Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

**Отчёт**

Лабораторная работа №1-2

По дисциплине ОСиСП

«Проектирование и разработка приложений»

Выполнил

Студент ФЭИС

3-го курса, группы ПО-5

Бриштен Д.С.

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 4

Задание. 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.

Файлы интерфейса

**ball.h**

#ifndef BALL\_H

#define BALL\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

class **Ball** : public QGraphicsItem

{

public:

**Ball**();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **drawballs**(QPainter \*painter);

void **setpoints**(QVector<QPoint> points);

void **removepoint**(QPoint p);

void **changeGeometry**() { prepareGeometryChange(); }

void **generatePointPixmap**();

int ballx,bally;

int ballw,ballh;

QPixmap ballpix;

QRectF rec;

QVector<QPoint> points;

};

**ghost.h**

#ifndef GHOST\_H

#define GHOST\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

class **Ghost**:public QGraphicsItem

{

public:

**Ghost**();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **advance**();

void **setgosx**(int);

void **setgosy**(int);

void **setDirection**(int dir);

void **setColor**(QString col);

void **changeGeometry**() { prepareGeometryChange(); }

QPixmap left1,left2;

QPixmap up1,up2;

QPixmap down1,down2;

QPixmap right1,right2;

QPixmap scareb,scareb1,scarew,scarew1;

int gosx,gosy;

int direction;

int animestate;

bool is\_Scared,whiteb;

};

#endif // GHOST\_H

**map.h**

#ifndef MAP\_H

#define MAP\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

#include <QVector>

#include "ball.h"

#include "powerball.h"

class **Map** : public QGraphicsItem

{

public:

**Map**();

Ball \*ball;

PowerBall \*powerball;

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **advance**();

void **generateBalls**();

void **AddPathPoints**(int, int, int, int);

bool **canmove**(QPoint);

void **fillpacpoints**(int pacx,int pacy);

void **setballpoints**(QVector<QPoint> points);

void **setpowerballpoints**(QVector<QPoint> points);

void **changeGeometry**() { prepareGeometryChange(); }

QVector<QPoint> **getballpoints**();

QVector<QPoint> **getpowerballpoints**();

QPixmap mappic;

QVector<QPoint> pacpoints,ballpoints;

QVector<QPoint> powerballpoints;

};

#endif // MAP\_H

**pacman.h**

#ifndef PACMAN\_H

#define PACMAN\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

class **Pacman** : public QGraphicsItem

{

public:

**Pacman**();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **advance**();

void **setpacx**(int);

void **setpacy**(int);

void **setDirection**(int dir);

void **changeGeometry**() { prepareGeometryChange(); }

QPixmap left1,left2,left3,left4;

QPixmap up1,up2,up3,up4;

QPixmap down1,down2,down3,down4;

QPixmap right1,right2,right3,right4;

int pacx,pacy;

int direction;

int animestate;

};

## #endif // PACMAN\_H

**powerball.h**

#ifndef POWERBALL\_H

#define POWERBALL\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

class **PowerBall** : public QGraphicsItem

{

public:

**PowerBall**();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **setpoints**(QVector<QPoint> points);

void **drawballs**(QPainter \*painter);

void **removepoint**(QPoint p);

void **generatePointPixmap**();

void **changeGeometry**() { prepareGeometryChange(); }

int ballx,bally;

int ballw,ballh;

QVector<QPoint> points;

QPixmap pBallPix;

};

## #endif // POWERBALL\_H

**textdrawing.h**

#ifndef TEXTDRAWING\_H

#define TEXTDRAWING\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

class **Textdrawing** : public QGraphicsItem

{

public:

**Textdrawing**();

QRectF ***boundingRect***() const;

void **SetOver**(bool over);

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

bool over,playing;

int x,y,w,h,score;

};

## #endif // TEXTDRAWING\_H

**window.h**

#ifndef WINDOW\_H

#define WINDOW\_H

#include <QDialog>

#include <QtCore>

#include <QtGui>

#include "pacman.h"

#include "map.h"

#include "ball.h"

#include "ghost.h"

#include "powerball.h"

#include "textdrawing.h"

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**();

Pacman \*pacman;

Ghost \*ghost;

Ghost \*ghost1;

Ghost \*ghost2;

Map \*pac\_map;

Ball \*ball;

PowerBall \*powerball;

Textdrawing \*text;

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**();

protected:

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

private:

Ui::window \*ui;

QGraphicsScene \*scene;

QTimer \*timer;

QTimer \*ghoststimer;

};

## #endif // WINDOW\_H

Файлы реализации

**ball.cpp**

#include "ball.h"

Ball::**Ball**()

{

ballx=0;

bally=0;

ballw=5;

ballh=5;

generatePointPixmap();

}

QRectF Ball::***boundingRect***() const

{

return QRect(0, 0, 450, 550);

}

void Ball::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

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

painter->drawPixmap( points[i].x(), points[i].y(), ballpix );

