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»

Выполнил:

студент 3 курса

группы ПО-4(1)

Калиновский В. Е.

Проверила:

Дряпко А.В.

Брест 2021

Лабораторная работа №1-2

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

Вариант 11

Задание:

Реализовать игру «Сапер».

**Текст программы:**

**minemine.pro**

QT += widgets

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

SOURCES += \

MyHeader.cpp \

main.cpp \

mainwindow.cpp

HEADERS += \

MyHeader.h \

mainwindow.h

FORMS += \

mainwindow.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

1. **mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QVBoxLayout>

#include <QHBoxLayout>

#include <QGridLayout>

#include <QPushButton>

#include <QLabel>

#include <QPushButton>

#include <QComboBox>

#include <QTimer>

#include <QMap>

#include <QMessageBox>

#include <QString>

#include "MyHeader.h"

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private:

QHBoxLayout\* firstLayout;

QGridLayout\* gameLayout;

QVBoxLayout\* mainLayout;

QComboBox\* difficulties;

QFont font;

MyButton\*\*\* buttons;

QPushButton\* resetButton;

QPushButton\* showHighscores;

QString\*\* buttonText;

QLabel\* flags;

QLabel\* timeLabel;

QTimer\* timer;

double time;

int difficulty;

int buttonRows, buttonColumns;

int numberOfBombs;

int disabledButtons;

int numberOfFlags;

bool endOfGame;

QMap <double, QString> highscoresEasy;

QMap <double, QString> highscoresMedium;

QMap <double, QString> highscoresHard;

QMessageBox\* top10;

QComboBox\* temp;

void **clearButtons**();

void **setButtons**(int n, int m, int size);

void **setBombs**();

void **setNumbers**();

void **onFreeButtonClicked**(int i, int j);

void **bombClicked**();

void **win**();

void **reset**();

void **showTop10**(int difficulty);

QComboBox\* **createComboBox**();

private:

Ui::MainWindow \*ui;

private slots:

void **difficultyChanged**();

void **buttonClicked**();

void **rightButtonClicked**();

void **tick**();

void **on\_actionAbout\_triggered**();

void **on\_actionFont\_triggered**();

};

#endif // MAINWINDOW\_H

1. **MyHeader.h**

#ifndef MYHEADER\_H

#define MYHEADER\_H

#include <QPushButton>

#include <QMouseEvent>

class MyButton: public QPushButton

{

Q\_OBJECT

public:

MyButton(QWidget \*parent = nullptr);

~MyButton();

protected:

void mousePressEvent(QMouseEvent \*e) override;

signals:

void rightClick();

};

#endif // MYHEADER\_H

1. **main.cpp**

#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"

#include <QDebug>

#include <QDateTime>

#include <QInputDialog>

#include <QMapIterator>

#include <QSettings>

#include <QFontDialog>

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

QWidget \*w = new QWidget;

setCentralWidget(*w*);

setWindowTitle("Minesweeper");

w->setStyleSheet("background-color:green");

font.setFamily("Comic Sans MS");

mainLayout = new QVBoxLayout;

firstLayout = new QHBoxLayout;

firstLayout->*setSpacing*(50);

gameLayout = new QGridLayout;

gameLayout->*setSpacing*(0);

gameLayout->setContentsMargins(0, 0, 0, 0);

mainLayout->addLayout(*firstLayout*);

mainLayout->addLayout(*gameLayout*);

w->setLayout(*mainLayout*);

time = 0;

disabledButtons = 0;

difficulty = 0;

endOfGame = false;

difficulties = createComboBox();

firstLayout->addWidget(*difficulties*);

flags = new QLabel;

font = flags->font();

font.setPointSize(font.pointSize() + 8);

flags->setFont(font);

flags->setText("Flags: 10");

firstLayout->addWidget(*flags*);

timeLabel = new QLabel("0:0");

timeLabel->setFont(font);

timer = new QTimer;

