Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Программирование

Отчет по лабораторной работе
Игра Сапер

Работу выполнил:
Осипов А.Ю.
Группа: 13501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2016

Содержание

1 Игра Сапер

Всем нам иногда бывает скучно. Требуется убить 5-10 минут до выхода или надоело ждать прихода гостей? В таких случаях можно поиграть в развиваючую логическую игру «Сапер». Игроку предстоит ощутить себя на минном поле, последовательно открывая ячейки и расчищть его шаг за шагом, по пути обозначая обнаруженные мины флагами.

1.1 Задание

Реализовать приложение для игры в «Сапер» Описание игры:

——— Цель игры ———

Найти пустые ячейки, но не трогать ячейки, содержащие мины. Чем быстрее вы очистите всю доску, тем лучше будет результат. ——— Игровая доска ———

В игре «Сапер» есть три стандартные доски, каждая следующая сложнее предыдущей.

Новичок: 81 ячейка, 10 мин

Любитель: 256 ячеек, 40 мин

Профессионал: 480 ячеек, 99 мин

——— Правила игры ———

Если открыта ячейка с миной, игра проиграна.

Если открыта пустая ячейка, игра продолжается.

Если в ячейке указано число, оно показывает, сколько мин скрыто в восьми ячейках вокруг данной. Это число помогает понять, где находятся безопасные ячейки.

1.2 Концепция

Готовым проектом является графическое приложение, реализующее взаимодействия с ядром игры. Игроку отображается стартовое меню с возможностью выбрать один из трех уровней сложности игры (новичок, любитель и профессионал). После старта игры отображается игровое поле, счетчик свободных клеток и оставшихся мин.

1.3 Минимально работоспособный продукт

Консольное приложение реализующее взаимодействие с ядром игры. Возможные действия:

- Отобразить информацию о поле (какие ячейки открыты, какие - нет, и сколько мин находятся рядом с выбранной ячейкой)
- Вскрыть ячейку или установить на ней флаг

Также это приложение должно определять конец игры и информировать об этом игрока.