}

}

void Ball::**drawballs**(QPainter \*painter)

{

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

painter->drawPixmap( points[i].x(), points[i].y(), ballpix );

}

}

void Ball::**setpoints**(QVector<QPoint> points)

{

this->points.clear();

this->points=points;

}

void Ball::**removepoint**(QPoint p)

{

}

void Ball::**generatePointPixmap**()

{

QRect bounds = QRect(0, 0, 3, 3);

QPainter painter;

ballpix = QPixmap(bounds.size());

ballpix.fill(Qt::transparent);

painter.begin(*&ballpix*);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(Qt::NoPen);

painter.setBrush(Qt::yellow);

painter.drawEllipse(0, 0, 3, 3);

## }

**ghost.cpp**

#include "ghost.h"

Ghost::**Ghost**()

{

animestate=0;

gosx=450/2;

gosy=480/2;

direction=1;

is\_Scared=false;

whiteb=false;

right1.load(":/images/ghostright1.png");

right2.load(":/images/ghostright2.png");

up1.load(":/images/ghostup1.png");

up2.load(":/images/ghostup2.png");

down1.load(":/images/ghostdown1.png");

down2.load(":/images/ghostdown2.png");

left1.load(":/images/ghostleft1.png");

left2.load(":/images/ghostleft2.png");

scareb.load(":/images/ghostscaredblue1.png");

scareb1.load(":/images/ghostscaredblue2.png");

scarew.load(":/images/ghostscaredwhite1.png");

scarew1.load(":/images/ghostscaredwhite2.png");

}

QRectF Ghost::***boundingRect***() const

{

return QRect(gosx-15,gosy-15,20,20);

}

void Ghost::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

if(!is\_Scared){

switch(direction){

case 1:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,left1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,left2);

}

break;

case 4:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,right1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,right2);

}

break;

case 3:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,down1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,down2);

}

break;

case 2:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,up1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,up2);

}

break;

}

}else{

if(whiteb){

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,scarew1);

}

}else{

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb1);

}

}

}

}

void Ghost::**advance**()

{

if(animestate>2){

animestate=0;

}else{

animestate++;

}

}

void Ghost::**setgosx**(int x)

{

gosx=x;

}

void Ghost::**setgosy**(int y)

{

gosy=y;

}

void Ghost::**setDirection**(int dir)

{

direction=dir;

}

void Ghost::**setColor**(QString col)

{

if(col=="blue"){

right1.load(":/images/ghostrightblue1.png");

right2.load(":/images/ghostrightblue2.png");

up1.load(":/images/ghostupblue1.png");

up2.load(":/images/ghostupblue2.png");

down1.load(":/images/ghostdownblue1.png");

down2.load(":/images/ghostdownblue2.png");

left1.load(":/images/ghostleftblue1.png");

left2.load(":/images/ghostleftblue2.png");

}else if(col=="orange"){

right1.load(":/images/ghostrightorange1.png");

right2.load(":/images/ghostrightorange2.png");

up1.load(":/images/ghostuporange1.png");

up2.load(":/images/ghostuporange2.png");

down1.load(":/images/ghostdownorange1.png");

down2.load(":/images/ghostdownorange2.png");

left1.load(":/images/ghostleftorange1.png");

left2.load(":/images/ghostleftorange2.png");

}else if(col=="red"){

right1.load(":/images/ghostrightred1.png");

right2.load(":/images/ghostrightred2.png");

up1.load(":/images/ghostupred1.png");

up2.load(":/images/ghostupred2.png");

down1.load(":/images/ghostdownred1.png");

down2.load(":/images/ghostdownred2.png");

left1.load(":/images/ghostleftred1.png");

left2.load(":/images/ghostleftred2.png");

}

## }

**main.cpp**

#include <QApplication>

#include "window.h"

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

{

QApplication a(argc, argv);

window w;

w.show();

return a.exec();

## }

**map.cpp**

#include "map.h"

Map::**Map**()

{

mappic.load(":/images/pac\_map.png");

ball=new Ball;

powerball=new PowerBall;

AddPathPoints(30, 30, 200, 30);

AddPathPoints(250, 30, 420, 30);

AddPathPoints(30, 90, 420, 90);

AddPathPoints(30, 130, 110, 130);

AddPathPoints(150, 130, 200, 130);

AddPathPoints(250, 130, 300, 130);

AddPathPoints(340, 130, 420, 130);

AddPathPoints(150, 180, 300, 180);

AddPathPoints(0, 230, 150, 230);

AddPathPoints(300, 230, 450, 230);

AddPathPoints(150, 270, 300, 270);

AddPathPoints(30, 320, 200, 320);

AddPathPoints(250, 320, 420, 320);

AddPathPoints(30, 360, 60, 360);

AddPathPoints(110, 360, 340, 360);

AddPathPoints(390, 360, 420, 360);

AddPathPoints(30, 410, 110, 410);

AddPathPoints(150, 410, 200, 410);

