Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №4-5

за 5 семестр

По дисциплине: «ОСиСП»

Выполнила:

студентка 3 курса

группы ПО-4(1)

Воробей А.П.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №4-5

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

Вариант 4

Задание:

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

**Текст программы:**

**Server:**

1. **mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QJsonObject>

#include <QJsonDocument>

#include <QJsonArray>

#include <QFile>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->pushButton\_stopServer->setVisible(false);

ui->pushButton\_update->setVisible(false);

this->server = new ServerStuff(this);

connect(this->server, &ServerStuff::gotNewMesssage,

this, &MainWindow::gotNewMesssage);

connect(this->server->tcpServer, &QTcpServer::newConnection,

this, &MainWindow::smbConnectedToServer);

connect(this->server, &ServerStuff::smbDisconnected,

this, &MainWindow::smbDisconnectedFromServer);

this->fileServer = new ServerStuff(this);

this->setVersion(this->getDefaultVersion());

}

MainWindow::~MainWindow()

{

delete server;

delete ui;

}

void MainWindow::on\_pushButton\_startServer\_clicked()

{

ui->pushButton\_stopServer->setVisible(true);

ui->pushButton\_update->setVisible(true);

ui->pushButton\_startServer->setVisible(false);

if (!server->tcpServer->listen(QHostAddress::Any, 6547))

{

ui->textEdit\_log->append(tr("<font color=\"red\"><b>Error!</b> The port is taken by some other service.</font>"));

return;

}

connect(server->tcpServer, &QTcpServer::newConnection, server, &ServerStuff::newConnection);

if (!fileServer->tcpServer->listen(QHostAddress::Any, 6788))

{

ui->textEdit\_log->append(tr("<font color=\"red\"><b>Error!</b> The port is taken by some other service.</font>"));

return;

}

connect(fileServer->tcpServer, &QTcpServer::newConnection, fileServer, &ServerStuff::newConnection);

ui->textEdit\_log->append(tr("<font color=\"green\"><b>Server started</b>, port is openned.</font>"));

}

void MainWindow::on\_pushButton\_stopServer\_clicked()

{

if(server->tcpServer->isListening())

{

ui->pushButton\_update->setVisible(false);

ui->pushButton\_stopServer->setVisible(false);

ui->pushButton\_startServer->setVisible(true);

disconnect(server->tcpServer, &QTcpServer::newConnection, server, &ServerStuff::newConnection);

QList<QTcpSocket \*> clients = server->getClients();

for(int i = 0; i < clients.count(); i++)

{

server->sendToClient(clients.at(i), "0");

}

server->tcpServer->close();

ui->textEdit\_log->append(tr("<b>Server stopped</b>, post is closed"));

}

else

{

ui->textEdit\_log->append(tr("<b>Error!</b> Server was not running"));

}

}

void MainWindow::on\_pushButton\_testConn\_clicked()

{

if(server->tcpServer->isListening())

{

ui->textEdit\_log->append(

QString("%1 %2")

.arg("Server is listening, number of connected clients:")

.arg(QString::number(server->getClients().count()))

);

}

else

{

ui->textEdit\_log->append(

QString("%1 %2")

.arg("Server is not listening, number of connected clients:")

.arg(QString::number(server->getClients().count()))

);

}

}

void MainWindow::smbConnectedToServer()

{

ui->textEdit\_log->append(tr("Somebody has connected"));

}

void MainWindow::smbDisconnectedFromServer()

{

ui->textEdit\_log->append(tr("Somebody has disconnected"));

}

void MainWindow::gotNewMesssage(QString msg)

{

ui->textEdit\_log->append(QString("New message: %1").arg(msg));

if (msg.contains("version:"))

{

QList<QTcpSocket \*> clients = server->getClients();

this->server->sendToClient(clients.at(0), "version:" + this->mVersion);

}

else if (msg == "OK ON UPDATE")

{

QList<QTcpSocket \*> clients = server->getClients();

// this->server->sendToClient(clients.at(0), "file:" + QApplication::applicationDirPath() + "/helper\_class.dll");

}

}

QString MainWindow::getDefaultVersion()

{

QString val;

QFile file;

file.setFileName("server.json");

file.open(QIODevice::ReadOnly | QIODevice::Text);

val = file.readAll();

file.close();

QString version = "1";

if (val != "") {

QJsonDocument doc = QJsonDocument::fromJson(val.toUtf8());

QJsonObject json = doc.object();

version = json["version"].toString();

}

return version;

}

void MainWindow::setVersion(QString version)

{

QJsonObject recordObject;

recordObject.insert("version", QJsonValue::fromVariant(version));

QJsonDocument doc(recordObject);

QString jsonString = doc.toJson(QJsonDocument::Indented);

QFile file;

file.setFileName("server.json");

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream stream( &file );

stream << jsonString;

file.close();

this->mVersion = version;

}

QString MainWindow::getVersion()

{

return this->mVersion;

}

void MainWindow::on\_pushButton\_update\_clicked()

{

this->setVersion(QString::number((this->getVersion()).toInt() + 1));

if(server->tcpServer->isListening())

{

QList<QTcpSocket \*> clients = server->getClients();

for(int i = 0; i < clients.count(); i++)

{

server->sendToClient(clients.at(i), "version:" + this->mVersion);

}

}

}

1. **mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QDebug>

#include <QString>

#include <QThread>

#include "serverstuff.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

void setVersion(QString version);

QString getVersion();

QString getDefaultVersion();

void sendFile();

private slots:

void on\_pushButton\_stopServer\_clicked();

void on\_pushButton\_startServer\_clicked();

void on\_pushButton\_testConn\_clicked();

void smbConnectedToServer();

void smbDisconnectedFromServer();

void gotNewMesssage(QString msg);

void on\_pushButton\_update\_clicked();

private:

Ui::MainWindow \*ui;