1.4 Диаграмма прецедентов использования

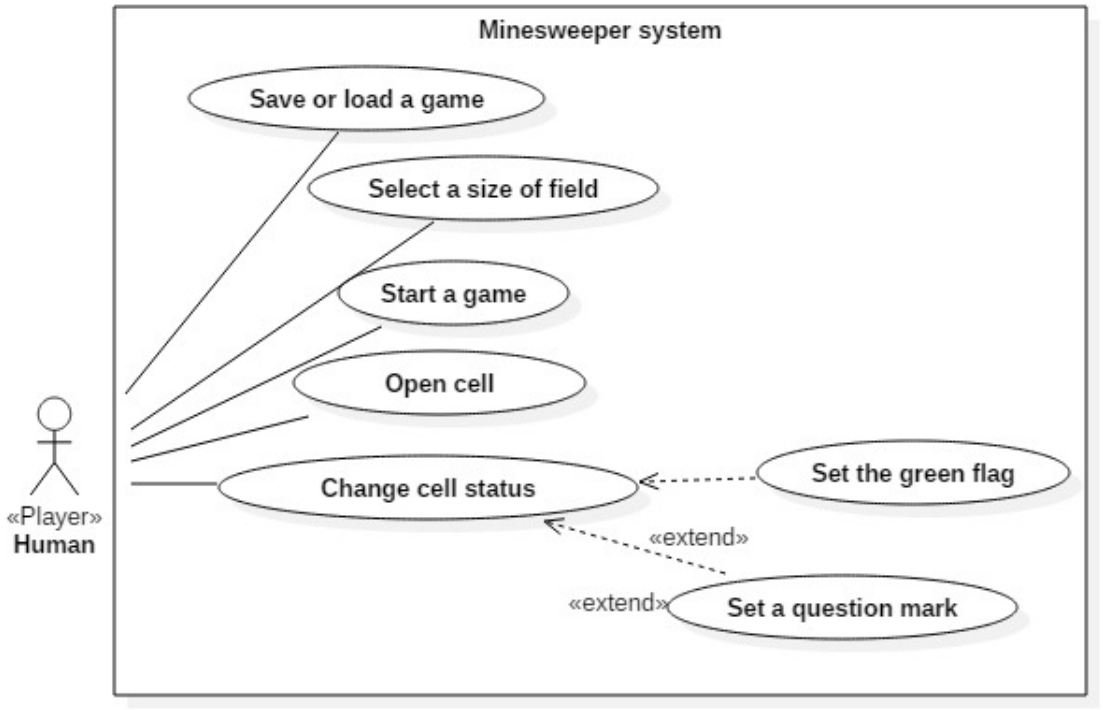

Рис. 1: Диаграмма прецедентов использования

1.5 Диаграмма последовательностей

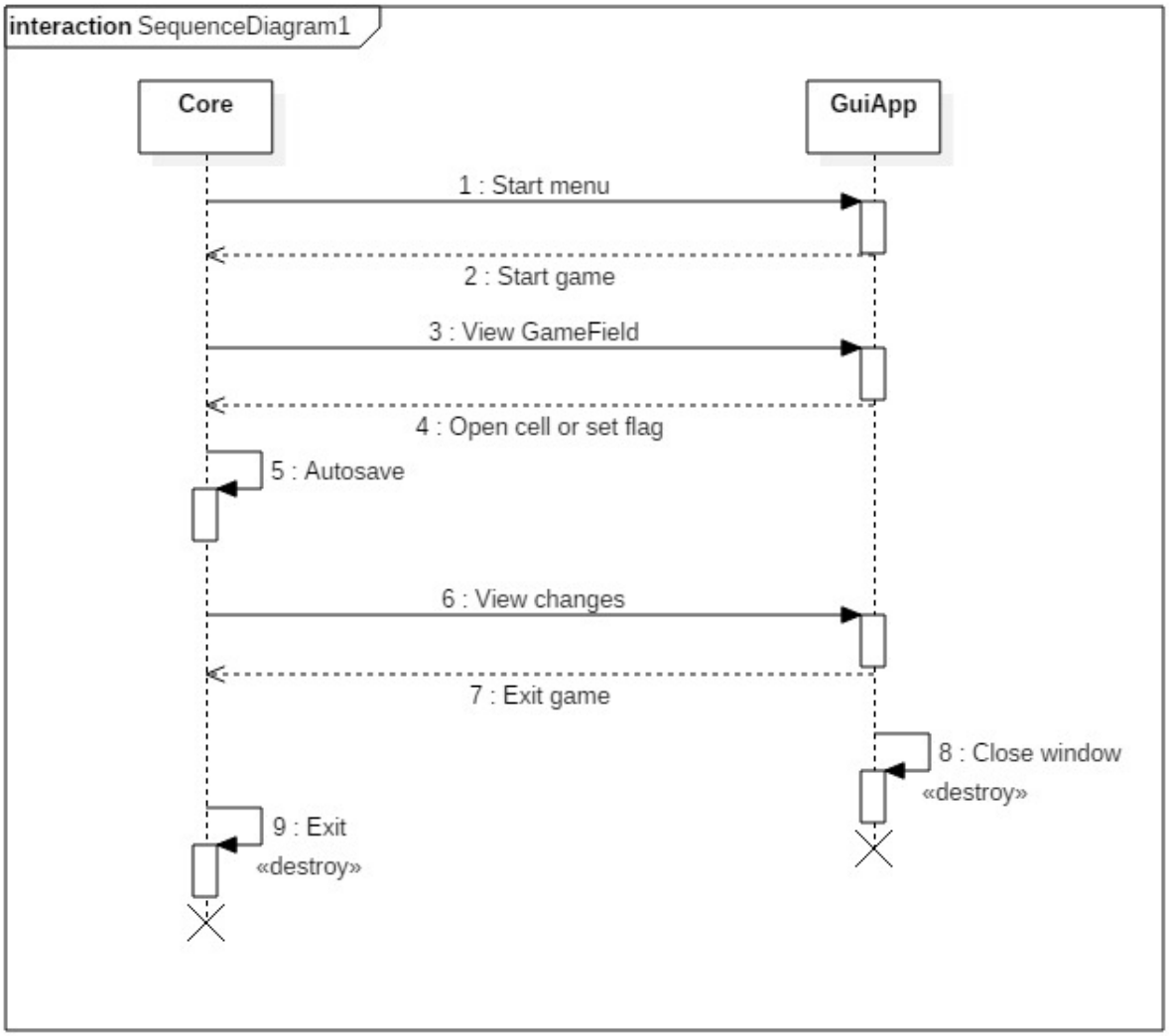

Рис. 2: Диаграмма последовательностей

2 Проектирование приложения

2.1 Архитектуру приложения

Было решено выделить 4 подпроекта:

1. Консольное приложение - промежуточное приложение, целью которого является минимально продемонстрировать возможности ядра
2. Библиотека - подпроект, содержащий основную бизнес-логику всего проекта
3. Графическое приложение - итоговое приложение для использования
4. Тесты - подпроект, созданный для того, чтобы тестировать библиотеку, содержащую основную бизнес-логику

2.2 Диаграмма компонентов

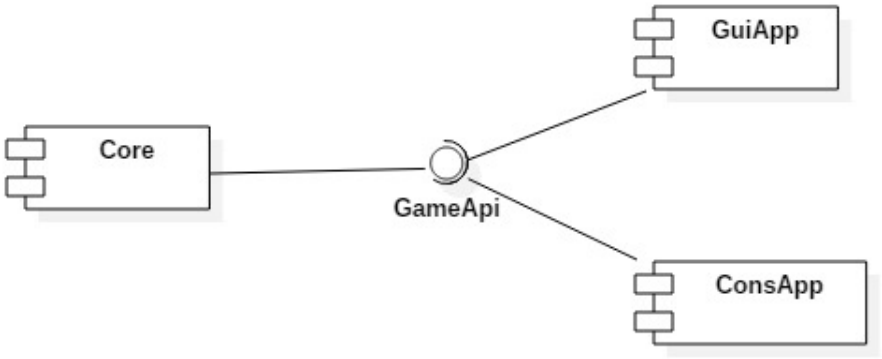

Рис. 3: Диаграмма компонентов

2.3 Файлы создаваемые в процессе работы приложения

Отсутствуют

2.4 Интерфейс ядра

В библиотека предоставляет следующую функциональность:

1. `int getSizeX() const;`
2. `int getSizeY() const;`

   Два метода с помощью которых можно узнать размеры поля по горизонтали и вертикали соответственно
3. `int getMinesNumber() const;`

   Узнать количество мин на поле
4. `Cell getPieceOfField(int x, int y);`

   С помощью это метода можно получить доступ к конкретной клетке поля
5. `void open(int x, int y);`

   Открыть ячейку по заданным координатам
6. `void setFlag(int x, int y);`

   Установить флаг по заданным координатам
7. `void openAllCells();`

   Открыть все ячейки поля
8. `bool isLose();`

   Возвращает 1 если игра проиграна
9. `bool isGameActive();`

   Возвращает 1 если игра не остановлена

3 Реализация Проекта

3.1 Используемые версии

- Qt Creator 4.0.0 (Qt 5.6.0)
- Стандарт c++11
- GCC 5.4.2 и GCC 5.5.1
- Операционная система: Windows 7
- cppcheckgui версии 1.7.2

3.2 Основные классы

В ядре программы всего 2 класса:

- Cell - класс, отвечающий за взаимодействие с каждой клеткой поля
- Field - класс, представляющий игровое поле