AddPathPoints(250, 410, 300, 410);

AddPathPoints(340, 410, 420, 410);

AddPathPoints(30, 450, 420, 450);

AddPathPoints(30, 30, 30, 130);

AddPathPoints( 30, 320, 30, 360);

AddPathPoints(30, 410, 30, 450);

AddPathPoints( 60, 360, 60, 410);

AddPathPoints(110, 30, 110, 410);

AddPathPoints( 150, 90, 150, 130);

AddPathPoints(150, 180, 150, 320);

AddPathPoints(150, 360, 150, 410);

AddPathPoints( 200, 30, 200, 90);

AddPathPoints(200, 135, 200, 180);

AddPathPoints( 200, 320, 200, 360);

AddPathPoints( 200, 410, 200, 450);

AddPathPoints( 250, 30, 250, 90);

AddPathPoints( 250, 135, 250, 180);

AddPathPoints( 250, 320, 250, 360);

AddPathPoints( 250, 410, 250, 450);

AddPathPoints( 300, 90, 300, 130);

AddPathPoints( 300, 180, 300, 320);

AddPathPoints( 300, 360, 300, 410);

AddPathPoints( 340, 30, 340, 410);

AddPathPoints( 390, 360, 390, 410);

AddPathPoints( 420, 30, 420, 130);

AddPathPoints( 420, 320, 420, 360);

AddPathPoints( 420, 410, 420, 450);

QPoint p;

QPoint p1,p2,p3,p4;

p1.setX(30);

p1.setY(450);

p2.setX(30);

p2.setY(35);

p3.setX(420);

p3.setY(35);

p4.setX(420);

p4.setY(450);

powerballpoints.push\_front(p1);

powerballpoints.push\_front(p2);

powerballpoints.push\_front(p3);

powerballpoints.push\_front(p4);

for (int i=0; i<450-1; i++)

{

for(int j=0;j<480-1;j++){

p.setX(i);

p.setY(j);

if(j%10==0 && i%10==0){

if (pacpoints.contains(p)){

if(p!=p1 && p!=p2 && p!=p3 && p!=p4){

if(p.x()>0){

ballpoints.push\_front(p);

}

}

}

}

}

}

generateBalls();

}

QRectF Map::***boundingRect***() const

{

return QRect(0,0,450,480);

}

void Map::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

painter->drawPixmap(0,0,450,480,mappic);

QPen pen=QPen(Qt::red);

pen.setWidth(2);

painter->setPen(pen);

}

void Map::**advance**()

{

}

void Map::**generateBalls**()

{

QPainter painter;

painter.begin(*&mappic*);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(Qt::NoPen);

painter.setBrush(Qt::yellow);

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

painter.drawEllipse(ballpoints[i].x(),ballpoints[i].y(),3,3);

}

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

painter.drawEllipse(powerballpoints[i].x()-5,powerballpoints[i].y()-5,10,10);

}

}

bool Map::**canmove**(QPoint point)

{

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

if(pacpoints[i]==point)

{

return true;

}

}

return false;

}

void Map::**fillpacpoints**(int pacx, int pacy)

{

QPainter painter;

QRect rec(pacx,pacy,3,3);

QRect rec2(pacx-5,pacy-5,10,10);

painter.begin(*&mappic*);

painter.fillRect(rec,Qt::black);

if((pacx==powerballpoints[0].x() && pacy==powerballpoints[0].y())||

(pacx==powerballpoints[1].x() && pacy==powerballpoints[1].y())||

(pacx==powerballpoints[2].x() && pacy==powerballpoints[2].y())||

(pacx==powerballpoints[3].x() && pacy==powerballpoints[3].y())){

painter.fillRect(rec2,Qt::black);

}

painter.end();

}

void Map::**setballpoints**(QVector<QPoint> points)

{

ball->setpoints(points);

}

void Map::**setpowerballpoints**(QVector<QPoint> points)

{

powerball->setpoints(points);

}

QVector<QPoint> Map::**getballpoints**()

{

return ballpoints;

}

QVector<QPoint> Map::**getpowerballpoints**()

{

return powerballpoints;

}

void Map::**AddPathPoints**(int x1, int y1, int x2, int y2)

{

QPoint p;

if (x1 == x2)

{

if (y1 < y2)

{

for (int y=y1; y<y2+1; y++)

{

p.setX(x1);

p.setY(y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

else

{

for (int y=y1; y>y2-1; y--)

{

p.setX(x1);

p.setY(y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

}

else

{

if (x1 < x2)

{

for (int x=x1; x<x2+1; x++)

{

p.setX(x);

p.setY(y1);

if (! pacpoints.contains(p)){

pacpoints.push\_front(p);

}

}

}

else

{

for (int x=x1; x>x2-1; x--)

{

p.setX(x);

p.setY(y1);

if (! pacpoints.contains(p)){

pacpoints.push\_front(p);

}

}

}

}

## }

**pacman.cpp**

#include "pacman.h"