ServerStuff \*server;

ServerStuff \*fileServer;

QString mVersion;

};

#endif // MAINWINDOW\_H

1. **serverstuff.cpp**

#include "serverstuff.h"

#include <QFile>

#include <QDebug>

#include <iostream>

ServerStuff::ServerStuff(QObject \*pwgt) : QObject(pwgt), m\_nNextBlockSize(0)

{

tcpServer = new QTcpServer(this);

this->serverIsVacant = true;

}

QList<QTcpSocket \*> ServerStuff::getClients()

{

return clients;

}

void ServerStuff::newConnection()

{

QTcpSocket \*clientSocket = tcpServer->nextPendingConnection();

connect(clientSocket, &QTcpSocket::disconnected, clientSocket, &QTcpSocket::deleteLater);

connect(clientSocket, &QTcpSocket::readyRead, this, &ServerStuff::readClient);

connect(clientSocket, &QTcpSocket::disconnected, this, &ServerStuff::gotDisconnection);

clients << clientSocket;

sendToClient(clientSocket, "Reply: connection established");

}

void ServerStuff::readClient()

{

QTcpSocket \*clientSocket = (QTcpSocket\*)sender();

QDataStream in(clientSocket);

while(true)

{

if (!m\_nNextBlockSize) {

if (clientSocket->bytesAvailable() < sizeof(quint16)) { break; }

in >> m\_nNextBlockSize;

}

if (clientSocket->bytesAvailable() < m\_nNextBlockSize)

{

break;

}

QString str;

in >> str;

emit gotNewMesssage(str);

m\_nNextBlockSize = 0;

}

}

void ServerStuff::gotDisconnection()

{

clients.removeAt(clients.indexOf((QTcpSocket\*)sender()));

emit smbDisconnected();

}

qint64 ServerStuff::sendToClient(QTcpSocket\* socket, const QString& str)

{

QByteArray arrBlock;

QDataStream out(&arrBlock, QIODevice::WriteOnly);

out << quint16(0) << str;

out.device()->seek(0);

out << quint16(arrBlock.size() - sizeof(quint16));

return socket->write(arrBlock);

}

qint64 ServerStuff::sendFileToClient(QTcpSocket\* socket, const QString& fileName)

{

this->serverIsVacant = false;

QByteArray arrBlock;

QFile file(fileName);

file.open(QIODevice::ReadOnly);

arrBlock = file.readAll();

file.close();

std::cout <<fileName.toStdString() << std::endl;

qDebug() << fileName;

this->serverIsVacant = true;

return socket->write(arrBlock);

}

bool ServerStuff::isVacant()

{

return this->serverIsVacant;

}

1. **serverstuff.cpp**

#ifndef SERVERSTUFF\_H

#define SERVERSTUFF\_H

#include <QTcpServer>

#include <QTcpSocket>

#include <QDataStream>

#include <QList>

class ServerStuff : public QObject

{

Q\_OBJECT

public:

ServerStuff(QObject \*pwgt);

QTcpServer \*tcpServer;

QList<QTcpSocket \*> getClients();

bool isVacant();

public slots:

virtual void newConnection();

void readClient();

void gotDisconnection();

qint64 sendToClient(QTcpSocket \*socket, const QString &str);

qint64 sendFileToClient(QTcpSocket \*socket, const QString &fileName);

signals:

void gotNewMesssage(QString msg);

void smbDisconnected();

private:

quint16 m\_nNextBlockSize;

QList<QTcpSocket\*> clients;

bool serverIsVacant;

};

#endif // SERVERSTUFF\_H

1. **Обновления по таймеру:**

this->mTimerUpdates = new QTimer(this);

connect(this->mTimerUpdates, SIGNAL(timeout()), this, SLOT(on\_CheckUpdateButton\_clicked()));

this->mTimerUpdates->start(20000);

**Измененные файлы игры:**

1. **pacman.cpp**

#include "pacman.h"

#include "mainwindow.h"

pacman::pacman(){

pics[0].push\_back(QPixmap(":/img2/img2/pacmanu1"));

pics[0].push\_back(QPixmap(":/img2/img2/pacmanu2"));

pics[1].push\_back(QPixmap(":/img2/img2/pacmand1"));

pics[1].push\_back(QPixmap(":/img2/img2/pacmand2"));

pics[2].push\_back(QPixmap(":/img2/img2/pacmanl1"));

pics[2].push\_back(QPixmap(":/img2/img2/pacmanl2"));

pics[3].push\_back(QPixmap(":/img2/img2/pacmanr1"));

pics[3].push\_back(QPixmap(":/img2/img2/pacmanr2"));

setPixmap(pics[3][0].scaled(50, 50, Qt::KeepAspectRatio));

direction = 3;

keyDirection = 3;

picsArg = 0;

}

void pacman::keyPressEvent(QKeyEvent \*event){

if(event->key() == Qt::Key\_Up){

keyDirection = 0;

if(direction == 1){

direction = 0;

}

}

if(event->key() == Qt::Key\_Down){

keyDirection = 1;

if(direction == 0){

direction = 1;

}

}

if(event->key() == Qt::Key\_Left){

keyDirection = 2;

if(direction == 3){

direction = 2;

}

}

if(event->key() == Qt::Key\_Right){

keyDirection = 3;

if(direction == 2){

direction = 3;

}

}

}

void pacman::move(){

if(direction == 0){

setPos(x(), y() - 2);

}

if(direction == 1){

setPos(x(), y() + 2);

}

if(direction == 2){

setPos(x() - 2, y());

}

if(direction == 3){

setPos(x() + 2, y());

}

if(canChange() && direction != keyDirection){

direction = keyDirection;

setPixmap(pics[direction][picsArg].scaled(50, 50, Qt::KeepAspectRatio));

}

if(isWall()){

direction = 4;

keyDirection = 4;

}

if(x() > 792){

setPos(-40, 342);

