Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

**Отчёт**

Лабораторная работа №4

По дисциплине ОСиСП

«Клиент-сервер»

Выполнил

Студент ФЭИС

3-го курса, группы ПО-5

Бриштен Д.С.

Проверила Дряпко А.В.

Брест 2021

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

Вариант 4

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. То есть обновляемые ресурсы должны быть отделены от основного приложения.

5) Обновляемый компонент по варианту: DLL - новый уровень.

Файлы интерфейса

**fontPluginInterface2.h**

#ifndef LEVELPLUGININTERFACE\_H

#define LEVELPLUGININTERFACE\_H

#include "pluginInterface\_global2.h"

#include <QFont>

class **LevelPluginInterface**

{

public:

virtual int ***newLevel***() = 0;

};

Q\_DECLARE\_INTERFACE(LevelPluginInterface, "Pacman.LevelPluginInterface");

## #endif // FONTPLUGININTERFACE\_H

**pluginInterface\_global2.h**

#ifndef PLUGININTERFACE\_GLOBAL\_H

#define PLUGININTERFACE\_GLOBAL\_H

#include <QtCore/qglobal.h>

#if defined(PLUGININTERFACE\_LIBRARY)

# define PLUGININTERFACE\_EXPORT Q\_DECL\_EXPORT

#else

# define PLUGININTERFACE\_EXPORT Q\_DECL\_IMPORT

#endif

## #endif // PLUGININTERFACE\_GLOBAL\_H

**fontPlugin.h**

#ifndef FONTPLUGIN\_H

#define FONTPLUGIN\_H

#include <QFont>

#include <fontPluginInterface2.h>

class **FontPlugin** : public QObject, public LevelPluginInterface

{

Q\_OBJECT

Q\_INTERFACES(**LevelPluginInterface**)

Q\_PLUGIN\_METADATA(**IID** "Pacman.LevelPluginInterface" **FILE** "fontPlugin.json")

public:

virtual int ***newLevel***();

};

## #endif // FONTPLUGIN\_H

**myserver.h**

#include <QTcpSocket>

#include <QFile>

#include <QDir>

#include <QtCore>

class **myserver**: public QTcpServer

{

Q\_OBJECT

public:

**myserver**();

~***myserver***();

QTcpServer \*tcpSever;

QTcpSocket\* socket;

QByteArray Data;

public slots:

void **startServer**();

void ***incomingConnection***(int socketDescriptor);

void **sockReady**();

void **sockDisc**();

};

## #endif // MYSERVER\_H

**window.h**

#ifndef WINDOW\_H

#define WINDOW\_H

#include <QDialog>

#include <QtCore>

#include <QtGui>

#include <QMenuBar>

#include <QPainter>

#include <QString>

#include <QTcpSocket>

#include <QMessageBox>

#include <QDebug>

#include <QListIterator>

#include "pacman.h"

#include "map.h"

#include "ball.h"

#include "ghost.h"

#include "powerball.h"

#include "textdrawing.h"

#include "fontPluginInterface.h"

#include "fontPluginInterface2.h"

typedef void (\*about)(QWidget \*painter);

namespace **Ui** {

class **window**;

}

class **window** : public QDialog

{

Q\_OBJECT

public:

explicit **window**(QWidget \*parent = 0);

void **pacman\_move**();

void **ghostsmove**();

void **ghostsmove1**();

void **ghostsmove2**();

void **moveghostsinrect2**();

void **moveghostsinrect1**();

void **moveghostsinrect**();

void **checklost**();

void **delay**();

void **start\_Game**();

void **end\_Game**();

void **loadPlugins**();

void **loadLevel**();

QTcpSocket\* socket;

QByteArray Data;

Pacman \*pacman;

Ghost \*ghost;

Ghost \*ghost1;

Ghost \*ghost2;

Map \*pac\_map;

Ball \*ball;

PowerBall \*powerball;

Textdrawing \*text;

about f2;

int level = 1, fileNum = 2;

int pacx,pacy,direction,nextdirection;

int gosx,gosy,ghostdir,nextghostdir;

int gosx1,gosy1,ghostdir1,nextghostdir1;

int gosx2,gosy2,ghostdir2,nextghostdir2;

bool moving,ghostmoving,ghostmoving1,ghostmoving2;

bool scared,scared1,scared2;

int score,state3,state2,state1,state;

bool start,delayb;

bool ghoststart,ghoststart1,ghoststart2;

bool playing;

