Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №1-2

за 5 семестр

По дисциплине: «ОСиСП»

Выполнила:

студентка 3 курса

группы ПО-4(1)

Воробей А.П.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №1-2

Цель работы: приобрести практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.

Вариант 4

Задание:

4)Игра «Пакман». Реализовать игру с одним уровнем и 3 врагами-привидениями. При получении бонуса герой (Пакман) «съедает» привидение. Бонус действует ограниченное время (15 секунд). Приведения движутся рандомно. При столкновении с героем, если бонус не действует, игра заканчивается.

**Текст программы:**

1. **bigpoint.cpp**

#include "bigpoint.h"

#include "mainwindow.h"

bigpoint::bigpoint(){

pics.push\_back(QPixmap(":/img/img/bigpoint"));

pics.push\_back(QPixmap(":/img/img/transparent"));

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

picsArg = 0;

isEaten = false;

}

void bigpoint::collide(){

isEaten = true;

setPixmap(pics[1].scaled(50, 50, Qt::KeepAspectRatio));

}

void bigpoint::changePics(){

if(!isEaten){

picsArg = picsArg \* -1 + 1;

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

}

}

1. **bigpoint.h**

#ifndef BIGPOINT\_H

#define BIGPOINT\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

class bigpoint : public QGraphicsPixmapItem

{

public:

bigpoint();

void collide();

void changePics();

private:

bool isEaten;

QList<QPixmap> pics;

int picsArg;

};

#endif // BIGPOINT\_H

1. **cyan.h**

#ifndef CYAN\_H

#define CYAN\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <ghost.h>

class cyan : public ghost

{

public:

cyan();

virtual void move();

};

#endif // CYAN\_H

1. **cyan.cpp**

#include "cyan.h"

#include "mainwindow.h"

cyan::cyan(){

pics[0].push\_back(QPixmap(":/img/img/cyanu1"));

pics[0].push\_back(QPixmap(":/img/img/cyanu2"));

pics[1].push\_back(QPixmap(":/img/img/cyand1"));

pics[1].push\_back(QPixmap(":/img/img/cyand2"));

pics[2].push\_back(QPixmap(":/img/img/cyanl1"));

pics[2].push\_back(QPixmap(":/img/img/cyanl2"));

pics[3].push\_back(QPixmap(":/img/img/cyanr1"));

pics[3].push\_back(QPixmap(":/img/img/cyanr2"));

pics[4].push\_back(QPixmap(":/img/img/blue1"));

pics[4].push\_back(QPixmap(":/img/img/blue2"));

pics[5].push\_back(QPixmap(":/img/img/white1"));

pics[5].push\_back(QPixmap(":/img/img/white2"));

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

setPos(322, 342);

direction = 0;

picsArg = 0;

}

void cyan::move(){

direction = changeDirection();

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(x() > 792){

setPos(-40, 342);

}

if(x() < -40){

setPos(792, 342);

}

}

1. **ghost.h**

#ifndef GHOST\_H

#define GHOST\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <cstdlib>

class ghost : public QGraphicsPixmapItem

{

public:

ghost();

virtual void move();

int changeDirection();

void changePics();

void changePics2();

void changePics3();

void setDirection(int d);

void setCyanStart();

void setOrangeStart();

protected:

bool mapDirection[10][10][4] = {{{0,1,0,1}, {1,1,0,1}, {1,0,0,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,1,0,1}, {1,0,0,1}, {0,1,0,1}, {1,0,0,1}},

{{0,0,1,1}, {0,0,1,1}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,0,1,1}},

{{0,1,1,1}, {1,1,1,1}, {1,1,1,0}, {1,1,0,0}, {1,1,1,1}, {1,1,0,0}, {1,1,1,1}, {1,1,0,1}, {1,0,1,0}, {0,0,1,1}},

{{0,0,1,1}, {0,1,1,1}, {1,0,0,1}, {0,1,0,1}, {1,1,1,0}, {1,1,0,1}, {1,0,1,1}, {0,1,1,1}, {1,0,0,1}, {0,0,1,1}},

{{0,1,1,0}, {1,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,1,1,0}, {1,0,1,1}},

{{0,1,0,1}, {1,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,1,0,1}, {1,0,1,1}},

{{0,0,1,1}, {0,1,1,1}, {1,0,1,0}, {0,1,1,0}, {1,1,0,1}, {1,1,1,0}, {1,0,1,1}, {0,1,1,1}, {1,0,1,0}, {0,0,1,1}},