#include <QMatrix>

Pacman::**Pacman**()

{

animestate=0;

pacx=410/2;

pacy=360;

direction=4;

right1.load(":/images/pacclose.png");

right2.load(":/images/pacopen1.png");

right3.load(":/images/pacopen2.png");

right4.load(":/images/pacopen3.png");

up1.load(":/images/paccloseu.png");

up2.load(":/images/pacopen1u.png");

up3.load(":/images/pacopen2u.png");

up4.load(":/images/pacopen3u.png");

down1.load(":/images/pacclosed.png");

down2.load(":/images/pacopen1d.png");

down3.load(":/images/pacopen2d.png");

down4.load(":/images/pacopen3d.png");

left1.load(":/images/pacclosel.png");

left2.load(":/images/pacopen1l.png");

left3.load(":/images/pacopen2l.png");

left4.load(":/images/pacopen3l.png");

}

QRectF Pacman::***boundingRect***() const

{

return QRect(pacx-15, pacy-15, 20, 20);

}

void Pacman::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

switch(direction){

case 1:

if(animestate<2){

painter->drawPixmap(pacx-15,pacy-15,30,30,left1);

}else if(animestate<4){

painter->drawPixmap(pacx-15,pacy-15,30,30,left2);

}else if(animestate<6){

painter->drawPixmap(pacx-15,pacy-15,30,30,left3);

}else if(animestate<8){

painter->drawPixmap(pacx-15,pacy-15,30,30,left4);

}

break;

case 4:

if(animestate<2){

painter->drawPixmap(pacx-15,pacy-15,30,30,right1);

}else if(animestate<4){

painter->drawPixmap(pacx-15,pacy-15,30,30,right2);

}else if(animestate<6){

painter->drawPixmap(pacx-15,pacy-15,30,30,right3);

}else if(animestate<8){

painter->drawPixmap(pacx-15,pacy-15,30,30,right4);

}

break;

case 3:

if(animestate<2){

painter->drawPixmap(pacx-15,pacy-15,30,30,down1);

}else if(animestate<4){

painter->drawPixmap(pacx-15,pacy-15,30,30,down2);

}else if(animestate<6){

painter->drawPixmap(pacx-15,pacy-15,30,30,down3);

}else if(animestate<8){

painter->drawPixmap(pacx-15,pacy-15,30,30,down4);

}

break;

case 2:

if(animestate<2){

painter->drawPixmap(pacx-15,pacy-15,30,30,up1);

}else if(animestate<4){

painter->drawPixmap(pacx-15,pacy-15,30,30,up2);

}else if(animestate<6){

painter->drawPixmap(pacx-15,pacy-15,30,30,up3);

}else if(animestate<8){

painter->drawPixmap(pacx-15,pacy-15,30,30,up4);

}

break;

}

}

void Pacman::**advance**()

{

if(animestate>6){

animestate=0;

}else{

animestate++;

}

}

void Pacman::**setpacx**(int x)

{

pacx=x;

}

void Pacman::**setpacy**(int y)

{

pacy=y;

}

void Pacman::**setDirection**(int dir)

{

direction=dir;

## }

**powerball.cpp**

#include "powerball.h"

PowerBall::**PowerBall**()

{

ballx=0;

bally=0;

ballw=5;

ballh=5;

generatePointPixmap();

}

QRectF PowerBall::***boundingRect***() const

{

return QRect(0, 0, 450, 550);

}

void PowerBall::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

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

painter->drawPixmap(points[i].x()-5, points[i].y()-5, pBallPix);

}

}

void PowerBall::**setpoints**(QVector<QPoint> points)

{

this->points.clear();

this->points=points;

}

void PowerBall::**drawballs**(QPainter \*painter)

{

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

painter->drawPixmap(points[i].x()-5, points[i].y()-5, pBallPix);

}

}

void PowerBall::**generatePointPixmap**()

{

QRect bounds = QRect(0, 0, 10, 10);

QPainter painter;

pBallPix = QPixmap(bounds.size());

pBallPix.fill(Qt::transparent);

painter.begin(*&pBallPix*);

QPen pen=QPen(Qt::yellow);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(pen);

painter.setBrush(Qt::red);

painter.drawEllipse(0, 0, 10, 10);

## }

**textdrawing.cpp**

#include "textdrawing.h"

Textdrawing::**Textdrawing**()

{

over=false;

w=350;

h=50;

x=450/2-w/2;

y=480/2-h/2;

score=0;

playing=false;

}

QRectF Textdrawing::***boundingRect***() const

{

return QRect(x, y, w, h);

}

void Textdrawing::**SetOver**(bool over)

{

this->over=over;

}

void Textdrawing::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

QPen pen(Qt::blue);

painter->setPen(pen);

QFont font=painter->font() ;