}

if(x() < -40){

setPos(792, 342);

}

}

void pacman::changePics(){

picsArg = picsArg \* -1 + 1;

if(direction < 4)

setPixmap(pics[direction][picsArg].scaled(50, 50, Qt::KeepAspectRatio));

}

bool pacman::canChange(){

if(keyDirection == 4) return true;

int i = 0, j = 0;

bool checkX = false, checkY = false;

if(x() == 10.0){

i = 0;

checkX = true;

}

if(x() == 74.0){

i = 1;

checkX = true;

}

if(x() == 170.0){

i = 2;

checkX = true;

}

if(x() == 252.0){

i = 3;

checkX = true;

}

if(x() == 334.0){

i = 4;

checkX = true;

}

if(x() == 418.0){

i = 5;

checkX = true;

}

if(x() == 500.0){

i = 6;

checkX = true;

}

if(x() == 584.0){

i = 7;

checkX = true;

}

if(x() == 678.0){

i = 8;

checkX = true;

}

if(x() == 740.0){

i = 9;

checkX = true;

}

if(y() == 8.0){

j = 0;

checkY = true;

}

if(y() == 110.0){

j = 1;

checkY = true;

}

if(y() == 194.0){

j = 2;

checkY = true;

}

if(y() == 272.0){

j = 3;

checkY = true;

}

if(y() == 342.0){

j = 4;

checkY = true;

}

if(y() == 416.0){

j = 5;

checkY = true;

}

if(y() == 492.0){

j = 6;

checkY = true;

}

if(y() == 576.0){

j = 7;

checkY = true;

}

if(y() == 660.0){

j = 8;

checkY = true;

}

if(y() == 742.0){

j = 9;

checkY = true;

}

if(checkX && checkY && mapDirection[i][j][keyDirection]){

return true;

}

else{

return false;

}

}

bool pacman::isWall(){

if(direction == 4) return true;

int i = 0, j = 0;

bool checkX = false, checkY = false;

if(x() == 10.0){

i = 0;

checkX = true;

}

if(x() == 74.0){

i = 1;

checkX = true;

}

if(x() == 170.0){

i = 2;

checkX = true;

}

if(x() == 252.0){

i = 3;

checkX = true;

}

if(x() == 334.0){

i = 4;

checkX = true;

}

if(x() == 418.0){

i = 5;

checkX = true;

}

if(x() == 500.0){

i = 6;

checkX = true;

}

if(x() == 584.0){

i = 7;

checkX = true;

}

if(x() == 678.0){

i = 8;

checkX = true;

}

if(x() == 740.0){

i = 9;

checkX = true;

}

if(y() == 8.0){

j = 0;

checkY = true;

}

if(y() == 110.0){

j = 1;

checkY = true;

}

if(y() == 194.0){

j = 2;

checkY = true;

}

if(y() == 272.0){

j = 3;

checkY = true;

}

if(y() == 342.0){

j = 4;

checkY = true;

}

if(y() == 416.0){

j = 5;

checkY = true;

}

if(y() == 492.0){

j = 6;

checkY = true;

}

if(y() == 576.0){

j = 7;

checkY = true;

}

if(y() == 660.0){

j = 8;

checkY = true;

}

if(y() == 742.0){

j = 9;

checkY = true;

}

if(checkX && checkY && !mapDirection[i][j][direction]){

return true;

}

else{

return false;

}

}

1. **mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFile>

#include <QRadioButton>

#include <QPushButton>

#include <QMessageBox>

#include <QJsonDocument>

#include <QJsonObject>

//#include <about/include/about.h>

//#include <helper\_2/include/helper\_2.h>

//#include <help\_class/include/mybutton.h>

#include <QFontDialog>

#include <QSettings>

void MainWindow::menu\_style\_game()

{

bool Changed;

QFont newFont = QFontDialog::getFont(&Changed);

if (Changed) {

QApplication::setFont(newFont);

QSettings settings(this);

settings.setValue("VIEWF", newFont);

}

}

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

mClient = new ClientStuff("localhost", 6547, 6788);

connect(this->mClient, &ClientStuff::hasReadSome, this, &MainWindow::receivedSomething);

this->mTimerUpdates = new QTimer(this);

connect(this->mTimerUpdates, SIGNAL(timeout()), this, SLOT(on\_CheckUpdateButton\_clicked()));

this->mTimerUpdates->start(20000);

ui->setupUi(this);

//ui->pushButton\_2->setText(Exit());

// ui->label->setText(ButtonName());

ui->Win->setVisible(false);

ui->Lose->setVisible(false);

scene = new QGraphicsScene(0, 0, ui->graphicsView->width() - 2, ui->graphicsView->height() - 2);

//scene->addPixmap(QPixmap(":/img/img/background"));

scene->addPixmap(QPixmap(":/img2/img2/background"));

// scene->addPixmap(QPixmap("D:/LABS/OOTPiSP/game/pd2-pacman/img2/background"));

ui->graphicsView->setScene(scene);

for(int i = 0; i < 29; i++){

for(int j = 0; j < 26; j++){

Smallpoint[i][j] = new smallpoint(18 + j \* 28.5, 8 + i \* 26.1, i, j);

scene->addItem(Smallpoint[i][j]);

}

}

QMenu\* menu\_style = new QMenu("&Style");

QAction\* menu\_style\_action = new QAction("&Style",menu\_style);

menu\_style->addAction(menu\_style\_action);

menu\_style\_action->setShortcut(QKeySequence(Qt::Key\_F6));

menuBar()->addMenu(menu\_style);

connect(menu\_style\_action, &QAction::triggered, this, &MainWindow::menu\_style\_game);

setVersion(getDefaultVersion());

smallpointNum = 0;

Bigpoint[0] = new bigpoint();

Bigpoint[0]->setPos(18, 60.2);

scene->addItem(Bigpoint[0]);