QVector<QPoint> ballpoints;

QVector<QPoint> Powerballpoints;

QThread \*sleeper;

~***window***();

public slots:

void **updater**();

void **ghostupdater**();

void **About**();

void **changeAppFont**(QFont font);

void **on\_pushButton\_clicked**();

void **sockReady**();

void **sockDisc**();

void **onRecentOpenFiles**(QAction \*action);

void **changeLevel**(int x);

void **defaultLevel**();

protected:

void ***keyPressEvent***(QKeyEvent \*event);

private:

Ui::window \*ui;

QGraphicsScene \*scene;

QTimer \*timer;

QTimer \*ghoststimer;

QMenuBar \*menuBar;

QMenu \*style, \*updated, \*levelMenu;

};

## #endif // WINDOW\_H

Файлы реализации

**fontPlugin.cpp**

#include "fontPlugin.h"

int FontPlugin::***newLevel***()

{

return 3;

}

**main.cpp**

#include <QCoreApplication>

#include "myserver.h"

int **main**(int argc, char \*argv[])

{

QCoreApplication a(*argc*, *argv*);

myserver Server;

Server.startServer();

return a.exec();

}

**myserver.cpp**

#include "myserver.h"

myserver::**myserver**(){}

myserver::~***myserver***(){}

void myserver::**startServer**()

{

if (this->listen(QHostAddress::Any,5555))

{

qDebug()<<"Listening";

}

else

{

qDebug()<<"Not listening";

}

}

void myserver::***incomingConnection***(int socketDescriptor)

{

socket = new QTcpSocket(this);

socket->*setSocketDescriptor*(socketDescriptor);

connect(socket,SIGNAL(readyRead()),this,SLOT(sockReady()));

connect(socket,SIGNAL(disconnected()),this,SLOT(sockDisc()));

qDebug()<<socketDescriptor<<" Client connected";

QByteArray data;

QFileInfoList dirContent = QDir(qApp->applicationDirPath() + "/plugins").entryInfoList(QStringList()<< "\*",QDir::Files);

for (int i = 0; i < dirContent.size(); ++i) {

QFileInfo fileInfo = dirContent.at(i);

QString fileName = dirContent.at(i).fileName();

qDebug() << fileName;

QFile file(dirContent.at(i).filePath());

file.*open*(QIODevice::ReadOnly);

data += file.readAll();

data += "\n\n\n\n\n";

file.remove();

}

if(data.isEmpty())

{

data = "empty";

}

socket->write(data);

qDebug()<<"Send client connect status - YES";

}

void myserver::**sockReady**()

{

}

void myserver::**sockDisc**()

{

qDebug()<<"Disconnect";

socket->deleteLater();

}

**window.cpp**

#include "window.h"

#include "ui\_window.h"

#include "textdrawing.h"

#include <QThread>

#include <QMenuBar>

#include <QMenu>

#include <QVBoxLayout>

#include <QFontDialog>

#include <QDebug>

window::**window**(QWidget \*parent) :

QDialog(*parent*),

ui(new Ui::window)

{

score=0;

gosx=450/2;

gosy=480/2;

gosx1=450/2;

gosy1=480/2;

gosx2=450/2;

gosy2=480/2;

pacx=410/2;

pacy=360;

ghostmoving=false;

ghostmoving1=false;

ghostmoving2=false;

scared=false;

scared1=false;

scared2=false;

start=false;

direction=0;

moving=false;

delayb=false;

playing=false;

ui->setupUi(this);

scene = new QGraphicsScene(this);

ui->graphicsView->setScene(*scene*);

ui->graphicsView->setRenderHint(QPainter::Antialiasing);

scene->setSceneRect(0,0,640,480);

ui->graphicsView->setSceneRect(scene->sceneRect());

pacman=new Pacman;

pacx=410/2;

pacy=410;

text=new Textdrawing;

text->over=false;

scene->addItem(*text*);

text->setZValue(7);

menuBar = new QMenuBar();