3.3 Скриншоты основных экранов пользовательского интерфейса

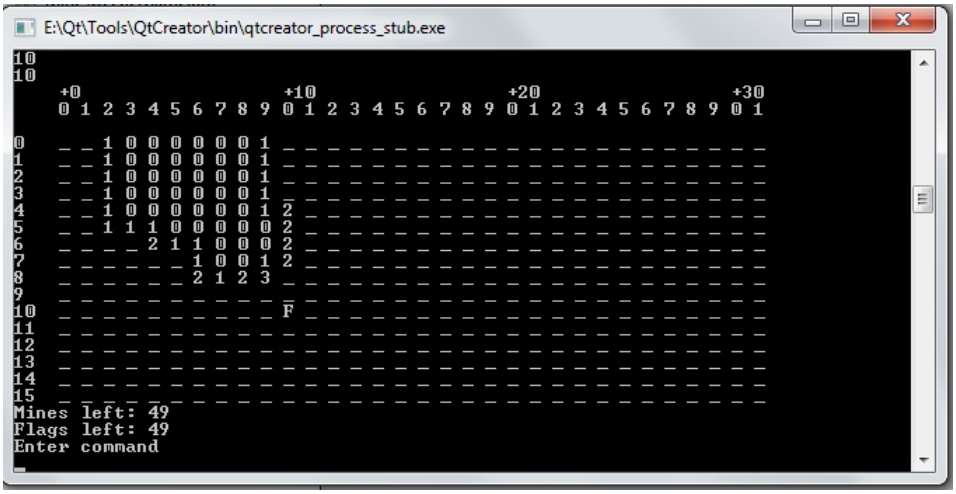

Рис. 4: Скрин консольного приложения

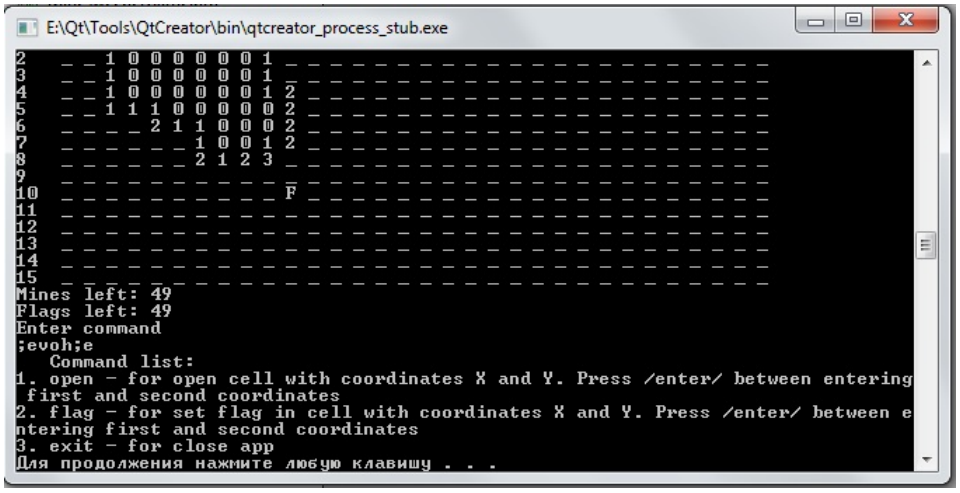

Рис. 5: Скрин консольного приложения 2

В консольном приложении пользователь имеет возможность:

- Открыть ячейку (open)
- Установить флаг (flag)
- Получить помощи по командам (help)
- Выйти (exit)

Рис. 6: Главное окно графического приложения

Стартовое окошко, в котором пользователь имеет возможность начать новую игру, выйти или изменить опции игры.