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

firstLayout->addWidget(*timeLabel*);

showHighscores = new QPushButton("Highscores");

showHighscores->setFont(font);

showHighscores->setEnabled(false);

connect(showHighscores, &QPushButton::clicked, this, [&](){showTop10(difficulty);});

firstLayout->addWidget(*showHighscores*);

resetButton = new QPushButton("Reset");

resetButton->setFont(font);

connect(resetButton, &QPushButton::clicked, this, [&](){reset();} );

firstLayout->addWidget(*resetButton*);

top10 = new QMessageBox;

top10->setWindowTitle("Top 10");

QPushButton\* ok = new QPushButton("Ok");

QPushButton\* resetScores = new QPushButton("Reset highscores");

connect(resetScores, &QPushButton::pressed, this, [&](){

if(difficulty == 0){

highscoresEasy.clear();

}

else if(difficulty == 1){

highscoresMedium.clear();

}

else{

highscoresHard.clear();

}

top10->setText("");

});

top10->addButton(*resetScores*, QMessageBox::ResetRole);

top10->addButton(*ok*, QMessageBox::AcceptRole);

setButtons(8, 10, 50);

setBombs();

setNumbers();

}

void MainWindow::**setButtons**(int n, int m, int size){

buttonRows = n;

buttonColumns = m;

buttons = new MyButton\*\*[n];

buttonText = new QString\*[n];

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

buttons[i] = new MyButton\*[m];

buttonText[i] = new QString[m];

}

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

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

buttonText[i][j] = '-';

}

}

int fontsize;

switch (difficulty) {

case 0:

fontsize = 12;

numberOfBombs = 10;

numberOfFlags = 10;

break;

case 1:

fontsize = 10;

numberOfBombs = 40;

numberOfFlags = 40;

break;

case 2:

fontsize = 8;

numberOfBombs = 99;

numberOfFlags = 99;

break;

}

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

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

buttons[i][j] = new MyButton(this);

buttons[i][j]->setMinimumSize(size, size);

buttons[i][j]->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

font = buttons[i][j]->font();

font.setPointSize(font.pointSize() + fontsize);

buttons[i][j]->setFont(font);

buttons[i][j]->installEventFilter(this);

connect(buttons[i][j], SIGNAL(clicked()), this, SLOT(buttonClicked()));

connect(buttons[i][j], SIGNAL(rightClick()), this, SLOT(rightButtonClicked()));

if( (i+j) % 2 == 0 ){

buttons[i][j]->setStyleSheet("background-color:palegreen");

}

else{

buttons[i][j]->setStyleSheet("background-color:limegreen");

}

gameLayout->addWidget(*buttons[i][j]*, i, j);

}

}

}

void MainWindow::**setBombs**(){

int nB = numberOfBombs;

while(nB > 0){

QTime time = QTime::currentTime();

srand((uint)time.msec());

int x = rand() % buttonRows;

int y = rand() % buttonColumns;

if (buttonText[x][y] == 'X'){

continue;

}

buttonText[x][y] = 'X';

--nB;

}

}

void MainWindow::**setNumbers**(){

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

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

if(buttonText[i][j] == 'X'){

continue;

}

int db = 0;

for(int x=i-1; x<i+2; ++x){

for(int y=j-1; y<j+2; ++y){

if(x<0 || x>buttonRows-1 || y<0 || y>buttonColumns-1){

continue;

}

if(buttonText[x][y] == "X"){

++db;

}

}

}

if(db != 0){

buttonText[i][j] = QString::number(db);

}

}

}

}

void MainWindow::**onFreeButtonClicked**(int i, int j){

if(i<0 || i>buttonRows-1 || j<0 || j>buttonColumns-1 || buttonText[i][j] == "X" || !buttons[i][j]->isEnabled()){

return;

}

if(buttonText[i][j] == '-'){ // ha üres gomb

buttons[i][j]->setStyleSheet("background-color:azure");

buttons[i][j]->setEnabled(false);

++disabledButtons;

for(int x=i-1; x<i+2; ++x){

for(int y=j-1; y<j+2; ++y){

onFreeButtonClicked(x,y); // rekurzív hívás a gomb körül levő gombokra

}

}

}

if(buttonText[i][j] != '-'){

buttons[i][j]->setStyleSheet("background-color:wheat");

buttons[i][j]->setText(buttonText[i][j]);

buttons[i][j]->setEnabled(false);