QAction \*aboutAction = menuBar->addAction(tr("О программе"));

connect(aboutAction, SIGNAL(triggered()), this, SLOT(About()));

style = new QMenu(tr("Стили"), this);

menuBar->addMenu(*style*);

QAction \*updateAction = menuBar->addAction(tr("Обновления"));

connect(updateAction, SIGNAL(triggered()), this, SLOT(on\_pushButton\_clicked()));

levelMenu = new QMenu(tr("Уровни"), this);

menuBar->addMenu(*levelMenu*);

QAction\* actionLevelEasy = levelMenu->addAction(tr("Лёгкий"));

connect(actionLevelEasy, SIGNAL(triggered()), this, SLOT(defaultLevel()));

//connect(levelMenu, SIGNAL(triggered(QAction\*)), this, SLOT(onRecentOpenFiles(QAction\*)));

QVBoxLayout \*mainLayout = new QVBoxLayout;

mainLayout->setMenuBar(*menuBar*);

setLayout(*mainLayout*);

loadPlugins();

socket = new QTcpSocket(this);

connect(socket,SIGNAL(readyRead()),this,SLOT(sockReady()));

connect(socket,SIGNAL(disconnected()),this,SLOT(sockDisc()));

loadLevel();

}

void window::**onRecentOpenFiles**(QAction \*action)

{

qDebug() << action;

}

void window::**on\_pushButton\_clicked**()

{

socket->*connectToHost*("127.0.0.1",5555);

}

void window::**sockDisc**()

{

socket->deleteLater();

}

void window::**sockReady**()

{

if (socket->*waitForConnected*(500))

{

QByteArray testing;

while (socket->*waitForReadyRead*(3000))

{

while(socket->*bytesAvailable*() > 0)

{

testing.append(socket->readAll());

socket->flush();

}

}

if(testing != "")

{

QTextCodec \*codec = QTextCodec::codecForName("CP866");

QString s = codec->toUnicode(testing);

QStringList list = s.split("\n\n\n\n\n");

for (int i = 0; i < list.size() - 1; i++) {

QFile target;

target.setFileName("D:/5 semester/OSISP/Lab4/Pacman-build/debug/updates/level" + QString::number(fileNum) + ".dll");

if (!target.*open*(QIODevice::WriteOnly | QIODevice::Append)) {

qDebug() << "Can't open file for written";

return;

}

QByteArray output;

output.append(codec->fromUnicode(list[i]));

target.write(output);

target.*close*();

output.clear();

fileNum += 1;

qDebug() << "file size: " << target.*size*();

qDebug() << "Finished!";

}

QMessageBox::information(this, "Информация", "Обновление успешно загружено");

loadLevel();

}

else

{

QMessageBox::information(this, "Информация", "Установлены последние обновления");

}

}

}

void window::**defaultLevel**()

{

qDebug() << "defaultLevel";

level = 1;

}

void window::**changeLevel**(int x)

{

qDebug() << "Cработал слот для значения: " << x;

level = x;

}

void window::**loadLevel**()

{

QDir dir(qApp->applicationDirPath() + "/updates");

QStringList filter("\*.dll");

QPluginLoader loader;

foreach(QFileInfo info, dir.entryInfoList(filter))

{

loader.setFileName(info.absoluteFilePath());

LevelPluginInterface\* levelPlugin = qobject\_cast<LevelPluginInterface\*>(*loader.instance()*);

if(levelPlugin)

{

QString name = loader.metaData().value("MetaData").toObject().value("fontName").toString();

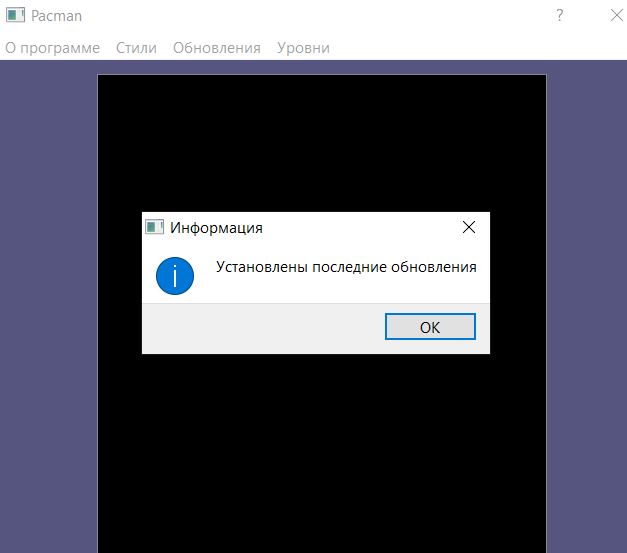
QAction\* actionFontPlugin = levelMenu->addAction(name);