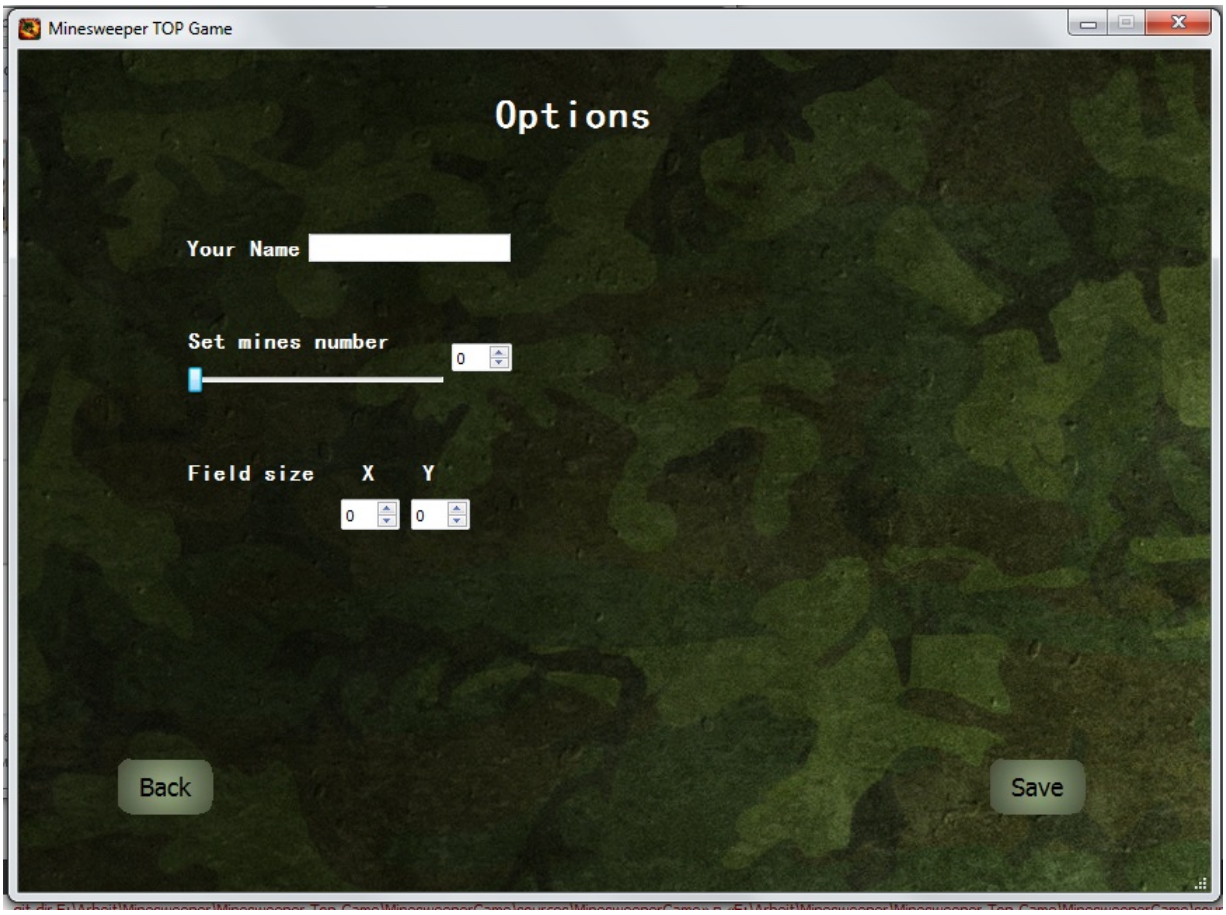

Рис. 7: Окно опций

В этом окне можно изменить размеры поля для игры, количество мин на нем.

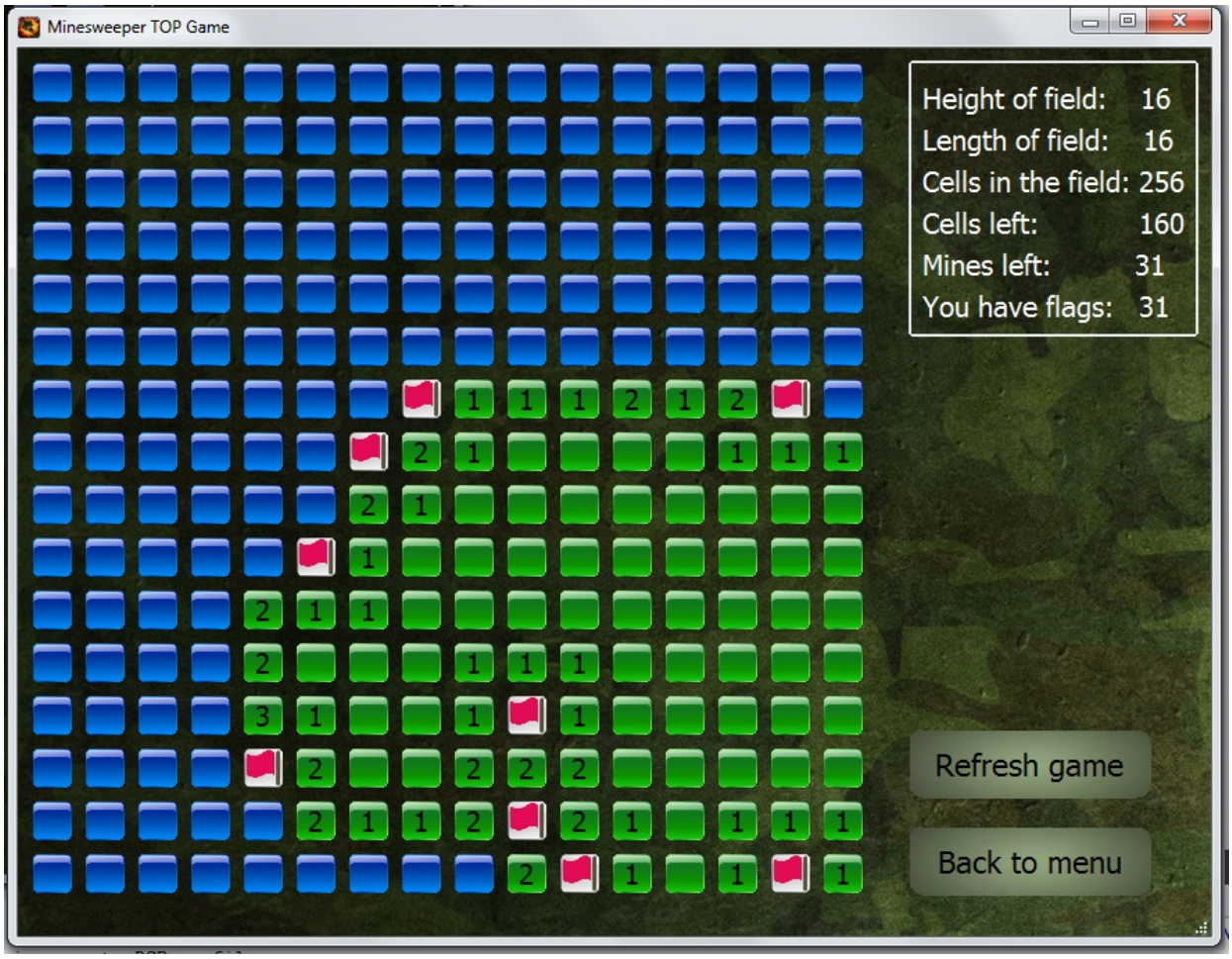

Рис. 8: Окно опций

Окно непосредственно самой игры. Здесь пользователь может открывать ячейки, ставить флаги. Справа выводится информация о текущем состоянии поля: его размеры, количатво неотмеченных мин, количество флагов, которые остались у игрока.

4 Процесс обеспечения качества и тестирование

4.1 О ревью

Было получено в сумме 5 ревью:

- Архитектурное - 11 замечаний
- Ревью по коду от Михаила - 34 замечания

  Из них 3 после проведенных ДЕМО:

- После первого - 3 замечания, в основном, стилевого характера
- После второго - 4 незначительных замечания, которые были запомнены и исправлены (но pull requests до сих пор не пришел)
- После третьего - опять же незначительные замечания вроде "добавить обводку для текста"или "изменить название окна"

  Почти все исправления были сделаны вовремя.

4.2 Список использованных утилит

Были использованы:

- cppcheck

4.3 Автоматические тесты

В ходе разработки были написаны автоматические тесты, которые сильно упростили процесс разработки оной. Во-первых, они немного изменили мое представление о том, как должны выглядеть методы, чтобы их можно было протестировать.
Тесты реализованы так, что они по очереди вызывают методы ядра с заданными параметрами или без них и сравнивают результат с ожидаемым. Со своей задачей справились отлично.