QBrush brush(Qt::blue);

painter->setBrush(brush);

if(!playing){

font.setPointSize ( 18 );

painter->setFont(font);

if(over){

painter->drawText( x+90,y-70, "GAME OVER." );

painter->drawText( x+20,y+120, "PRESS ENTER TO PLAY." );

painter->drawText(*boundingRect*(),Qt::AlignCenter, "SCORE : "+QString::number(score) );

}else{

painter->drawText(*boundingRect*(),Qt::AlignCenter, "PRESS SPACE TO START");

}

}else{

font.setPointSize (13);

painter->setFont(font);

painter->drawText( 50,15, "SCORE : "+QString::number(score));

}

## }

**window.cpp**

#include "window.h"

#include "ui\_window.h"

#include <QThread>

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);

}

void window::**start\_Game**()

{

gosx=450/2;

gosy=480/2;

gosx1=450/2;

gosy1=480/2;

gosx2=450/2;

gosy2=480/2;

pacx=410/2;

pacy=360;

pac\_map=new Map;

ghost=new Ghost;

ghost1=new Ghost;

ghost2=new Ghost;

ghost1->setColor("orange");

ghost2->setColor("red");

ballpoints=pac\_map->getballpoints();

Powerballpoints=pac\_map->getpowerballpoints();

ball=new Ball;

powerball=new PowerBall;

ball->setpoints(ballpoints);

scene->removeItem(*text*);

powerball->setpoints(Powerballpoints);

scene->addItem(*pac\_map*);

scene->addItem(*pacman*);

scene->addItem(*ghost*);

scene->addItem(*ghost1*);

scene->addItem(*ghost2*);

scene->addItem(*text*);

text->playing=true;

ghostmoving=false;

ghostmoving1=false;

ghostmoving2=false;

direction=0;

moving=false;

ghoststart=false;

ghoststart1=false;

ghoststart2=false;

state=0;

state1=0;

state2=0;

state3=0;

timer = new QTimer(this);

ghoststimer=new QTimer(this);

connect(timer, SIGNAL(timeout()), this,SLOT(updater()));

connect(ghoststimer, SIGNAL(timeout()), this,SLOT(ghostupdater()));

this->show();

scene->update();

this->update();

delayb=true;

timer->start(30);

ghoststimer->start(45);

this->setFocus();

}

void window::**delay**()

{

QTime dieTime= QTime::currentTime().addSecs(2);

while( QTime::currentTime() < dieTime ){}

delayb=false;

}

void window::**end\_Game**()

{

text->over=true;

text->playing=false;

scene->removeItem(*pac\_map*);

scene->removeItem(*pacman*);

scene->removeItem(*ghost*);

scene->removeItem(*ghost1*);

scene->removeItem(*ghost2*);

score=0;

playing=false;

scene->update();

}

void window::**pacman\_move**()

{

QPoint p;

if(nextdirection!=direction){

switch(nextdirection)

{

case 1:

p.setX(pacx-5);

p.setY(pacy);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 4:

p.setX(pacx+5);

p.setY(pacy);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 3:

p.setX(pacx);

p.setY(pacy+5);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 2:

p.setX(pacx);

p.setY(pacy-5);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

}

}

switch(direction)

{

case 1:

p.setX(pacx-5);

p.setY(pacy);

pacman->setDirection(direction);

if(pac\_map->canmove(p)){

pacx-=5;

moving=true;

}else{

moving=false;

}

break;

case 4:

pacman->setDirection(direction);

p.setX(pacx+5);

p.setY(pacy);

if(pac\_map->canmove(p)){

pacx+=5;

moving=true;

}else{

moving=false;

}

break;

case 3:

pacman->setDirection(direction);

p.setX(pacx);

p.setY(pacy+5);

if(pac\_map->canmove(p)){

pacy+=5;

moving=true;

}else{

moving=false;

}

break;

case 2:

pacman->setDirection(direction);

p.setX(pacx);

p.setY(pacy-5);

if(pac\_map->canmove(p)){

pacy-=5;

moving=true;

}else{

moving=false;

}

break;

}

if(pacx<=0){

pacx=450;

pacy=230;

}else if(pacx>=450){

pacx=0;

pacy=230;

}

pacman->setpacx(pacx);

pacman->setpacy(pacy);

}

void window::**ghostsmove**()

{

QPoint p;

if(!ghostmoving){

ghostdir=(qrand()%4)+1;

}else{

if(ghostdir==4 &&gosy<pacy || ghostdir==1 && gosy>pacy){

if(ghostdir==1 && gosy>pacy){

nextghostdir=2;

}else if(ghostdir==4 &&gosy<pacy){

nextghostdir=3;

}

}else if(ghostdir==3 && gosx<pacx || ghostdir==2 && gosx>pacx){

if(ghostdir==2 && gosx>pacx){

nextghostdir=1;

}else if(ghostdir==3 && gosx<pacx){

nextghostdir=4;

}

}

}

if(nextghostdir!=ghostdir){

switch(nextghostdir)