Bigpoint[1] = new bigpoint();

Bigpoint[1]->setPos(730.5, 60.2);

scene->addItem(Bigpoint[1]);

Bigpoint[2] = new bigpoint();

Bigpoint[2]->setPos(18, 582.2);

scene->addItem(Bigpoint[2]);

Bigpoint[3] = new bigpoint();

Bigpoint[3]->setPos(730.5, 582.2);

scene->addItem(Bigpoint[3]);

bigpointNum = 0;

Pacman = new pacman();

Pacman->setPos(376, 576);

Pacman->setFlag(QGraphicsItem::ItemIsFocusable);

Pacman->setFocus();

scene->addItem(Pacman);

ghost1 = new red();

scene->addItem(ghost1);

ghost2 = new pink();

scene->addItem(ghost2);

ghost3 = new cyan();

scene->addItem(ghost3);

ghost4 = new orange();

scene->addItem(ghost4);

point = new QLabel(this);

point->setGeometry(0, 0, 49, 17);

point->setFont(QFont("Ubuntu Regular", 14));

QPalette pa;

pa.setColor(QPalette::WindowText, Qt::darkGreen);

point->setPalette(pa);

point->setVisible(false);

timer1 = new QTimer(this);

connect(timer1, SIGNAL(timeout()), this, SLOT(allMove()));

timer1->start(13);

timer2 = new QTimer(this);

connect(timer2, SIGNAL(timeout()), this, SLOT(allChangePics()));

timer2->start(100);

timer3 = new QTimer(this);

connect(timer3, SIGNAL(timeout()), this, SLOT(bigpointChangePics()));

timer3->start(200);

modeRed = modePink = modeCyan = modeOrange = 0;

addpoint = 200;

pause = false;

isBonus2 = false;

timer8 = new QTimer(this);

timer8->setSingleShot(true);

connect(timer8, SIGNAL(timeout()), this, SLOT(pauseTime()));

timer8->start(2000);

timer9 = new QTimer(this);

connect(timer9, SIGNAL(timeout()), this, SLOT(slowMove()));

timer9->start(26);

timer10 = new QTimer(this);

timer10->setSingleShot(true);

connect(timer10, SIGNAL(timeout()), this, SLOT(cyanStartSlot()));

timer10->start(7000);

timer11 = new QTimer(this);

timer11->setSingleShot(true);

connect(timer11, SIGNAL(timeout()), this, SLOT(orangeStartSlot()));

timer11->start(17000);

pause = true;

slowRed = false;

slowPink = slowCyan = slowOrange = true;

}

void MainWindow::allMove(){

if(!pause){

Pacman->move();

if(modeRed == 0 && !slowRed)

ghost1->move();

if(modePink == 0 && !slowPink)

ghost2->move();

if(modeCyan == 0 && !slowCyan)

ghost3->move();

if(modeOrange == 0 && !slowOrange)

ghost4->move();

for(int i = 0; i < 29; i++){

for(int j = 0; j < 26; j++){

bool isCollided = Pacman->collidesWithItem(Smallpoint[i][j]);

if(isCollided){

Smallpoint[i][j]->collide();

ui->lcdNumber->display(ui->lcdNumber->value() + 10);

smallpointNum++;

}

}

}

for(int i = 0; i < 4; i++){

bool isCollided = Pacman->collidesWithItem(Bigpoint[i]);

if(isCollided){

if(!q.empty()){

int temp = q.front();

q.pop\_front();

delete timer4[temp];

delete timer5[temp];

delete timer6[temp];

delete timer7[temp];

addpoint = 200;

isBonus2 = false;

}

Bigpoint[i]->collide();

ui->lcdNumber->display(ui->lcdNumber->value() + 50);

modeRed = modePink = modeCyan = modeOrange = 1;

timer4[i] = new QTimer(this);

timer4[i]->setSingleShot(true);

connect(timer4[i], SIGNAL(timeout()), this, SLOT(bonusTime()));

timer5[i] = new QTimer(this);

timer5[i]->setSingleShot(true);

connect(timer5[i], SIGNAL(timeout()), this, SLOT(bonusTime2()));

timer6[i] = new QTimer(this);

connect(timer6[i], SIGNAL(timeout()), this, SLOT(flicker()));

timer7[i] = new QTimer(this);

connect(timer7[i], SIGNAL(timeout()), this, SLOT(ghostmove2()));

timer4[i]->start(9000);

timer5[i]->start(6500);

timer6[i]->start(200);

timer7[i]->start(20);

q.push\_back(i);

bigpointNum++;

}

}

bool isCollided1 = Pacman->collidesWithItem(ghost1);

bool isCollided2 = Pacman->collidesWithItem(ghost2);

bool isCollided3 = Pacman->collidesWithItem(ghost3);

bool isCollided4 = Pacman->collidesWithItem(ghost4);

if((isCollided1 && !modeRed) || (isCollided2 && !modePink) || (isCollided3 && !modeCyan) || (isCollided4 && !modeOrange)){

pause = true;

ui->Lose->setVisible(true);

}

//если кого-то съели

if(isCollided1 && modeRed){

modeRed = 0;

point->setText(QString::number(addpoint));

point->setGeometry(ghost1->x(), ghost1->y() + 30, 49, 17);

point->setVisible(true);

ui->lcdNumber->display(ui->lcdNumber->value() + addpoint);

pause = true;

timer8 = new QTimer(this);

timer8->setSingleShot(true);

connect(timer8, SIGNAL(timeout()), this, SLOT(getPointTime()));

timer8->start(1000);

ghost1->setPos(376, 342);

ghost1->setDirection(0);

slowRed = true;

addpoint \*= 2;