{{0,1,1,1}, {1,1,1,1}, {1,1,0,1}, {1,1,0,0}, {1,1,1,1}, {1,1,0,0}, {1,1,1,1}, {1,1,1,0}, {1,0,0,1}, {0,0,1,1}},

{{0,0,1,1}, {0,0,1,1}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,0,1,1}},

{{0,1,1,0}, {1,1,1,0}, {1,0,1,0}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,1,1,0}, {1,0,1,0}, {0,1,1,0}, {1,0,1,0}}};

bool intersection[10][10] = {{1, 1, 1, 0, 0, 0, 1, 1, 1, 1},

{0, 0, 0, 0, 0, 0, 0, 1, 1, 0},

{1, 1, 1, 0, 1, 0, 1, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{1, 1, 1, 1, 0, 0, 1, 1, 1, 1},

{1, 1, 1, 1, 0, 0, 1, 1, 1, 1},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{1, 1, 1, 0, 1, 0, 1, 1, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 1, 1, 0},

{1, 1, 1, 0, 0, 0, 1, 1, 1, 1}};

QList<QPixmap> pics[6];

int direction; // up := 0, down := 1, left := 2, right := 3, stop := 4

int picsArg;

bool cyanStart;

bool orangeStart;

};

#endif // GHOST\_H

1. **ghost.cpp**

#include "ghost.h"

#include "mainwindow.h"

ghost::ghost(){

cyanStart = false;

orangeStart = false;

}

void ghost::move(){

}

int ghost::changeDirection(){

int i = 0, j = 0, tempDirect;

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(x() == 376.0 && y() == 272.0){

tempDirect = rand() % 2 + 2;

return tempDirect;

}

if(x() == 322.0 && y() == 330.0){

return 1;

}

if(x() == 322.0 && y() == 354.0){

return 0;

}

if(x() == 430.0 && y() == 330.0){

return 1;

}

if(x() == 430.0 && y() == 354.0){

return 0;

}

if(x() == 322.0 && y() == 342.0 && cyanStart){

return 3;

}

if(x() == 430.0 && y() == 342.0 && orangeStart){

return 2;

}

if(x() == 376.0 && y() == 342.0){

return 0;

}

if(checkX && checkY){

tempDirect = rand() % 4;

if(mapDirection[i][j][tempDirect] && intersection[i][j]){

return tempDirect;

}

else if(!mapDirection[i][j][tempDirect] && intersection[i][j]){

return 4;

}

else {

return direction;

}

}

else {

return direction;

}

}

void ghost::changePics(){

picsArg = picsArg \* -1 + 1;

if(direction < 4 )

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

}

void ghost::changePics2(){

picsArg = picsArg \* -1 + 1;

setPixmap(pics[4][picsArg].scaled(50, 50, Qt::KeepAspectRatio));

}

void ghost::changePics3(){

picsArg = picsArg \* -1 + 1;

setPixmap(pics[5][picsArg].scaled(50, 50, Qt::KeepAspectRatio));

}

void ghost::setDirection(int d){

direction = d;

}

void ghost::setCyanStart(){

cyanStart = true;

}

void ghost::setOrangeStart(){

orangeStart = true;

}

1. **main.h**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

1. **mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

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"));

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]);

}

}

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;

}

1. **mainwindow.g**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QGraphicsScene>

#include <QLabel>

#include <QTimer>

#include <QString>

#include <QLabel>

#include <cstdlib>

#include <ctime>

#include <QQueue>

#include "pacman.h"

#include "smallpoint.h"

#include "bigpoint.h"

#include "ghost.h"

#include "red.h"

#include "pink.h"

#include "cyan.h"

#include "orange.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

public slots:

void allMove();

void allChangePics();

void bigpointChangePics();

void bonusTime();

void bonusTime2();

void flicker();

void ghostmove2();

void getPointTime();

void pauseTime();

void slowMove();

void cyanStartSlot();

void orangeStartSlot();

private:

Ui::MainWindow \*ui;

QGraphicsScene \*scene;

smallpoint \*Smallpoint[29][26];

bigpoint \*Bigpoint[4];

pacman \*Pacman;

ghost \*ghost1;

ghost \*ghost2;

ghost \*ghost3;

ghost \*ghost4;

QTimer \*timer1; //for move

QTimer \*timer2; //for changePics

QTimer \*timer3; //for bigpoint

QTimer \*timer4[4]; //for bonus time 1

QTimer \*timer5[4]; //for bonus time 2

QTimer \*timer6[4]; //for flicker

QTimer \*timer7[4]; //for move in bonus time

QTimer \*timer8; //for point, pause time

QTimer \*timer9; //for slow move

QTimer \*timer10; //for cyan start

QTimer \*timer11; //for orange start

int smallpointNum;

int bigpointNum;