5 Выводы

Семестр очень понравился. Было много нового, вполне интересно. Трудности вызывала только подготовка к семинарам, сложно читаемые книги. Больше всего понравился формат курсовой работы, то есть какой-то целый проект, когда можно приложить руки к созданию каждой его части, довести от начала до конца, добавить авторские штучки и.т.д
В начале года (учебного) меня спросили, чего я ожидаю от первого курса обучения. Я тогда ответил, что хочу к концу года создать проект, который можно будет использовать в практических целях (а не просто втупую писать бесполезные куски кода), на практике решая всплывающие проблемы. В целом, все мои ожидания оправдались, чему я очень рад. Желаю в следующем семестре продолжить обучение по такой же схеме.

В ходе работы над проектом мной был получен ценный опыт, кроме того, я:

- Наконец-то освоил автоматическое тестирование средствами Qt
- Осознал значимость ревью, понял что важно как просить, так и самому делать его другим
- Научился создавать проект с несколькими компонентами и обозначать их взаимодействие
- Освоил новые средства Github
- Сделал свой первый графический интерфейс
- Осознал необходимость наследования классов, для эффективной разработки проекта
- Опробовал практику ревью в коллективе
- Поучаствовал в семинарах по улучшению работоспособности кода и приложения в целом

6 Листинги

```
1  #ifndef FIELD_H
2  #define FIELD_H
3
4  #include "cell.h"
5  #include <stdlib.h>
6  #include <time.h>
7  #include <vector>
8  #include <iostream>
9  #include "str_switch.h"
10
11 using namespace std;
12
13 /**
14  * @brief The Field class Класс целого игрового поля, по сути это интерфейс ядра
15  */
16 class Field
17 {
18 public:
19     Field();
20     /**
21      * @brief Field Конструктор поля
22      * @param sizeX Заданный размер X поля
```

```
23          * @param sizeY Заданный размер Y поля
24          * @param minesNumber Желаемое количество мин
25          */
26         Field(int sizeX, int sizeY, int minesNumber);
27 
28         /**
29          * @brief openAllCells Открывает все ячейки поля
30          */
31         void openAllCells();
32 
33         /**
34          * @brief checkValidCoord Проверяет верность координат
35          * @param coord1 Первая координата
36          * @param coord2 Вторая координата
37          * @return 1 - если координаты валидные, 0 - если нет
38          */
39         bool checkValidCoord(int coord1, int coord2) const;
40 
41         /**
42          * @brief random Генерирует случайное число до максимального
43          * @param maxValue Максимальное число
44          * @return Сгенерированное число
45          */
46         int random(int maxValue);
47 
48         /**
49          * @brief getSizeX Возвращает размер X поля
50          * @return Длину поля
51          */
52         int getSizeX() const;
53 
54         /**
55          * @brief getSizeY Возвращает размер Y поля
56          * @return Ширину поля
57          */
58         int getSizeY() const;
59 
60         /**
61          * @brief getMinesNumber Возвращает количество мин на поле
62          * @return Количество мин всего на поле
63          */
64         int getMinesNumber() const;
65 
66         /**
67          * @brief getPieceOfField Интерфейс для доступа к конкретной ячейе поля по заданным координатам
68          * @param x Первая координата
69          * @param y Вторая координата
70          * @return Объект Cell - конкретную ячейку поля
71          */
72         Cell getPieceOfField(int x, int y);
73 
74         /**
75          * @brief openCellsAround Открывает ячейку и все пустые ячейки в радиусе 1 клетка
76          * @param x Первая координата
77          * @param y Вторая координата
78          */
79         void open(int x, int y);
80 
81         /**
82          * @brief isLose Проверяет, проиграна ли игра
83          * @return 1 - если проиграна, 0 - если нет
84          */
85         bool isLose();
86 
87         /**
88          * @brief isGameActive Проверяет, продолжается ли игра
89          * @return 1 - если продолжается, 0 - если нет
90          */
91         bool isGameActive();
92 
93         /**
94          * @brief setFlag Устанавливает флаг по заданным координатам
95          * @param x Первая координата
96          * @param y Вторая координата
97          */
98         void setFlag(int x, int y);
```

```
99      bool gameActive;
100
101     Cell* getCell(int x, int y);
102
103     int minesLeft;
104     int cellsLeft;
105     int flagsLeft;
106
107 protected:
108     /**
109      * @brief setValuesInCells Устанавливает значение в клетке по заданным координатам равное
        ↪ количеству мин вокруг в радиусе 1 клетка
110      * @param x Первая координата
111      * @param y Вторая координата
112      */
113     void setValuesInCells(int x, int y);
114
115     /**
116      * @brief spawnMines Рандомно спавнит мины на поле
117      * @return Количество заспавненных мин
118      */
119     int spawnMines();
120
121     /**
122      * @brief consField Собственно, само игровое поле, вектор из объектов типа "ячейка"
123      */
124     vector<vector<Cell> > consField;
125
126     /**
127      * @brief fieldSizeX размер X поля
128      */
129     int fieldSizeX;
130
131     /**
132      * @brief fieldSizeY Размер Y поля
133      */
134     int fieldSizeY;
135
136     /**
137      * @brief minesNumber Количество мин на поле
138      */
139     int minesNumber;
140
141     /**
142      * @brief loseFlag Флаг, отвечающий за поражение в игре
143      */
144     bool loseFlag;
145
146
147     void calculateMinesLeft();
148     void calculateCellsLeft();
149     void calculateFlagsLeft();
150
151
152 };
153
154 #endif // FIELD_H
```

```
1  #include "field.h"
2
3  Field::Field()
4  {
5      fieldSizeX = 16;
6      fieldSizeY = 16;
7      minesNumber = 20;
8      gameActive = 1;
9      loseFlag = 0;
10
11     minesLeft = minesNumber;
12     flagsLeft = minesNumber;
13     cellsLeft = fieldSizeX * fieldSizeY;
14
15         vector <Cell> tmpVect;
16         for (int i = 0; i < fieldSizeX; i++)
17         {
18             for (int j = 0; j < fieldSizeY; j++)
```

```
19                  tmpVect.push_back(Cell(i, j));
20              consField.push_back(tmpVect);
21          }
22
23              srand(static_cast<unsigned int>(time(NULL)));
24              spawnMines();
25
26              for (int i = 0; i < fieldSizeX; i++)
27                  for (int j = 0; j < fieldSizeY; j++)
28                      setValuesInCells(i, j);
29  }
30
31  Field::Field(int sizeX, int sizeY, int minesNumber)
32  {
33      Field::minesNumber = minesNumber;
34      gameActive = 1;
35      loseFlag = 0;
36         fieldSizeX = sizeX;
37         fieldSizeY = sizeY;
38