}

if(isCollided2 && modePink){

modePink = 0;

point->setText(QString::number(addpoint));

point->setGeometry(ghost2->x(), ghost2->y() + 30, 49, 17);

point->setVisible(true);

ui->lcdNumber->display(ui->lcdNumber->value() + addpoint);

pause = true;

timer8 = new QTimer(this);

timer8->setSingleShot(true);

connect(timer8, SIGNAL(timeout()), this, SLOT(getPointTime()));

timer8->start(1000);

ghost2->setPos(376, 342);

ghost2->setDirection(0);

slowPink = true;

addpoint \*= 2;

}

if(isCollided3 && modeCyan){

modeCyan = 0;

point->setText(QString::number(addpoint));

point->setGeometry(ghost3->x(), ghost3->y() + 30, 49, 17);

point->setVisible(true);

ui->lcdNumber->display(ui->lcdNumber->value() + addpoint);

pause = true;

timer8 = new QTimer(this);

timer8->setSingleShot(true);

connect(timer8, SIGNAL(timeout()), this, SLOT(getPointTime()));

timer8->start(1000);

ghost3->setPos(376, 342);

ghost3->setDirection(0);

slowCyan = true;

addpoint \*= 2;

}

if(isCollided4 && modeOrange){

modeOrange = 0;

point->setText(QString::number(addpoint));

point->setGeometry(ghost4->x(), ghost4->y() + 30, 49, 17);

point->setVisible(true);

ui->lcdNumber->display(ui->lcdNumber->value() + addpoint);

pause = true;

timer8 = new QTimer(this);

timer8->setSingleShot(true);

connect(timer8, SIGNAL(timeout()), this, SLOT(getPointTime()));

timer8->start(1000);

ghost4->setPos(376, 342);

ghost4->setDirection(0);

slowOrange = true;

addpoint \*= 2;

}

//когда все съедены конец игры

if(smallpointNum == 242 && bigpointNum == 4){

pause = true;

ui->Win->setVisible(true);

}

}

}

void MainWindow::allChangePics(){

if(!pause){

Pacman->changePics();

if(modeRed == 0){

ghost1->changePics();

}

else if(modeRed == 1){

ghost1->changePics2();

}

else if(modeRed == 2){

ghost1->changePics3();

}

if(modePink == 0){

ghost2->changePics();

}

else if(modePink == 1){

ghost2->changePics2();

}

else if(modePink == 2){

ghost2->changePics3();

}

if(modeCyan == 0){

ghost3->changePics();

}

else if(modeCyan == 1){

ghost3->changePics2();

}

else if(modeCyan == 2){

ghost3->changePics3();

}

if(modeOrange == 0){

ghost4->changePics();

}

else if(modeOrange == 1){

ghost4->changePics2();

}

else if(modeOrange == 2){

ghost4->changePics3();

}

}

}

void MainWindow::bigpointChangePics(){

if(!pause){

Bigpoint[0]->changePics();

Bigpoint[1]->changePics();

Bigpoint[2]->changePics();

Bigpoint[3]->changePics();

}

}

void MainWindow::bonusTime(){

int i = q.front();

q.pop\_front();

modeRed = modePink = modeCyan = modeOrange = 0;

isBonus2 = false;

addpoint = 200;

delete timer4[i];

delete timer5[i];

delete timer6[i];

delete timer7[i];

}

void MainWindow::bonusTime2(){

isBonus2 = true;

}

void MainWindow::flicker(){

if(!pause){

if(isBonus2){

if(modeRed == 1){

modeRed = 2;

}

else if(modeRed == 2){

modeRed = 1;

}

if(modePink == 1){

modePink = 2;

}

else if(modePink == 2){

modePink = 1;

}

if(modeCyan == 1){

modeCyan = 2;

}

else if(modeCyan == 2){

modeCyan = 1;

}

if(modeOrange == 1){

modeOrange = 2;

}

else if(modeOrange == 2){

modeOrange = 1;

}

}

}

}

void MainWindow::ghostmove2(){

if(!pause){

if((modeRed == 1 || modeRed == 2) && !slowRed)

ghost1->move();

if((modePink == 1 || modePink == 2) && !slowPink)

ghost2->move();

if((modeCyan == 1 || modeCyan == 2) && !slowCyan)

ghost3->move();

if((modeOrange == 1 || modeOrange == 2) && !slowOrange)

ghost4->move();

}

}

void MainWindow::getPointTime(){

delete timer8;

point->setVisible(false);

pause = false;

}

void MainWindow::pauseTime(){

delete timer8;

ui->ready->setVisible(false);

pause = false;

}

void MainWindow::slowMove(){

if(!pause){

if(slowRed)

ghost1->move();

if(slowPink)

ghost2->move();

if(slowCyan)

ghost3->move();

if(slowOrange)

ghost4->move();

}

if(ghost1->x() == 376.0 && ghost1->y() == 272.0){

slowRed = false;

}

if(ghost2->x() == 376.0 && ghost2->y() == 272.0){

slowPink = false;

}

if(ghost3->x() == 376.0 && ghost3->y() == 272.0){

slowCyan = false;

}

if(ghost4->x() == 376.0 && ghost4->y() == 272.0){

slowOrange = false;

}

}

void MainWindow::cyanStartSlot(){

delete timer10;

ghost3->setCyanStart();

}

void MainWindow::orangeStartSlot(){

delete timer11;

ghost4->setOrangeStart();

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

// About();

}

void MainWindow::on\_pushButton\_2\_clicked()

{

close();

//About();

}

void MainWindow::on\_CheckUpdateButton\_clicked()

{

QByteArray arrBlock;

QDataStream out(&arrBlock, QIODevice::WriteOnly);

out << quint16(0) << "version:" + this->mVersion;

out.device()->seek(0);

out << quint16(arrBlock.size() - sizeof(quint16));

mClient->tcpSocket->write(arrBlock);

}

void MainWindow::on\_ConnectButton\_clicked()

{

mClient->connect2host();

}

void MainWindow::on\_DisconnetButton\_clicked()