// mode: 0 := not bonus time, 1 := bonus time, 2 := bonus time2

int modeRed;

int modePink;

int modeCyan;

int modeOrange;

QLabel \*point;

QQueue<int> q;

int addpoint;

bool pause;

bool isBonus2;

bool slowRed;

bool slowPink;

bool slowCyan;

bool slowOrange;

};

#endif // MAINWINDOW\_H

**10.orange.cpp**

#include "orange.h"

#include "mainwindow.h"

orange::orange(){

pics[0].push\_back(QPixmap(":/img/img/orangeu1"));

pics[0].push\_back(QPixmap(":/img/img/orangeu2"));

pics[1].push\_back(QPixmap(":/img/img/oranged1"));

pics[1].push\_back(QPixmap(":/img/img/oranged2"));

pics[2].push\_back(QPixmap(":/img/img/orangel1"));

pics[2].push\_back(QPixmap(":/img/img/orangel2"));

pics[3].push\_back(QPixmap(":/img/img/oranger1"));

pics[3].push\_back(QPixmap(":/img/img/oranger2"));

pics[4].push\_back(QPixmap(":/img/img/blue1"));

pics[4].push\_back(QPixmap(":/img/img/blue2"));

pics[5].push\_back(QPixmap(":/img/img/white1"));

pics[5].push\_back(QPixmap(":/img/img/white2"));

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

setPos(430, 342);

direction = 0;

picsArg = 0;

}

void orange::move(){

direction = changeDirection();

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(x() > 792){

setPos(-40, 342);

}

if(x() < -40){

setPos(792, 342);

}

}

**11. orange.h**

#ifndef ORANGE\_H

#define ORANGE\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <ghost.h>

class orange : public ghost

{

public:

orange();

virtual void move();

};

#endif // ORANGE\_H

**12. pacman.cpp**

#include "pacman.h"

#include "mainwindow.h"

pacman::pacman(){

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

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

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

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

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

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

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

pics[3].push\_back(QPixmap(":/img/img/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;

}

}

**13. pacman.h**

#ifndef PACMAN\_H

#define PACMAN\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QKeyEvent>

#include <QTimer>

class pacman : public QGraphicsPixmapItem

{

public:

pacman();

void keyPressEvent(QKeyEvent \*event);

void move();

void changePics();

bool canChange();

bool isWall();

private:

bool mapDirection[10][10][4] = {{{0,1,0,1}, {1,1,0,1}, {1,0,0,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,1,0,1}, {1,0,0,1}, {0,1,0,1}, {1,0,0,1}},

{{0,0,1,1}, {0,0,1,1}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,0,1,1}},

{{0,1,1,1}, {1,1,1,1}, {1,1,1,0}, {1,1,0,0}, {1,1,1,1}, {1,1,0,0}, {1,1,1,1}, {1,1,0,1}, {1,0,1,0}, {0,0,1,1}},

{{0,0,1,1}, {0,1,1,1}, {1,0,0,1}, {0,1,0,1}, {1,1,1,0}, {1,1,0,1}, {1,0,1,1}, {0,1,1,1}, {1,0,0,1}, {0,0,1,1}},

{{0,1,1,0}, {1,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,1,0}, {1,0,1,1}, {0,1,1,0}, {1,0,1,1}},

{{0,1,0,1}, {1,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,1,0,1}, {1,0,1,1}},

{{0,0,1,1}, {0,1,1,1}, {1,0,1,0}, {0,1,1,0}, {1,1,0,1}, {1,1,1,0}, {1,0,1,1}, {0,1,1,1}, {1,0,1,0}, {0,0,1,1}},

{{0,1,1,1}, {1,1,1,1}, {1,1,0,1}, {1,1,0,0}, {1,1,1,1}, {1,1,0,0}, {1,1,1,1}, {1,1,1,0}, {1,0,0,1}, {0,0,1,1}},

{{0,0,1,1}, {0,0,1,1}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,0,1,1}, {0,1,0,1}, {1,0,1,1}, {0,0,1,1}},

{{0,1,1,0}, {1,1,1,0}, {1,0,1,0}, {0,0,0,0}, {0,0,1,1}, {0,0,0,0}, {0,1,1,0}, {1,0,1,0}, {0,1,1,0}, {1,0,1,0}}};