++disabledButtons;

return;

}

}

void MainWindow::**bombClicked**(){

endOfGame = true;

timer->stop();

flags->setText("You lost!");

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

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

if(buttonText[i][j] == "X"){

buttons[i][j]->setStyleSheet("background-color:tomato");

buttons[i][j]->setText(buttonText[i][j]);

}

}

}

showHighscores->setEnabled(true);

}

void MainWindow::**win**(){ // nyerés esetén

endOfGame = true;

timer->stop();

flags->setText("You win!");

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

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

if(buttonText[i][j] == "X"){

buttons[i][j]->setStyleSheet("background-color:pink");

buttons[i][j]->setText(buttonText[i][j]);

}

else{

if(buttonText[i][j] == "-"){

buttons[i][j]->setStyleSheet("background-color:powderblue");

}

else{

buttons[i][j]->setStyleSheet("background-color:powderblue");

buttons[i][j]->setText(buttonText[i][j]);

}

}

}

}

// bekér egy nevet, amit eltárolunk (max. 10)

QString name = QInputDialog::getText(this, "Best Results", "Your Result: " +

QString::number(time) + "s\nName:");

if(difficulty == 0){

highscoresEasy[time] = name;

if(highscoresEasy.size() > 10){

auto it = highscoresEasy.end();

highscoresEasy.erase(--it);

}

}

else if(difficulty == 1){

highscoresMedium[time] = name;

if(highscoresMedium.size() > 10){

auto it = highscoresMedium.end();

highscoresMedium.erase(--it);

}

}

else{

highscoresHard[time] = name;

if(highscoresHard.size() > 10){

auto it = highscoresHard.end();

highscoresHard.erase(--it);

}

}

showTop10(difficulty);

showHighscores->setEnabled(true);

}

void MainWindow::**reset**(){

endOfGame = false;

time = 0;

timer->stop();

timeLabel->setText("0:0");

showHighscores->setEnabled(false);

disabledButtons = 0;

if(difficulty == 0){

clearButtons();

setButtons(8, 10, 50);

setBombs();

setNumbers();

flags->setText("Flags: 10");

}

else if(difficulty == 1){

clearButtons();

setButtons(14, 18, 40);

setBombs();

setNumbers();

flags->setText("Flags: 40");

}

else{

clearButtons();

setButtons(20, 24, 30);

setBombs();

setNumbers();

flags->setText("Flags: 99");

}

}

void MainWindow::**clearButtons**(){

while(gameLayout->*count*()){

for(int i=0; i<gameLayout->*count*(); ++i){

QWidget \*tempWidget = gameLayout->*itemAt*(i++)->*widget*();

mainLayout->removeWidget(*tempWidget*);

delete tempWidget;

}

}

delete buttons;

delete buttonText;

}

void MainWindow::**buttonClicked**(){

if(endOfGame){

return;

}

if(!timer->isActive()){

timer->start(80);

}

int x, y;

QPushButton\* tmp = (QPushButton\*)sender();

if(tmp->styleSheet() == "background-color:brown"){

return;

}

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

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

if(buttons[i][j] == tmp){

x = i;

y = j;

break;

}

}

}

if(buttonText[x][y] != "X"){

onFreeButtonClicked(x,y);

if( disabledButtons == ((buttonRows\*buttonColumns) - numberOfBombs) ){

win();

}

}

else{

bombClicked();

}

}

void MainWindow::**rightButtonClicked**(){

if(!timer->isActive()){

timer->start(80);

}

QPushButton\* tmp = (QPushButton\*)sender();

int x, y;

if(tmp->styleSheet() == "background-color:brown"){

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

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

if(buttons[i][j] == tmp){

x = i;

y = j;

break;

}

}

}

if( (x+y) % 2 == 0 ){

buttons[x][y]->setStyleSheet("background-color:palegreen");

}

else{

buttons[x][y]->setStyleSheet("background-color:limegreen");

}

++numberOfFlags;

flags->setText("Flags: " + QString::number(numberOfFlags));

}

else{ // ha nincs flag rajta, beállítjuk

tmp->setStyleSheet("background-color:brown");