{

case 1:

p.setX(gosx-5);

p.setY(gosy);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 4:

p.setX(gosx+5);

p.setY(gosy);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 3:

p.setX(gosx);

p.setY(gosy+5);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 2:

p.setX(gosx);

p.setY(gosy-5);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

}

}

switch(ghostdir)

{

case 1:

p.setX(gosx-5);

p.setY(gosy);

ghost->setDirection(ghostdir);

if(pac\_map->canmove(p)){

gosx-=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 4:

ghost->setDirection(ghostdir);

p.setX(gosx+5);

p.setY(gosy);

if(pac\_map->canmove(p)){

gosx+=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 3:

ghost->setDirection(ghostdir);

p.setX(gosx);

p.setY(gosy+5);

if(pac\_map->canmove(p)){

gosy+=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 2:

ghost->setDirection(ghostdir);

p.setX(gosx);

p.setY(gosy-5);

if(pac\_map->canmove(p)){

gosy-=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

}

if(gosx<=0){

gosx=450;

gosy=230;

}else if(gosx>=450){

gosx=0;

gosy=230;

}

ghost->setgosx(gosx);

ghost->setgosy(gosy);

}

void window::**ghostsmove1**()

{

QPoint p;

if(!ghostmoving1){

ghostdir1=(qrand()%4)+1;

}else{

if(ghostdir1==4 &&gosy1<pacy || ghostdir1==1 && gosy1>pacy){

if(ghostdir1==1 && gosy>pacy){

nextghostdir1=2;

}else if(ghostdir1==4 &&gosy1<pacy){

nextghostdir1=3;

}

}else if(ghostdir1==3 && gosx1<pacx || ghostdir1==2 && gosx1>pacx){

if(ghostdir1==2 && gosx1>pacx){

nextghostdir1=1;

}else if(ghostdir1==3 && gosx1<pacx){

nextghostdir1=4;

}

}

}

if(nextghostdir1!=ghostdir1){

switch(nextghostdir1)

{

case 1:

p.setX(gosx1-5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 4:

p.setX(gosx1+5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 3:

p.setX(gosx1);

p.setY(gosy1+5);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 2:

p.setX(gosx1);

p.setY(gosy1-5);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

}

}

switch(ghostdir1)

{

case 1:

p.setX(gosx1-5);

p.setY(gosy1);

ghost1->setDirection(ghostdir1);

if(pac\_map->canmove(p)){

gosx1-=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 4:

ghost1->setDirection(ghostdir1);

p.setX(gosx1+5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

gosx1+=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 3:

ghost1->setDirection(ghostdir1);

p.setX(gosx1);

p.setY(gosy1+5);

if(pac\_map->canmove(p)){

gosy1+=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 2:

ghost1->setDirection(ghostdir1);

p.setX(gosx1);

p.setY(gosy1-5);

if(pac\_map->canmove(p)){

gosy1-=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

}

if(gosx1<=0){

gosx1=450;

gosy1=230;

}else if(gosx1>=450){

gosx1=0;

gosy1=230;

}

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

}

void window::**ghostsmove2**()

{

QPoint p;

if(!ghostmoving2){

ghostdir2=(qrand()%4)+1;

}else{

if(ghostdir2==4 &&gosy2<pacy || ghostdir2==1 && gosy2>pacy){

if(ghostdir2==1 && gosy2>pacy){

nextghostdir2=2;

}else if(ghostdir2==4 &&gosy2<pacy){

nextghostdir2=3;

}

}else if(ghostdir2==3 && gosx2<pacx || ghostdir2==2 && gosx2>pacx){

if(ghostdir2==2 && gosx2>pacx){

nextghostdir2=1;

}else if(ghostdir2==3 && gosx2<pacx){

nextghostdir2=4;

}

}

}

if(nextghostdir2!=ghostdir2){

switch(nextghostdir2)

{

case 1:

p.setX(gosx2-5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 4:

p.setX(gosx2+5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 3:

p.setX(gosx2);

p.setY(gosy2+5);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 2:

p.setX(gosx2);

p.setY(gosy2-5);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

}

}

switch(ghostdir2)

{

case 1:

p.setX(gosx2-5);

p.setY(gosy2);

ghost2->setDirection(ghostdir2);

if(pac\_map->canmove(p)){

gosx2-=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 4:

ghost2->setDirection(ghostdir2);

p.setX(gosx2+5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

gosx2+=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 3:

ghost2->setDirection(ghostdir2);

p.setX(gosx2);

p.setY(gosy2+5);

if(pac\_map->canmove(p)){

gosy2+=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 2:

ghost2->setDirection(ghostdir2);

p.setX(gosx2);

p.setY(gosy2-5);

if(pac\_map->canmove(p)){

gosy2-=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

}

if(gosx2<=0){

gosx2=450;

gosy2=230;

}else if(gosx2>=450){

gosx2=0;

gosy2=230;

}

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

}

void window::**moveghostsinrect2**()

{

if(gosx2==450/2+40 || gosx2==450/2-40){

if(ghostdir2==4){

ghostdir2=1;