39         minesLeft = minesNumber;
40         flagsLeft = minesNumber;
41         cellsLeft = fieldSizeX * fieldSizeY;
42
43
44          vector<Cell> tmpVect;
45          for (int i = 0; i < fieldSizeX; i++)
46          {
47              for (int j = 0; j < fieldSizeY; j++)
48                  tmpVect.push_back(Cell(i, j));
49              consField.push_back(tmpVect);
50          }
51
52              srand(static_cast<unsigned int>(time(NULL)));
53
54              spawnMines();
55
56              for (int i = 0; i < fieldSizeX; i++)
57                  for (int j = 0; j < fieldSizeY; j++)
58                      setValuesInCells(i, j);
59  }
60
61  bool Field::checkValidCoord(int coord1, int coord2) const
62  {
63      return (coord1 >= 0) && (coord2 >= 0) && (coord1 <  fieldSizeX) && (coord2 <  fieldSizeY);
64  }
65  int Field::spawnMines()
66  {
67      int numberOfSpawnedMines = 0;
68      int x,y;
69      while (numberOfSpawnedMines < minesNumber)
70      {
71          x = random(fieldSizeX);
72          y = random(fieldSizeY);
73          Cell *cell = &consField[x][y];
74          if (cell->isMine() == 0)
75          {
76              cell->setMine();
77              numberOfSpawnedMines++;
78          }
79
80      }
81  return numberOfSpawnedMines;
82  }
83  int Field::random(int maxValue)
84  {
85      return ((rand() % maxValue));
86
87  }
88  void Field::setValuesInCells(int x, int y)
89  {
90      if (checkValidCoord(x - 1, y - 1) && consField[x-1][y-1].isMine())
91          consField[x][y].incValue();
92      if (checkValidCoord(x - 1, y) && consField[x-1][y].isMine())
93          consField[x][y].incValue();
94      if (checkValidCoord(x - 1, y + 1) && consField[x-1][y+1].isMine())
```

```
95          consField[x][y].incValue();
96      if (checkValidCoord(x, y - 1) && consField[x][y-1].isMine())
97          consField[x][y].incValue();
98      if (checkValidCoord(x, y + 1) && consField[x][y+1].isMine())
99          consField[x][y].incValue();
100     if (checkValidCoord(x + 1, y - 1) && consField[x+1][y-1].isMine())
101         consField[x][y].incValue();
102     if (checkValidCoord(x + 1, y) && consField[x+1][y].isMine())
103         consField[x][y].incValue();
104     if (checkValidCoord(x + 1, y + 1) && consField[x+1][y+1].isMine())
105         consField[x][y].incValue();
106 }
107 void Field::openAllCells()
108 {
109     for (int i = 0; i < fieldSizeX; i++)
110         for (int j = 0; j < fieldSizeY; j++)
111             consField[i][j].openCell();
112 }
113 int Field::getSizeX() const
114 {
115     return fieldSizeX;
116 }
117 int Field::getSizeY() const
118 {
119     return fieldSizeY;
120 }
121 int Field::getMinesNumber() const
122 {
123     return minesNumber;
124 }
125 void Field::open(int x, int y)
126 {
127     if (checkValidCoord(x, y))
128     {
129     consField[x][y].openCell();
130     if (consField[x][y].isMine())
131         loseFlag = 1;
132     if (consField[x][y].getValue() == 0)
133     {
134         if (checkValidCoord(x - 1, y - 1))
135         {
136             if (consField[x-1][y-1].getValue() == 0 && consField[x-1][y-1].isOpen() == 0)
137                 open(x - 1, y - 1);
138             else
139                 consField[x-1][y-1].openCell();
140         }
141         if (checkValidCoord(x - 1, y))
142         {
143             if (consField[x-1][y].getValue() == 0 && consField[x-1][y].isOpen() == 0)
144                 open(x - 1, y);
145             else  consField[x-1][y].openCell();
146         }
147
148         if (checkValidCoord(x - 1, y + 1))
149         {
150             if (consField[x-1][y+1].getValue() == 0 && consField[x-1][y+1].isOpen() == 0)
151                 open(x - 1, y + 1);
152             else
153                 consField[x-1][y+1].openCell();
154         }
155
156         if (checkValidCoord(x, y - 1))
157         {
158             if (consField[x][y-1].getValue() == 0 && consField[x][y-1].isOpen() == 0)
159                 open(x, y - 1);
160             else   consField[x][y-1].openCell();
161         }
162
163         if (checkValidCoord(x, y + 1))
164         {
165             if (consField[x][y+1].getValue() == 0 && consField[x][y+1].isOpen() == 0)
166                 open(x, y + 1);
167             else   consField[x][y+1].openCell();
168         }
169
170         if (checkValidCoord(x + 1, y - 1))
```

```
171         {
172             if (consField[x+1][y-1].getValue() == 0 && consField[x+1][y-1].isOpen() == 0)
173                 open(x + 1, y - 1);
174             else    consField[x+1][y-1].openCell();
175         }
176 
177         if (checkValidCoord(x + 1, y))
178         {
179             if (consField[x+1][y].getValue() == 0 && consField[x+1][y].isOpen() == 0)
180                 open(x + 1, y);
181             else    consField[x+1][y].openCell();
182         }
183 
184         if (checkValidCoord(x + 1, y + 1))
185         {
186             if (consField[x+1][y+1].getValue() == 0 && consField[x+1][y+1].isOpen() == 0)
187                 open(x + 1, y + 1);
188             else    consField[x+1][y+1].openCell();
189         }
190     }
191 }
192     calculateMinesLeft();
193     calculateCellsLeft();
194     calculateFlagsLeft();
195 }
196 bool Field::isLose()
197 {
198     return loseFlag;
199 }
200 bool Field::isGameActive()
201 {
202     return gameActive;
203 }
204 Cell Field::getPieceOfField(int x, int y)
205 {
206     return consField[x][y];
207 }
208 void Field::setFlag(int x, int y)
209 {
210     if (flagsLeft > 0 || consField[x][y].isFlag())
211     {
212     consField[x][y].swapFlag();
213     calculateMinesLeft();
214     calculateCellsLeft();
215     calculateFlagsLeft();
216     }
217 }
218 Cell* Field::getCell(int x, int y)
219 {
220    Cell *cell = &consField[x][y];
221    return cell;
222 }
223 void Field::calculateCellsLeft()
224 {
225     int closedCells = 0;
226     for (int i = 0; i < fieldSizeX; i++)
227         for (int j = 0; j < fieldSizeY; j++)
228             if (consField[i][j].isOpen() == 0)
229                 closedCells++;
230     cellsLeft = closedCells;
231 }
232 void Field::calculateFlagsLeft()
233 {
234     int flagsSetted = 0;
235     for (int i = 0; i < fieldSizeX; i++)
236         for (int j = 0; j < fieldSizeY; j++)
237             if (consField[i][j].isFlag())
238                 flagsSetted++;
239     flagsLeft = minesNumber - flagsSetted;
240 }
241 void Field::calculateMinesLeft()
242 {
243     int minesDetected = 0;
244     for (int i = 0; i < fieldSizeX; i++)
245         for (int j = 0; j < fieldSizeY; j++)
246             if (consField[i][j].isMine() && consField[i][j].isFlag())
```

```
                minesDetected++;
    minesLeft = minesNumber - minesDetected;
}
```

//

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "optionswindow.h"
#include "gamewindow.h"
#include "choicelevelwindow.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

public slots:

private slots:
    void goToOptions();
    void goToStartGame();

signals:

private:
    Ui::MainWindow *ui;
    OptionsWindow *optionsWnd;
    GameWindow *gameWnd;


    const QSize SCREEN_SIZE{850, 600};

    QString QPushButtonStyle =
                "QPushButton "
                "{"
                    " border: 1px solid #000000;"
                    " border-image: url(:/resources/images/button.jpg);"
                    " padding: 7.5px 15px;"
                    " border-radius: 10px;"
                    " color: #000000;"
                    " font-size: 22px;"
                "}"
                "QPushButton:flat "
                "{"
                    " border: none;"
                "}"
                "QPushButton:hover "
                "{"
                    " border-image: url(:/resources/images/button_hover);"
                "}"
                "QPushButton:pressed "
                "{"
                    " border-image: url(:/resources/images/button_background.jpg);"
                    " border-color #00ffff;"
                "}";


};

#endif // MAINWINDOW_H
```

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
```

```
6       ui(new Ui::MainWindow)
7   {
8       ui->setupUi(this);
9
10      QPixmap background(":/resources/images/background.png");
11          QPalette qPalette;
12          qPalette.setBrush(this->backgroundRole(),QBrush(background));
13          this->setPalette(qPalette);
14
15          this->setFixedSize(SCREEN_SIZE);
16          this->setWindowTitle("Minesweeper TOP Game");
17          this->setWindowIcon(QIcon(":/resources/images/icon.png"));
18
19          ui->exitButton->setStyleSheet(QPushButtonStyle);
20          ui->startGameButton->setStyleSheet(QPushButtonStyle);
21          ui->optionsButton->setStyleSheet(QPushButtonStyle);
22
23          connect(ui->optionsButton, SIGNAL(clicked(bool)),this, SLOT(goToOptions()));
24          connect(ui->startGameButton, SIGNAL(clicked(bool)), this, SLOT(goToStartGame()));
25          connect(ui->exitButton, SIGNAL(clicked(bool)), this, SLOT(close()));
26  }
27
28  MainWindow::~MainWindow()
29  {
30      delete ui;
31  }
32
33  void MainWindow::goToOptions()
34  {
35      optionsWnd = new OptionsWindow;
36      optionsWnd->show();
37      this->close();
38  }
39
40  void MainWindow::goToStartGame()
41  {
42     ChoiceLevelWindow *window = new ChoiceLevelWindow;
43     window->show();
44     this->close();
45  }
```