QList<QPixmap> pics[4];

int direction; // up := 0, down := 1, left := 2, right := 3, stop := 4

int keyDirection;

int picsArg;

};

#endif // PACMAN\_H

**14. pink.cpp**

#include "pink.h"

#include "mainwindow.h"

pink::pink(){

pics[0].push\_back(QPixmap(":/img/img/pinku1"));

pics[0].push\_back(QPixmap(":/img/img/pinku2"));

pics[1].push\_back(QPixmap(":/img/img/pinkd1"));

pics[1].push\_back(QPixmap(":/img/img/pinkd2"));

pics[2].push\_back(QPixmap(":/img/img/pinkl1"));

pics[2].push\_back(QPixmap(":/img/img/pinkl2"));

pics[3].push\_back(QPixmap(":/img/img/pinkr1"));

pics[3].push\_back(QPixmap(":/img/img/pinkr2"));

pics[4].push\_back(QPixmap(":/img/img/blue1"));

pics[4].push\_back(QPixmap(":/img/img/blue2"));

pics[5].push\_back(QPixmap(":/img/img/white1"));

pics[5].push\_back(QPixmap(":/img/img/white2"));

setPixmap(pics[1][0].scaled(50, 50, Qt::KeepAspectRatio));

setPos(376, 342);

direction = 0;

picsArg = 0;

}

void pink::move(){

direction = changeDirection();

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(x() > 792){

setPos(-40, 342);

}

if(x() < -40){

setPos(792, 342);

}

}

**15.pink.h**

#ifndef PINK\_H

#define PINK\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <ghost.h>

class pink : public ghost

{

public:

pink();

virtual void move();

};

#endif // PINK\_H

**16. red.cpp**

#include "red.h"

#include "mainwindow.h"

red::red(){

pics[0].push\_back(QPixmap(":/img/img/redu1"));

pics[0].push\_back(QPixmap(":/img/img/redu2"));

pics[1].push\_back(QPixmap(":/img/img/redd1"));

pics[1].push\_back(QPixmap(":/img/img/redd2"));

pics[2].push\_back(QPixmap(":/img/img/redl1"));

pics[2].push\_back(QPixmap(":/img/img/redl2"));

pics[3].push\_back(QPixmap(":/img/img/redr1"));

pics[3].push\_back(QPixmap(":/img/img/redr2"));

pics[4].push\_back(QPixmap(":/img/img/blue1"));

pics[4].push\_back(QPixmap(":/img/img/blue2"));

pics[5].push\_back(QPixmap(":/img/img/white1"));

pics[5].push\_back(QPixmap(":/img/img/white2"));

setPixmap(pics[2][0].scaled(50, 50, Qt::KeepAspectRatio));

setPos(376, 272);

direction = 0;

picsArg = 0;

}

void red::move(){

direction = changeDirection();

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(x() > 792){

setPos(-40, 342);

}

if(x() < -40){

setPos(792, 342);

}

}

17.red.h

#ifndef RED\_H

#define RED\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <ghost.h>

class red : public ghost

{

public:

red();

virtual void move();

};

#endif // RED\_H

**18. smallpoint.cpp**

#include "smallpoint.h"

#include "mainwindow.h"

smallpoint::smallpoint(double X, double Y, int i, int j){

pics.push\_back(QPixmap(":/img/img/smallpoint"));

pics.push\_back(QPixmap(":/img/img/transparent"));

if(map[i][j]){

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

}

else {

setPixmap(pics[1].scaled(50, 50, Qt::KeepAspectRatio));

}

setPos(X, Y);

}

void smallpoint::collide(){

setPixmap(pics[1].scaled(50, 50, Qt::KeepAspectRatio));

}

**19. smallpoint.h**

#ifndef SMALLPOINT\_H

#define SMALLPOINT\_H

#include <QMainWindow>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

class smallpoint : public QGraphicsPixmapItem

{

public:

smallpoint(double X, double Y, int i, int j);

void collide();

private:

bool map[29][26] = {{1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1},

{1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1},

{0,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0},

{1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1},

{1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1},

{1,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,1,1},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1},