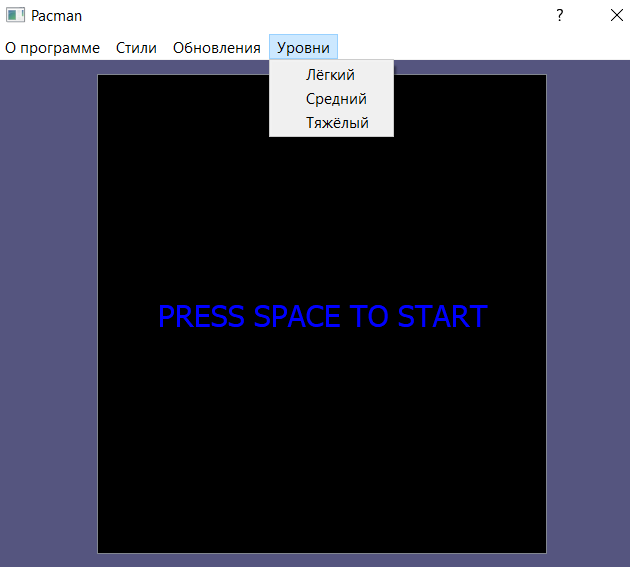
connect(actionFontPlugin, &QAction::triggered, this, [this, levelPlugin]{changeLevel(levelPlugin->*newLevel*());});

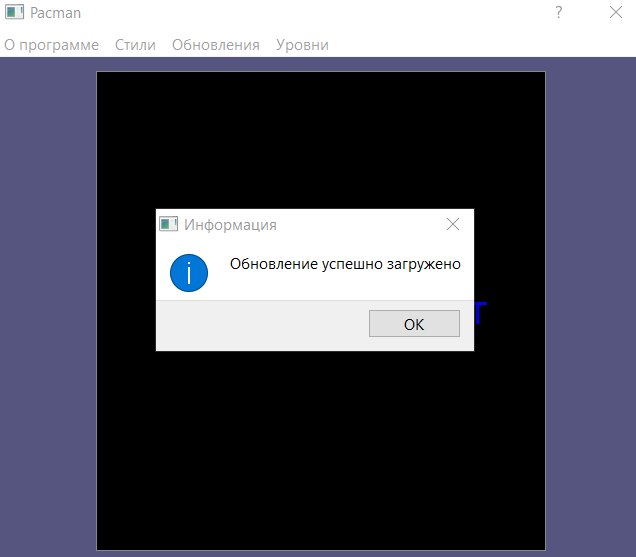
}

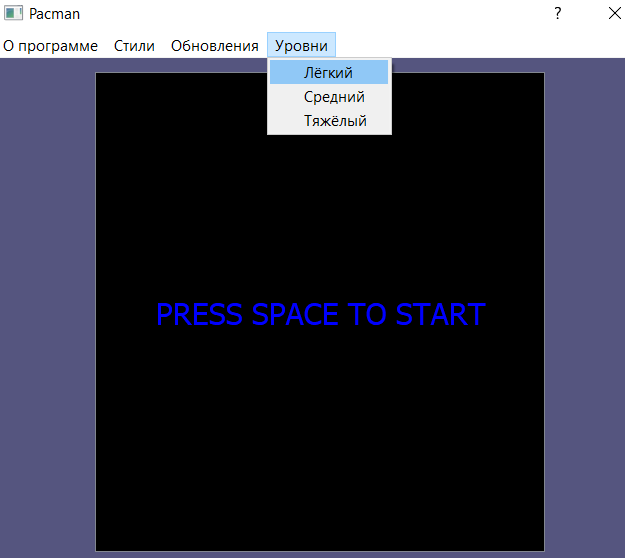
}

## }

****

****

****

****

Вывод: Разработана сетевая утилита для автоматического обновления приложения.