--numberOfFlags;

flags->setText("Flags: " + QString::number(numberOfFlags));

}

}

void MainWindow::**showTop10**(int difficulty){

if(difficulty == 0){

QMapIterator<double, QString> it(highscoresEasy);

top10->setText("Level: Easy\n\n");

for(int i=1; i<11; ++i){

if(it.hasNext()){

it.next();

top10->setText(top10->text() + QString::number(i) + ". " + it.value() + ": " + QString::number(it.key()) + "s\n");

}

else{

break;

}

}

}

else if(difficulty == 1){

QMapIterator<double, QString> it(highscoresMedium);

top10->setText("Level: Medium\n\n");

for(int i=1; i<11; ++i){

if(it.hasNext()){

it.next();

top10->setText(top10->text() + QString::number(i) + ". " + it.value() + ": " + QString::number(it.key()) + "s\n");

}

else{

break;

}

}

}

else{

QMapIterator<double, QString> it(highscoresHard);

top10->setText("Level: Hard\n\n");

for(int i=1; i<11; ++i){

if(it.hasNext()){

it.next();

top10->setText(top10->text() + QString::number(i) + ". " + it.value() + ": " + QString::number(it.key()) + "s\n");

}

else{

break;

}

}

}

top10->*exec*();

}

QComboBox\* MainWindow::**createComboBox**(){

temp = new QComboBox;

temp->setStyleSheet("background-color:aliceblue");

font = temp->font();

font.setPointSize(font.pointSize() + 8);

temp->setFont(font);

temp->addItem(“Easy”);

temp->addItem(“Medium”);

temp->addItem(“Hard”);

connect(temp, SIGNAL(currentIndexChanged(int)), this, SLOT(difficultyChanged()));

return temp;

}

void MainWindow::**tick**(){

time += 0.1;

timeLabel->setText(QString::number(time));

}

void MainWindow::**difficultyChanged**(){

difficulty = difficulties->currentIndex();

reset();

}

MainWindow::~***MainWindow***()

{

}

**16. MyHeader.cpp**

#include "MyHeader.h"

MyButton::MyButton(QWidget \*parent): QPushButton(parent)

{

}

MyButton::~MyButton(){

}

void MyButton::mousePressEvent(QMouseEvent \*e){

if(e->button() == Qt::RightButton){

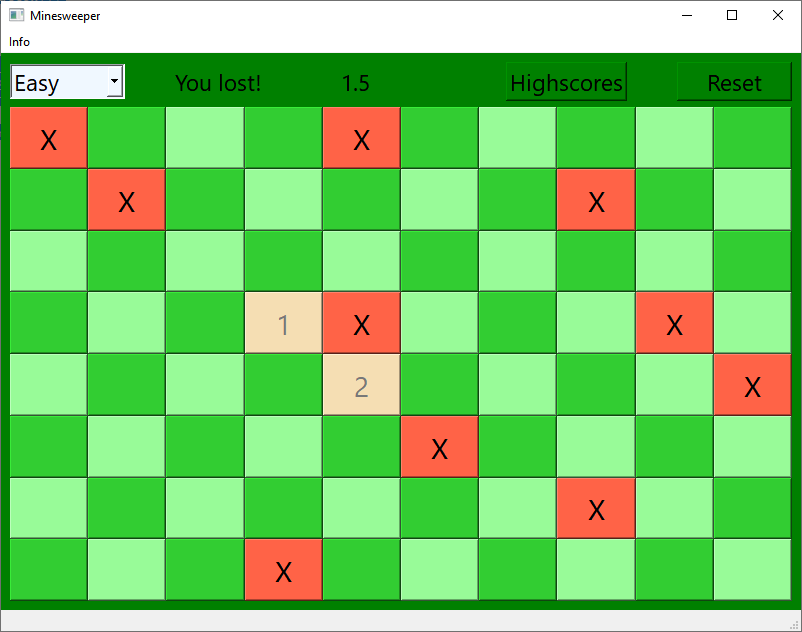
emit rightClick();

}

QPushButton::mousePressEvent(e);

}

**Результаты тестирования программы:**

****

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