{1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1},

{1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1},

{0,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,0},

{0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0},

{0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0},

{1,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,1,1},

{1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1},

{1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1},

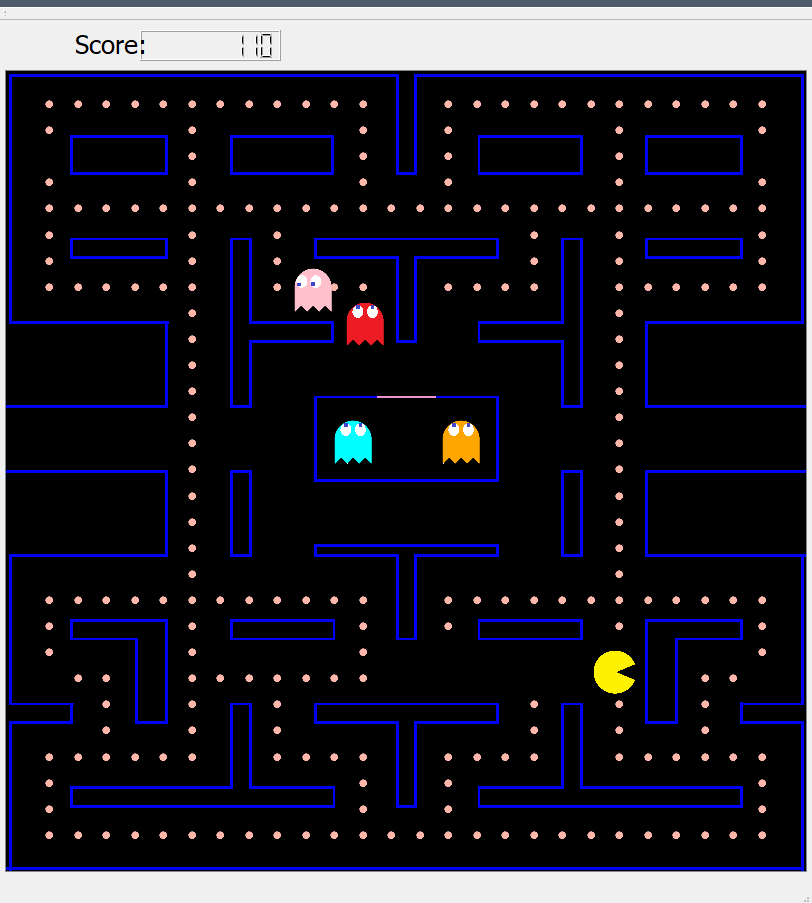
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};

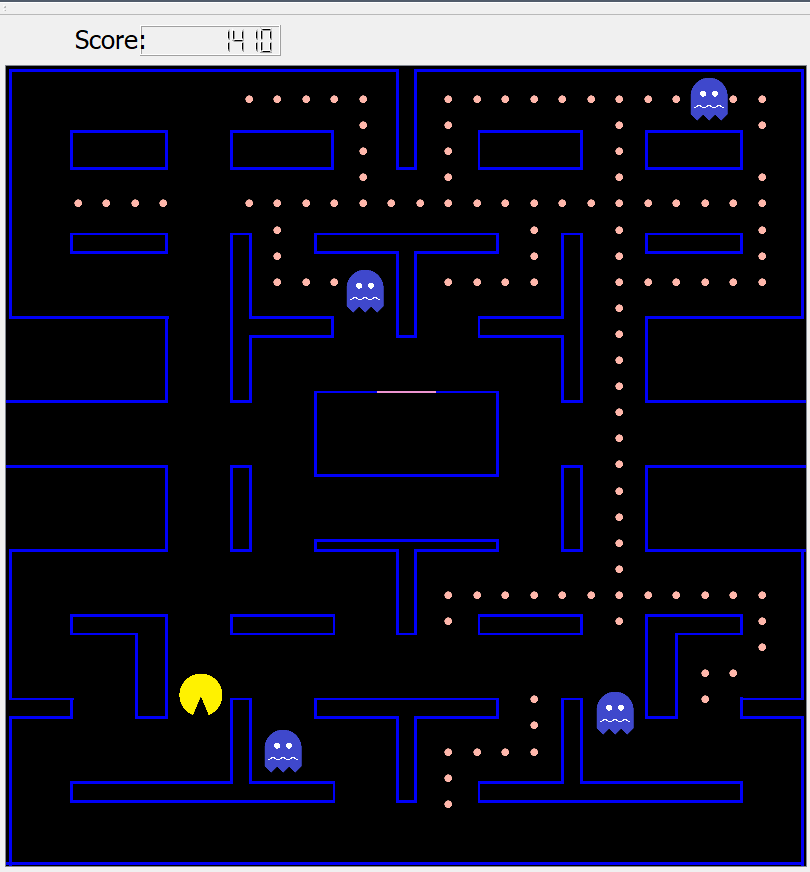
QList<QPixmap> pics;

};

#endif // SMALLPOINT\_H

**Результаты тестирования программы:**

****

****

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