{

mClient->closeConnection();

}

void MainWindow::on\_Version\_clicked()

{

QString version = QString("Version: %1").arg(mVersion);

QMessageBox::information(this, tr("Info"), version);

}

void MainWindow::receivedSomething(QString msg)

{

if(msg.contains("version:"))

{

QString version = msg.remove(0, 8);

QMessageBox msgBox;

if(version == this->mVersion)

{

msgBox.setText("No available updates");

msgBox.exec();

}

else

{

msgBox.setText("New version " + version + " is available");

msgBox.setInformativeText("Do you want to update app?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setDefaultButton(QMessageBox::Yes);

int res = msgBox.exec();

if (res == QMessageBox::Yes)

{

this->setVersion(version);

QByteArray arrBlock;

QDataStream out(&arrBlock, QIODevice::WriteOnly);

QString answer = "OK ON UPDATE";

out << quint16(0) << answer;

download();

out.device()->seek(0);

out << quint16(arrBlock.size() - sizeof(quint16));

this->mClient->tcpSocket->write(arrBlock);

}

}

}

if(msg.contains("file:"))

{

download();

for(int i = 0; i < 100; ++i)

{

this->mTimerBar->start(500);

}

}

}

QString MainWindow::getDefaultVersion()

{

QString val;

QFile file;

file.setFileName("client.json");

file.open(QIODevice::ReadOnly | QIODevice::Text);

val = file.readAll();

file.close();

QString version = "1";

if (val != "") {

QJsonDocument doc = QJsonDocument::fromJson(val.toUtf8());

QJsonObject json = doc.object();

version = json["version"].toString();

}

return version;

}

void MainWindow::setVersion(QString version)

{

QJsonObject recordObject;

recordObject.insert("version", QJsonValue::fromVariant(version));

QJsonDocument doc(recordObject);

QString jsonString = doc.toJson(QJsonDocument::Indented);

QFile file;

file.setFileName("client.json");

file.open(QIODevice::WriteOnly | QIODevice::Text);

QTextStream stream( &file );

stream << jsonString;

file.close();

mVersion = version;

}

QString MainWindow::getVersion()

{

return mVersion;

}

void MainWindow::download()

{

QFile::rename("D:\\101\\client\\pd2-pacman\\mainwindow.cpp", "D:\\101\\client\\pd2-pacman\\mainwindow.cpp.bak");

QFile::rename("D:\\101\\client\\pd2-pacman\\pacman.cpp", "D:\\101\\client\\pd2-pacman\\pacman.cpp.bak");

QFile::rename("D:\\101\\client\\pd2-pacman\\resources.qrc", "D:\\101\\client\\pd2-pacman\\resources.qrc.bak");

QFile::copy("D:\\101\\server\\update\\mainwindow.cpp", "D:\\101\\client\\pd2-pacman\\mainwindow.cpp");

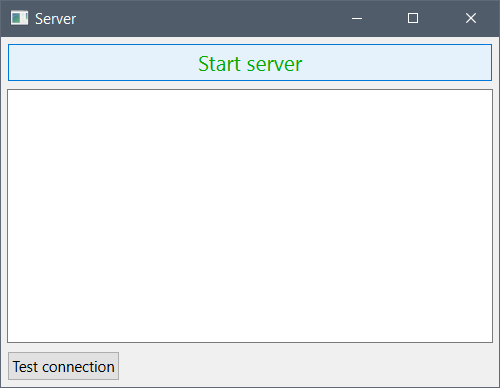
QFile::copy("D:\\101\\server\\update\\pacman.cpp", "D:\\101\\client\\pd2-pacman\\pacman.cpp");

QFile::copy("D:\\101\\server\\update\\resources.qrc", "D:\\101\\client\\pd2-pacman\\resources.qrc");