}else{

ghostdir2=4;

}

}

if(ghostdir2==4){

gosx2+=5;

}else{

gosx2-=5;

}

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

}

void window::**moveghostsinrect1**()

{

if(gosx1==450/2+40 || gosx1==450/2-40){

if(ghostdir1==4){

ghostdir1=1;

}else{

ghostdir1=4;

}

}

if(ghostdir1==4){

gosx1+=5;

}else{

gosx1-=5;

}

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

}

void window::**moveghostsinrect**()

{

if(gosx==450/2+40 || gosx==450/2-40){

if(ghostdir==4){

ghostdir=1;

}else{

ghostdir=4;

}

}

if(ghostdir==4){

gosx+=5;

}else{

gosx-=5;

}

ghost->setgosx(gosx);

ghost->setgosy(gosy);

}

void window::**checklost**()

{

if(ballpoints.isEmpty()){

timer->stop();

ghoststimer->stop();

end\_Game();

}

if(pacman->*collidesWithItem*(ghost) ||

pacman->*collidesWithItem*(ghost1) ||

pacman->*collidesWithItem*(ghost2)){

if(scared){

score+=100;

if(pacman->*collidesWithItem*(ghost)){

gosx=450/2;

gosy=450/2;

ghoststart=false;

}

if(pacman->*collidesWithItem*(ghost1)){

gosx1=450/2;

gosy1=450/2;

ghoststart1=false;

}

if(pacman->*collidesWithItem*(ghost2)){

gosx2=450/2;

gosy2=450/2;

ghoststart2=false;

}

}else{

delay();

timer->stop();

ghoststimer->stop();

end\_Game();

}

}

}

void window::***keyPressEvent***(QKeyEvent \*event)

{

QPoint p;

switch(event->key())

{

case Qt::Key\_Left:

if(!moving){

direction=1;

}else{

nextdirection=1;

}

break;

case Qt::Key\_Right:

if(!moving){

direction=4;

}else{

nextdirection=4;

}

break;

case Qt::Key\_Down:

if(!moving){

direction=3;

}else{

nextdirection=3;

}

break;

case Qt::Key\_Up:

if(!moving){

direction=2;

}else{

nextdirection=2;

}

break;

case Qt::Key\_Space:

if(!playing){

start\_Game();

playing=true;

}else{

end\_Game();

start\_Game();

}

break;

default:

break;

}

}

int scarestate=0;

void window::**updater**()

{

checklost();

pacman\_move();

for(int i=0;i<ballpoints.size();i++){ // обычные мячи

if(pacman->pacx==ballpoints[i].x() && pacman->pacy==ballpoints[i].y()){

ballpoints.remove(i);

score++;

text->score=score;

}

}

for(int i=0;i<Powerballpoints.size();i++){ // большие мячи

if(pacman->pacx==Powerballpoints[i].x() && pacman->pacy==Powerballpoints[i].y()){

Powerballpoints.remove(i);

if(scared){scarestate=0;}

scared=true;

}

}

if(scared){

ghost->is\_Scared=true;

ghost1->is\_Scared=true;

ghost2->is\_Scared=true;

scarestate+=1;

if(scarestate==1){ghoststimer->setInterval(80);} // скорость синих призраков

if(scarestate==450){

ghost->whiteb=true;

ghost1->whiteb=true;

ghost2->whiteb=true;

}

if(scarestate==500){

scared=false; scarestate=0; ghoststimer->setInterval(45);

ghost->whiteb=false;

ghost1->whiteb=false;

ghost2->whiteb=false;

}

}else{

ghost->is\_Scared=false;

ghost1->is\_Scared=false;

ghost2->is\_Scared=false;

}

ball->setpoints(ballpoints);

powerball->setpoints(Powerballpoints);

pac\_map->setballpoints(ballpoints);

pac\_map->setpowerballpoints(Powerballpoints);

pac\_map->fillpacpoints(pacx,pacy);

pacman->advance();

ghost->advance();

ghost1->advance();

ghost2->advance();

this->setFocus();

scene->update(pac\_map->*boundingRect*());

if(delayb){delay();}

}

void window::**ghostupdater**()

{

if(ghoststart || ghoststart1 || ghoststart2){

if(ghoststart){ghostsmove();}

if(ghoststart1){ghostsmove1();}

if(ghoststart2){ghostsmove2();}

}

if(gosx2==450/2){ // gosx3->gosx2

state++;

}

if(state==3){

start=true;

state=0;

}

if(!start){

moveghostsinrect2();

moveghostsinrect1();

moveghostsinrect();

}else if(!ghoststart2 || !ghoststart1 || !ghoststart){

QPoint p,p1,p2;

if(!ghoststart2){

gosy2-=5;

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

p2.setX(gosx2);

p2.setY(gosy2);

if(pac\_map->pacpoints.contains(p2)){ghoststart2=true;}

}

if(!ghoststart1){

gosy1-=5;

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

p1.setX(gosx1);

p1.setY(gosy1);

if(pac\_map->pacpoints.contains(p1)){ghoststart1=true;}

}

if(!ghoststart){

gosy-=5;

ghost->setgosx(gosx);

ghost->setgosy(gosy);

p.setX(gosx);

p.setY(gosy);

if(pac\_map->pacpoints.contains(p)){ghoststart=true;}

}

}

if(delayb){delay();}

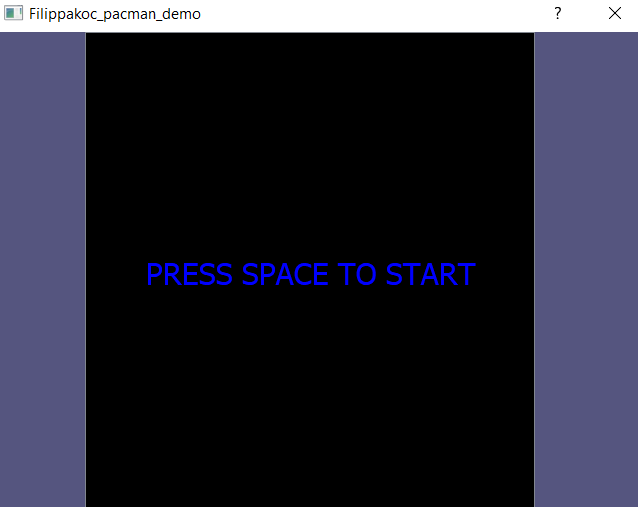
}

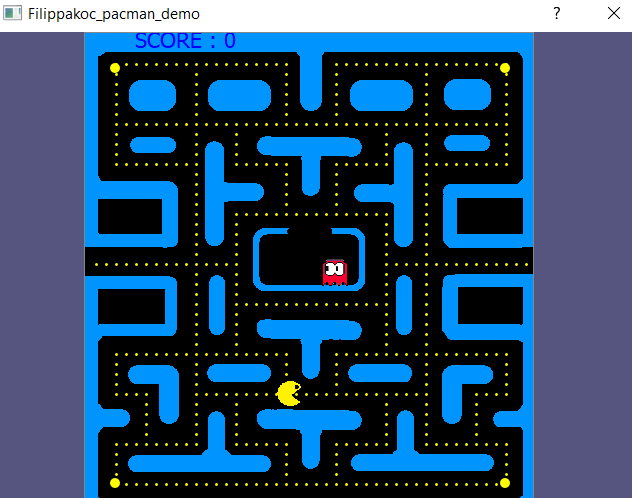
window::~***window***()

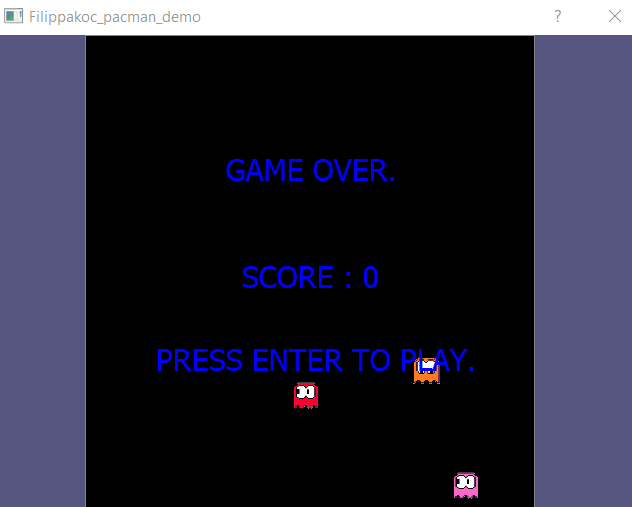
{

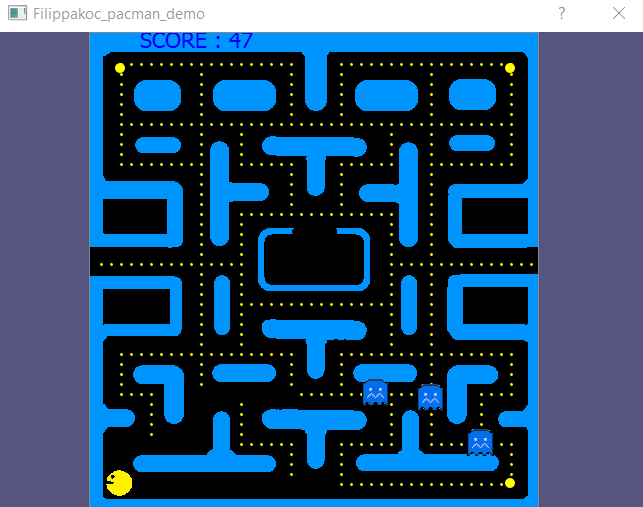
delete ui;

}







****

Вывод: Разработана программ с графическим пользовательским интерфейсом, с использованием фреймворка Qt.