUpdateConfirm(local, remote) == QMessageBox.Yes):
                self.updProg = QWidget()
                self.updProg.setWindowTitle('QProgressBar')
                self.updProg.pbar = QProgressBar(self.updProg)
                self.updProg.pbar.setValue(0)
                self.updProg.resize(300, 100)
                self.updProg.vbox = QVBoxLayout()
                self.updProg.vbox.addWidget(self.updProg.pbar)
                self.updProg.setLayout(self.updProg.vbox)
                self.updProg.show()
                self.updProg.pbar.show()
                self.thread = Thread(self.contents,self.repo)
                self.thread._signal.connect(self.signal_accept)
                self.thread.start()
        else:
            self.noUpdateNeeded()

    def signal_accept(self, msg):
        msg = int(msg)
        print("received",msg)
        if msg == 999:
            self.versionText.setText("ver. " + self.getLocalVersion())
            self.updProg.close()
            self.updProg = QMessageBox(
                QMessageBox.Information, "App updated",
                "The app has been updated to the latest version.",
                (QMessageBox.Ok)
                )
            self.updProg.exec()
        else:
            if self.updProg.pbar.value() > 99:
                self.updProg.pbar.setValue(100)
            else:
                self.updProg.pbar.setValue(msg)


#================================##================================#

app = QApplication(sys.argv)
```

```
launcher = LauncherWindow()
app.exec_()

#================================##================================#
```

Результат работы программы:

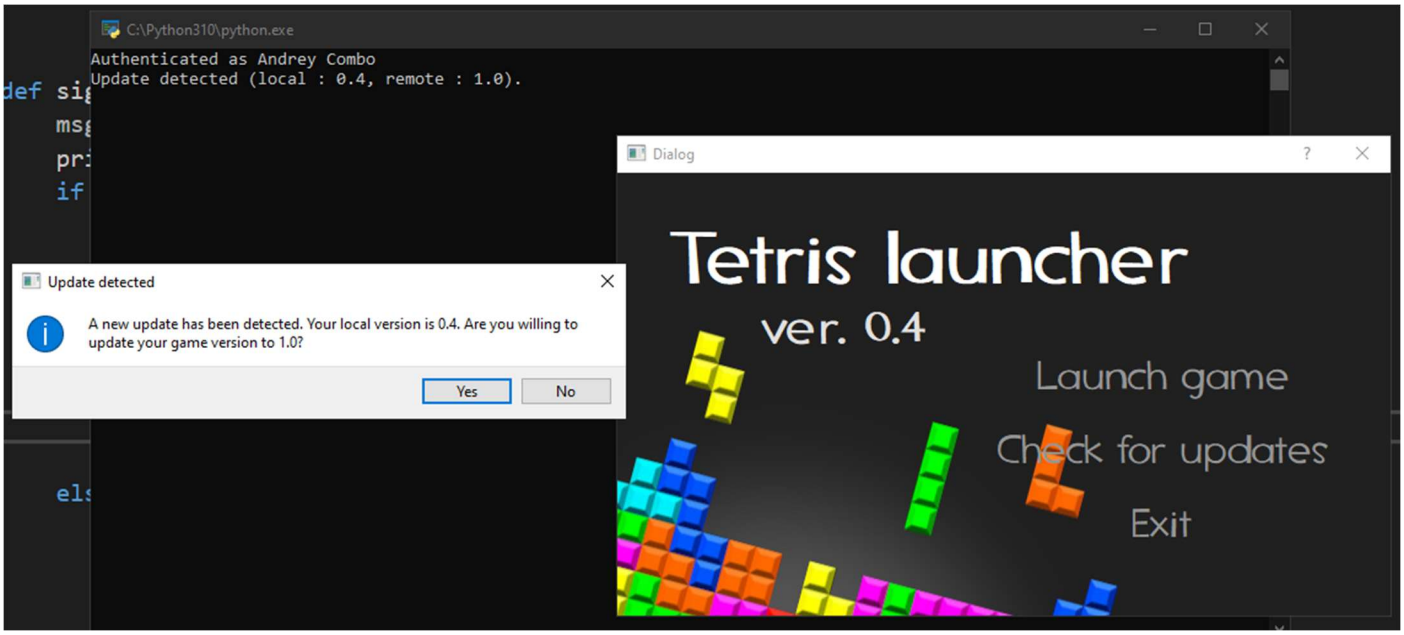

Рис. 1,2 – Результат работы программы

Вывод: я ознакомился с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ.