МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

\

ЛАБОРАТОРНАЯ РАБОТА №1-2

По дисциплине: «ОСИСП»

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Брич М.Н.

Проверила:

Дряпко А.В.

Брест 2021

**Цель работы:** приобрести практические навыки проектирования и разработки приложений с графическим

пользовательским интерфейсом в ОС Windows средствами Qt

Вариант 3

*Задание:*

Игра «Змейка».  
***snake.h:***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPaintEvent>

#include <QKeyEvent>

#include <QDialog>

#include <QComboBox>

#include "interface.h"

const int BLOCK\_SIZE=15;

const int MARGIN=22;

const int AREA\_ROW=30;

const int AREA\_COL=30;

const int TIME\_INTERVAL=100;

enum Direction

{

UP,

DOWN,

LEFT,

RIGHT

};

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

virtual void paintEvent(QPaintEvent \*event);

virtual void keyPressEvent(QKeyEvent \*event);

QDialog dlg;

QComboBox\* modeMenu = new QComboBox;

public:

void InitMenu();

void SetPlugins();

void InitGame();

void InitPause();

void PauseResumeGame();

void GameOver();

void GenerateFood();

void GenerateSnake();

bool IsGameOver();

private slots:

void SnakeUpdate();

void aboutQt();

void applyPlugin(int index);

void BtnDlgClick1();

void BtnDlgClick2();

void BtnDlgClick3();

void LoadingLevel();

void CheckUpdate();

private:

Ui::MainWindow\*ui;

QTimer \*gameTimer;

bool isPause;

QPoint foodPoint;

QList<QPoint> wall;

QList<QPoint> snake;

Direction dir;

int score;

int level;

QVector< Interface\* > mPlugins;

};

#endif // MAINWINDOW\_H

***menu.cpp:***

#include <QPainter>

#include <time.h>

#include <QTimer>

#include <QString>

#include <QtWidgets>

#include <QMessageBox>

#include "snake.h"

#include "ui\_snake.h"

typedef void (\*about)();

typedef bool (\*GenerateWall)(int level, int i, int j);

typedef int (\*countLevel)();

void MainWindow::SetPlugins()

{

QDir \*dir;

#ifdef QT\_DEBUG

dir = new QDir("/home/eve9te/bstu/osisp2.0/Snake/plugins/debug");

#else

#ifdef QT\_RELEASE

dir = new QDir("/home/eve9te/bstu/osisp2.0/Snake/plugins/release");

#endif

#endif

if(dir->entryList(QDir::Files).isEmpty())

qDebug() << "dir isEmpty";

foreach(QString str, dir->entryList(QDir::Files))

{

QPluginLoader loader(dir->absoluteFilePath(str));

QObject \*object = qobject\_cast<QObject\*>(loader.instance());//извлекаем плагин

Interface \*plugin = qobject\_cast<Interface\*>(object);//приводим к интерфейсу игры

if(plugin)

{

this->mPlugins.push\_back(plugin);

// const char \* namePlugin = plugin->pluginName().toLocal8Bit().data();

// QAction \*applyPlugin = new QAction(tr(namePlugin), this);

modeMenu->addItem(plugin->pluginName());

connect(modeMenu, SIGNAL(activated(int)), this, SLOT(applyPlugin(int)));

}

else

{

qDebug() << "error";

}

}

delete dir;

}

void MainWindow::applyPlugin(int index)

{

QFont palette = this->mPlugins[index]->changeView();

QApplication::setFont(palette);

}

void MainWindow::InitMenu()

{

dlg.setWindowTitle(tr("Выбирите уровень"));

dlg.setFixedSize(QSize(MARGIN\*7+(AREA\_COL+1)\*BLOCK\_SIZE,MARGIN\*2+AREA\_ROW\*BLOCK\_SIZE));

dlg.setGeometry(

QStyle::alignedRect(

Qt::LeftToRight,

Qt::AlignCenter,

dlg.size(),

QApplication::desktop()->availableGeometry(this)

)

);

QBoxLayout\* layout = new QVBoxLayout;

layout->setAlignment(Qt::AlignVCenter);

QMenuBar\* menuBar = new QMenuBar();

QMenu\* menu = new QMenu("О программе");

menuBar->addMenu(menu);

menu->addAction("about.dll", this,SLOT(aboutQt()),

Qt::CTRL + Qt::Key\_Q);

layout->setMenuBar(menuBar);

QPushButton\* btn\_easy = new QPushButton("Easy");

QPushButton\* btn\_medium = new QPushButton("Medium");

QPushButton\* btn\_hard = new QPushButton("Hard");

QPushButton\* btn\_update = new QPushButton("Update");

btn\_update->setContentsMargins(QMargins(0,30,0,0));

QPushButton\* btn\_exit = new QPushButton("Exit");

// btn\_exit->setStyleSheet("margin-top: 5px;"

// "padding-top: 3px; padding-bottom: 3px;");

connect(btn\_easy, SIGNAL(clicked()), this, SLOT(BtnDlgClick1()));

connect(btn\_medium, SIGNAL(clicked()), this, SLOT(BtnDlgClick2()));

connect(btn\_hard, SIGNAL(clicked()), this, SLOT(BtnDlgClick3()));

connect(btn\_update, SIGNAL(clicked()), this, SLOT(CheckUpdate()));

connect(btn\_exit, SIGNAL(clicked()), &dlg, SLOT(close()));

layout->addWidget(btn\_easy);

layout->addWidget(btn\_medium);

layout->addWidget(btn\_hard);

layout->addWidget(btn\_update);

layout->addWidget(btn\_exit);

layout->addWidget(modeMenu);

dlg.setLayout(layout);

if(dlg.exec() == QDialog::Accepted) {

InitGame();

}

else

{

exit(0);

}

}

void MainWindow::BtnDlgClick1()

{

level = 1;

LoadingLevel();

}

void MainWindow::BtnDlgClick2()

{

level = 2;

LoadingLevel();

}

void MainWindow::BtnDlgClick3()

{

level = 3;

LoadingLevel();

}

void MainWindow::LoadingLevel()

{

QLibrary \*helperLib = new QLibrary("helper");

if (!helperLib->load())

qDebug() << helperLib->errorString();

if (helperLib->load())

qDebug() << "library loaded";

countLevel countLevel\_ = (countLevel) helperLib->resolve("countLevel");

if(level > countLevel\_())

{

QMessageBox msgBox;

msgBox.setWindowTitle("Error");

msgBox.setText("Уровень еще не добавлен");

msgBox.exec();

return;

}

GenerateWall generateWall\_ = (GenerateWall)helperLib->resolve("GenerateWall");

for (int i = 0; i < AREA\_COL; ++i) {

for (int j = 0; j < AREA\_ROW; ++j) {

if(generateWall\_(level, i, j)){

wall.push\_back(QPoint(i, j));

}

}

}

delete helperLib;

dlg.accept();

}

void MainWindow::CheckUpdate()

{

QMessageBox msgBox;

msgBox.setWindowTitle("Update");

msgBox.setText("This is latest version");

msgBox.exec();

}

void MainWindow::aboutQt()

{

QLibrary \*aboutLib = new QLibrary("about");

if (!aboutLib->load())

qDebug() << aboutLib->errorString();

if (aboutLib->load())

qDebug() << "library loaded";

about showWindowAboutProgramm = (about)aboutLib->resolve("about");

showWindowAboutProgramm();

delete aboutLib;

}

MainWindow::~MainWindow()

{

delete ui;

}

***snake.cpp***

#include <QPainter>

#include <time.h>

#include <QTimer>

#include <QString>

#include <QtWidgets>

#include <QMessageBox>

#include "snake.h"

#include "ui\_snake.h"

MainWindow::MainWindow(QWidget \*parent): QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

setFixedSize(QSize(MARGIN\*7+(AREA\_COL+1)\*BLOCK\_SIZE,MARGIN\*2+AREA\_ROW\*BLOCK\_SIZE));

SetPlugins();

InitMenu();

}

void MainWindow::GameOver()

{

gameTimer->stop();

QMessageBox::information(this,"Failed","Game over!");

InitMenu();

}

void MainWindow::PauseResumeGame()

{

if(!isPause)

{

isPause=!isPause;

gameTimer->stop();

InitPause();

}

else

{

isPause=!isPause;

gameTimer->start(TIME\_INTERVAL);

}

}

void MainWindow::InitPause()

{

QMessageBox::StandardButton pause;

pause = QMessageBox::question(this, "Pause", "Продолжить?",

QMessageBox::Yes|QMessageBox::No);

if (pause == QMessageBox::Yes) {

PauseResumeGame();

} else {

InitMenu();

}

}

***game.cpp***

#include <QPainter>

#include <time.h>

#include <QTimer>

#include <QString>

#include <QtWidgets>

#include <QMessageBox>

#include "snake.h"

#include "ui\_snake.h"

typedef bool (\*Generate\_Snake)(int i);

void MainWindow::InitGame()

{

snake.clear();

GenerateSnake();

dir=RIGHT;

srand(time(0));

GenerateFood();

score=0;

isPause=false;

gameTimer=new QTimer(this);

connect(gameTimer, SIGNAL(timeout()), this, SLOT(SnakeUpdate()));

gameTimer->start(TIME\_INTERVAL);

}

void MainWindow::GenerateSnake()

{

QLibrary \*helperLib = new QLibrary("helper");

if (!helperLib->load())

qDebug() << helperLib->errorString();

if (helperLib->load())

qDebug() << "library loaded";

Generate\_Snake generateSnake\_ = (Generate\_Snake)helperLib->resolve("GenerateSnake");

for (int i=AREA\_COL;i>=0;i--) {

if(generateSnake\_(i)){

snake.push\_back(QPoint(i, 0));

}

}

delete helperLib;

}

void MainWindow::paintEvent(QPaintEvent \*event)

{

Q\_UNUSED(event);

QPainter painter(this);

painter.setBrush(Qt::yellow);

painter.setPen(Qt::blue);

painter.drawRect(MARGIN,MARGIN,AREA\_COL\*BLOCK\_SIZE,AREA\_ROW\*BLOCK\_SIZE);

painter.setBrush(Qt::blue);

painter.setPen(Qt::black);

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

painter.drawRect(MARGIN+wall[i].x()\*BLOCK\_SIZE,MARGIN+wall[i].y()\*BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE);

painter.setBrush(Qt::red);

painter.setPen(Qt::green);

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

painter.drawRect(MARGIN+snake[i].x()\*BLOCK\_SIZE,MARGIN+snake[i].y()\*BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE);

painter.setBrush(Qt::green);

painter.drawEllipse(MARGIN+foodPoint.x()\*BLOCK\_SIZE,MARGIN+foodPoint.y()\*BLOCK\_SIZE,BLOCK\_SIZE,BLOCK\_SIZE);

painter.setPen(Qt::black);

painter.setFont(QFont("Arial",14));

painter.drawText(MARGIN\*3+AREA\_COL\*BLOCK\_SIZE,MARGIN+2\*BLOCK\_SIZE,"score: "+QString::number(score));

}

void MainWindow::keyPressEvent(QKeyEvent \*event)

{

switch(event->key())

{

case Qt::Key\_Up:

if(dir!=DOWN)

dir=UP;

break;

case Qt::Key\_Down:

if(dir!=UP)

dir=DOWN;

break;

case Qt::Key\_Left:

if(dir!=RIGHT)

dir=LEFT;

break;

case Qt::Key\_Right:

if(dir!=LEFT)

dir=RIGHT;

break;

case Qt::Key\_P:

PauseResumeGame();

break;

case Qt::Key\_Escape:

PauseResumeGame();

break;

default:

break;

}

}

bool MainWindow::IsGameOver()

{

int x=snake.front().x();

int y=snake.front().y();

if(x<0||x>AREA\_COL-1||y<0||y>AREA\_ROW-1)

return true;

for(int i=1;i<snake.size();i++)

if(snake[i]==snake.front())

return true;

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

if(wall[i]==snake.front())

return true;

return false;

}

void MainWindow::SnakeUpdate()

{

switch(dir)

{

case UP:

snake.push\_front(QPoint(snake.front().x(),snake.front().y()-1));

break;

case DOWN:

snake.push\_front(QPoint(snake.front().x(),snake.front().y()+1));

break;

case LEFT:

snake.push\_front(QPoint(snake.front().x()-1,snake.front().y()));

break;

case RIGHT:

snake.push\_front(QPoint(snake.front().x()+1,snake.front().y()));

break;

default:

break;

}

if(snake.contains(foodPoint))

{

score+=1;

GenerateFood();

gameTimer->start(TIME\_INTERVAL - score);

}

else

snake.pop\_back();

if(IsGameOver())

{

GameOver();

return;

}

update();

}

***food.cpp***

***#include <QPainter>***

***#include <time.h>***

***#include <QTimer>***

***#include <QString>***

***#include <QtWidgets>***

***#include <QMessageBox>***

***#include "snake.h"***

***#include "ui\_snake.h"***

***void MainWindow::GenerateFood()***

***{***

***foodPoint.setX(rand()%AREA\_COL);***

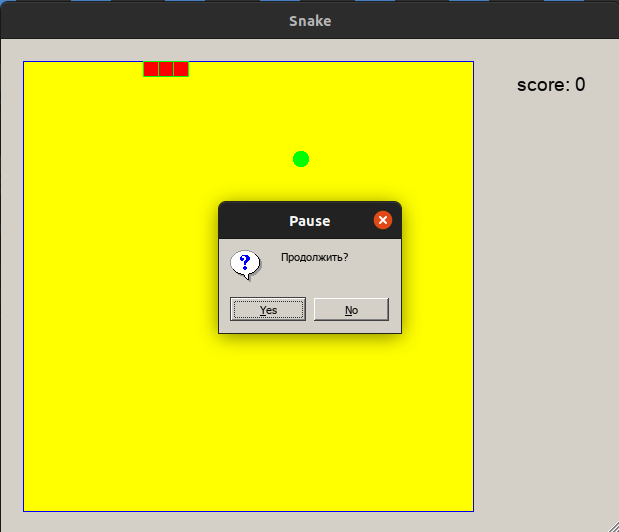
***foodPoint.setY(rand()%AREA\_ROW);***

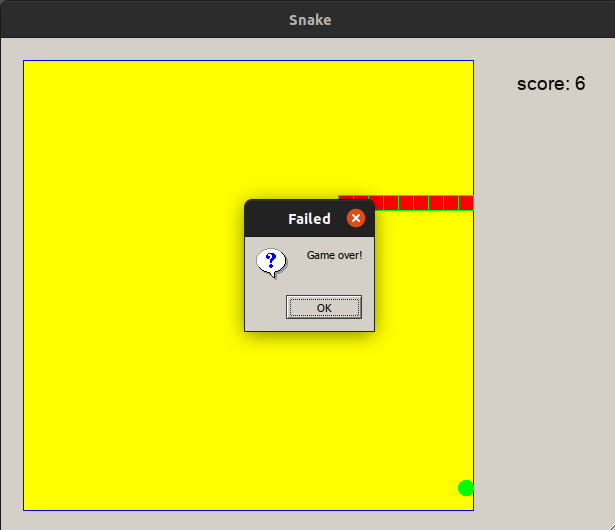
***if(snake.contains(foodPoint) || wall.contains(foodPoint))***

***GenerateFood();***

***}***

*Тестирование:*





Вывод: приобрел практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.