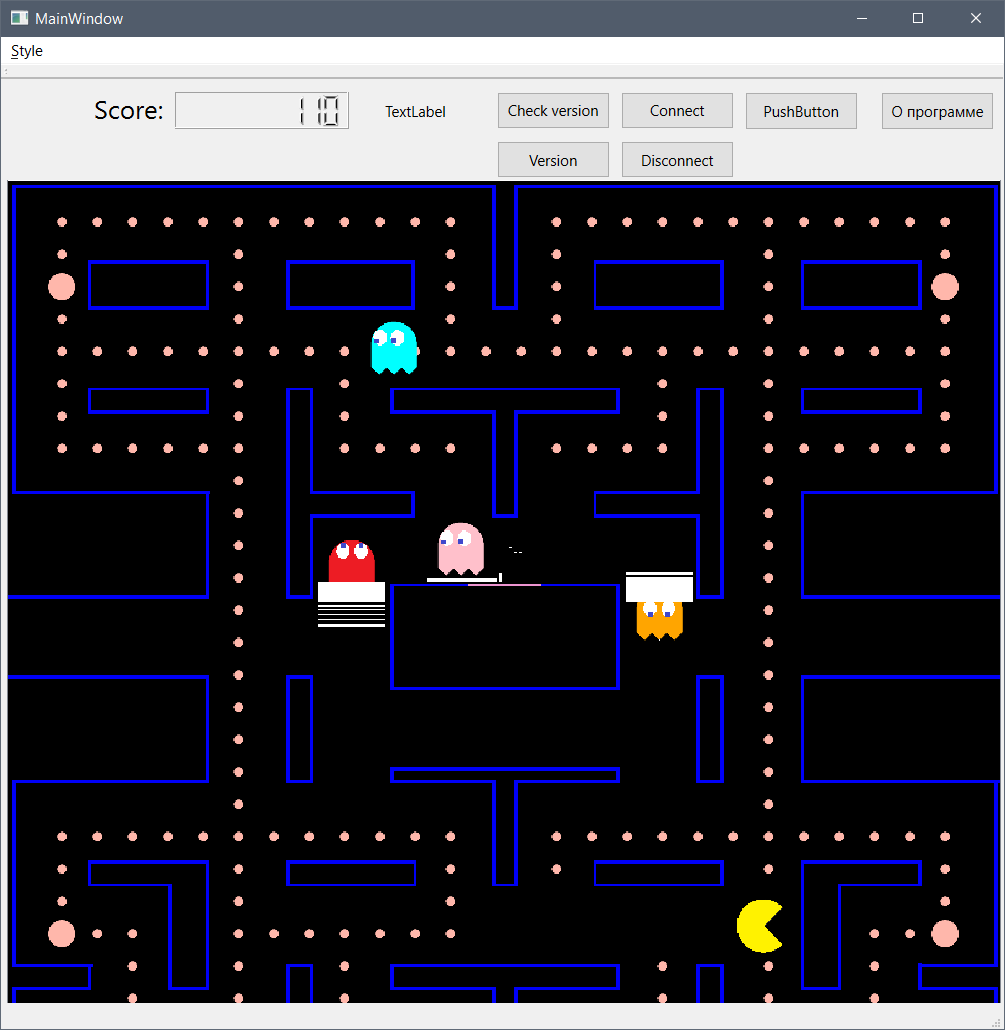
}

**Результаты тестирования программы:**

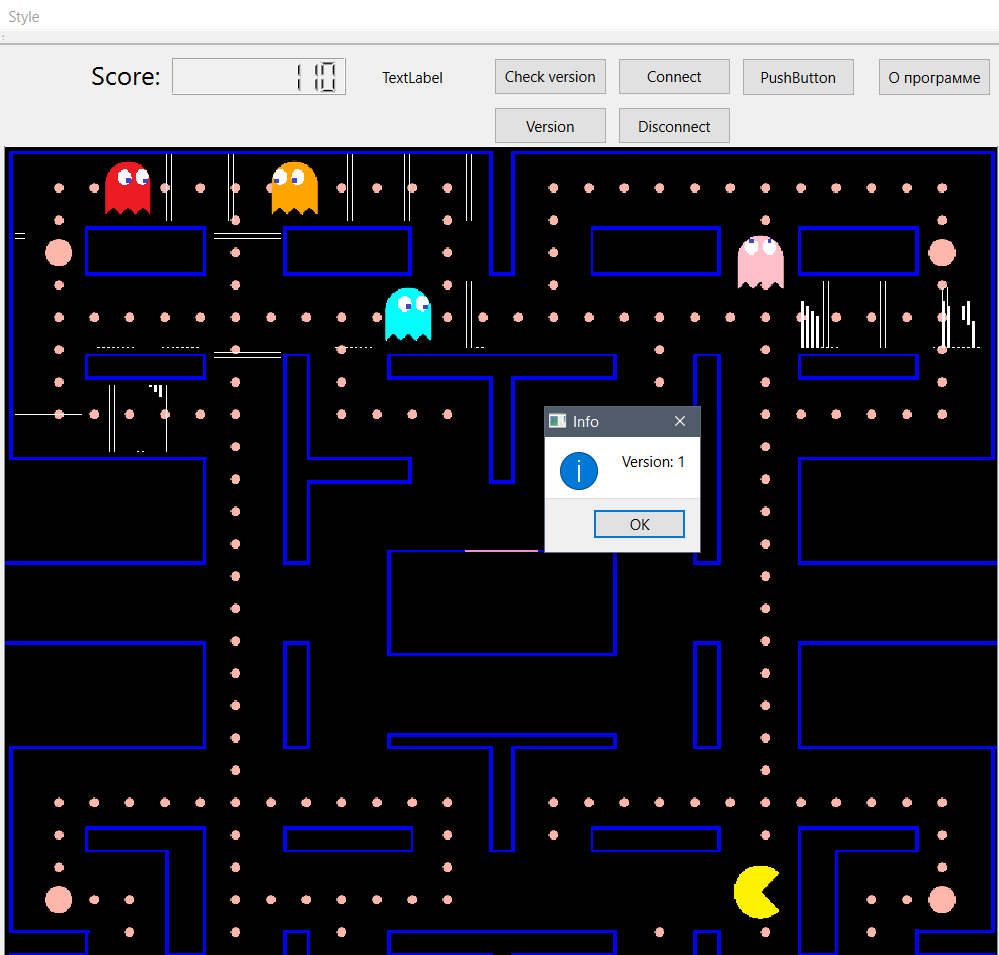
**Сервер:**



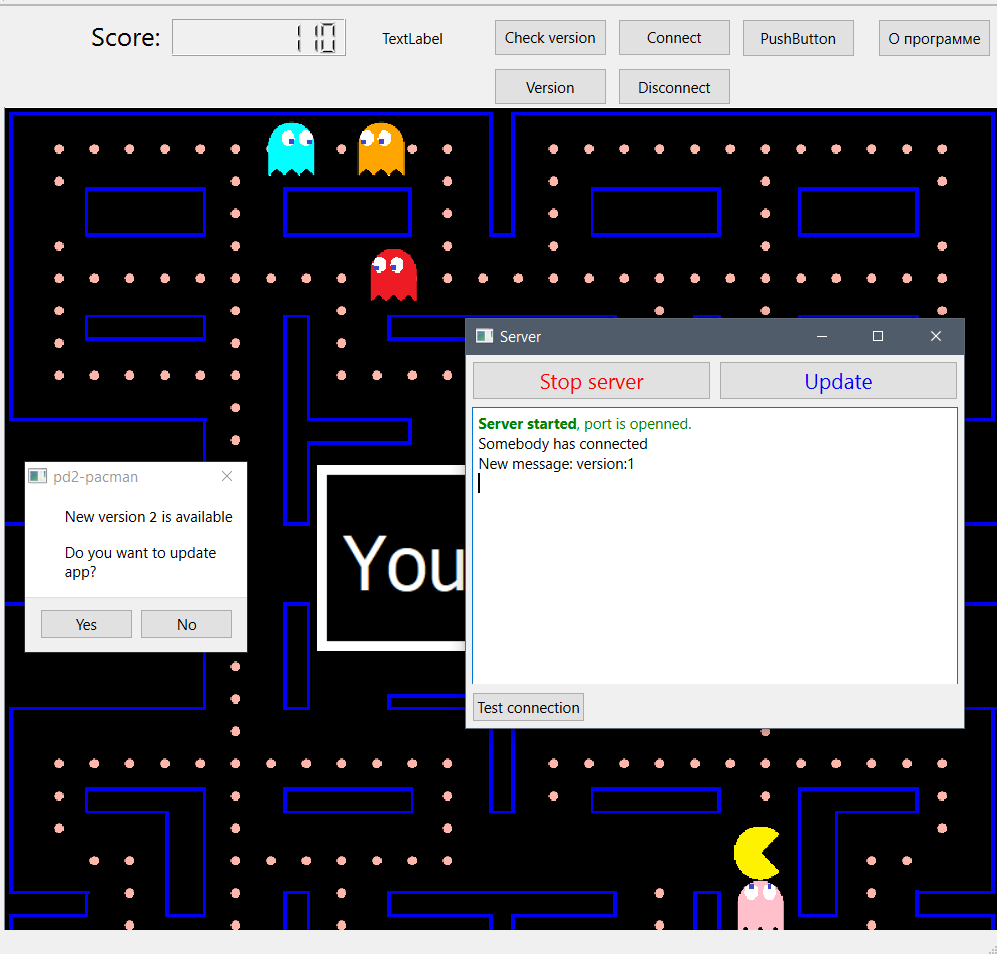
**Приложение:**



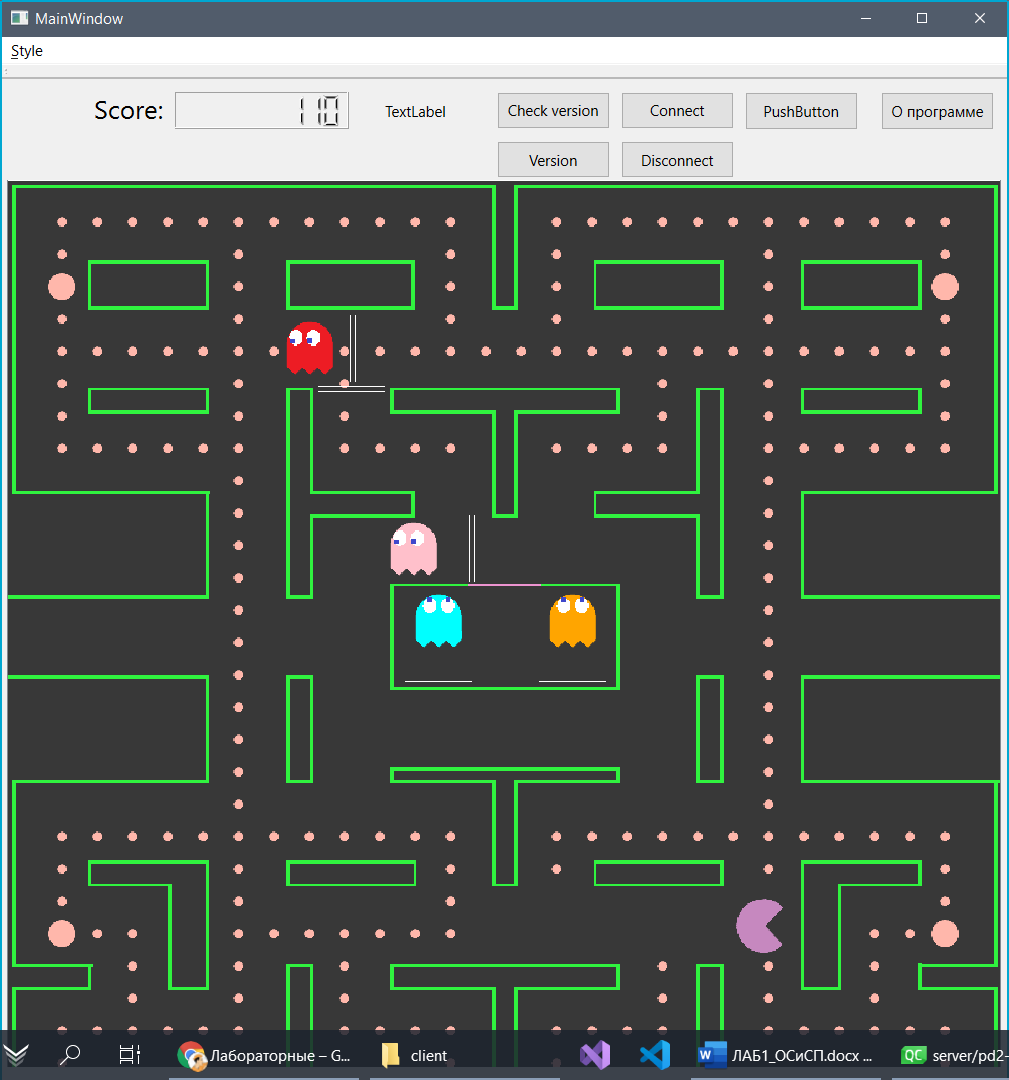
**Проверяем текущую версию, нажав Version:**

****

**Нажимаем Connect и подключаемся к серверу, после этого через 20 секунд будет предложено обновиться до новой версии:**

****

**Нажимаем yes, закрываем приложение и сервер, уадляем папку с билдом и перезапускаем приложение.**



**Вывод:** ознакомилась с возможностями, предлагаемыми Qt для поддержки сетевого взаимодействия программ.