//

```
1   #ifndef GAMEWINDOW_H
2   #define GAMEWINDOW_H
3
4   #include <QMainWindow>
5   #include "ui_gamewindow.h"
6   #include "field.h"
7   #include "qmypushbutton.h"
8   #include <QGridLayout>
9   #include <QPushButton>
10  #include <vector>
11  #include <QMouseEvent>
12  #include <QGraphicsScene>
13  #include <QGraphicsView>
14  #include <QPixmap>
15  #include <QLabel>
16  #include <QIcon>
17  #include <QSize>
18  #include <QMouseEvent>
19  #include <QIcon>
20  #include <QMessageBox>
21  #include <QGroupBox>
22  #include <QTimer>
23
24  namespace Ui {
25  class GameWindow;
26  }
27
28  class GameWindow : public QMainWindow
29  {
30      Q_OBJECT
31
32  public:
33      explicit GameWindow(int sizeX, int sizeY, int minesNumber);
34      ~GameWindow();
```

```
35      void repaint();
36
37  signals:
38      void allah_BABAH();
39      void allCellsOpen();
40
41  private:
42      vector<vector<QMyPushButton*> > buttons;
43      Ui::GameWindow *ui;
44      Field *core;
45      QPushButton *backButton;
46      QPushButton *refreshButton;
47      QGridLayout *fieldLayout;
48      QVBoxLayout *mainFieldLayout;
49      QVBoxLayout *panelLayout;
50      QHBoxLayout *mainLayout;
51      int i,j;
52      int sizeOfFieldX;
53      int sizeOfFieldY;
54      int minesInTheField;
55      QSize *fieldButtonSize;
56      QLabel *fieldHeightNumber;
57      QLabel *fieldLengthNumber;
58      QLabel *allCellsNumber;
59      QLabel *cellsLeftNumber;
60      QLabel *minesLeftNumber;
61      QLabel *flagsLeftNumber;
62      QGroupBox *panelBox;
63      QVBoxLayout *mainPanelLayout;
64      QTimer *timer;
65
66
67      QString QPushButtonStyle =
68                  "QPushButton "
69                  "{"
70                      " border: 1px solid #000000;"
71                      " border-image: url(:/resources/images/button.jpg);"
72                      " padding: 7.5px 15px;"
73                      " border-radius: 10px;"
74                      " color: #000000;"
75                      " font-size: 22px;"
76                  "}"
77                  "QPushButton:flat "
78                  "{"
79                      " border: none;"
80                  "}"
81                  "QPushButton:hover "
82                  "{"
83                      " border-image: url(:/resources/images/button_hover);"
84                  "}"
85                  "QPushButton:pressed "
86                  "{"
87                      " border-image: url(:/resources/images/button_background.jpg);"
88                      " border-color #00ffff;"
89                  "}";
90
91      QString QFieldButtonStyle =
92              "QPushButton "
93              "{"
94                  " border-image: url(:/resources/images/field_button2.png);"
95              "}"
96              "QPushButton:hover "
97              "{"
98                  " border-image: url(:/resources/images/field_button2_pressed.png);"
99              "}";
100
101     QString MinesButtonStyle =
102                 "QPushButton "
103                 "{"
104                     " border-image: url(:/resources/images/field_button5.png);"
105                 "}";
106     QString FlagButtonStyle =
107                 "QPushButton "
108                 "{"
109                      " border-image: url(:/resources/images/field_button1.png);"
110                 "}";
```

```
111     QString OpenButtonStyle =
112             "QPushButton "
113             "{"
114                 " border-image: url(:/resources/images/field_button3.png);"
115                 " font-size: 20px;"
116             "}";
117     QString QLabelStyle =
118             "QLabel "
119             "{"
120                 " color: #ffffff;"
121                 " font-size: 20px;"
122             "}";
123
124 private slots:
125     void clickedLeft();
126     void backToMenu();
127     void setFlag();
128     void winGame();
129     void loseGame();
130     void refreshGame();
131 };
132
133 #endif // GAMEWINDOW_H
```

```
1  #include "gamewindow.h"
2  #include "ui_gamewindow.h"
3  #include "mainwindow.h"
4
5
6  GameWindow::GameWindow(int sizeX, int sizeY, int minesNumber) :
7      QMainWindow(),
8      ui(new Ui::GameWindow)
9  {
10     ui->setupUi(this);
11
12     QPixmap background(":/resources/images/options_background.jpg");
13         QPalette qPalette;
14         qPalette.setBrush(this->backgroundRole(),QBrush(background));
15         this->setPalette(qPalette);
16
17     sizeOfFieldX = sizeX;
18     sizeOfFieldY = sizeY;
19     minesInTheField = minesNumber;
20
21     core = new Field(sizeOfFieldX, sizeOfFieldY, minesInTheField);
22     fieldLayout = new QGridLayout;
23
24     backButton = new QPushButton("Back to menu");
25     backButton->setStyleSheet(QPushButtonStyle);
26     backButton->setFixedSize(175,50);
27
28     refreshButton = new QPushButton("Refresh game");
29     refreshButton->setStyleSheet(QPushButtonStyle);
30     refreshButton->setFixedSize(175,50);
31
32     QObject::connect(backButton, SIGNAL(clicked(bool)), this, SLOT(backToMenu()));
33     QObject::connect(refreshButton, SIGNAL(clicked(bool)), this, SLOT(refreshGame()));
34
35     mainFieldLayout = new QVBoxLayout;
36     panelLayout = new QVBoxLayout;
37     mainLayout = new QHBoxLayout;
38
39     vector<QMyPushButton*> tmpVect;
40     for (int i = 0; i < core->getSizeX(); i++)
41     {
42         for (int j = 0; j < core->getSizeY(); j++)
43         {
44             QMyPushButton* newButton;
45             tmpVect.push_back(newButton);
46         }
47         buttons.push_back(tmpVect);
48     }
49
50     QLabel *fieldHeight = new QLabel("Height of field:   ");
51     fieldHeight->setStyleSheet(QLabelStyle);
52     QLabel *fieldLength = new QLabel("Length of field:   ");
```

```
53      fieldLength->setStyleSheet(QLabelStyle);
54      QLabel *allCells = new QLabel("Cells in the field:");
55      allCells->setStyleSheet(QLabelStyle);
56      QLabel *cellsLeft = new QLabel("Cells left:          ");
57      cellsLeft->setStyleSheet(QLabelStyle);
58      QLabel *minesLeft = new QLabel("Mines left:         ");
59      minesLeft->setStyleSheet(QLabelStyle);
60      QLabel *flagsLeft = new QLabel("You have flags:  ");
61      flagsLeft->setStyleSheet(QLabelStyle);
62
63
64      fieldHeightNumber = new QLabel(QString::number(core->getSizeX()));
65          fieldHeightNumber->setStyleSheet(QLabelStyle);
66      fieldLengthNumber = new QLabel(QString::number(core->getSizeY()));
67          fieldLengthNumber->setStyleSheet(QLabelStyle);
68      allCellsNumber = new QLabel(QString::number(core->getSizeX() * core->getSizeY()));
69          allCellsNumber->setStyleSheet(QLabelStyle);
70      cellsLeftNumber = new QLabel(QString::number(core->cellsLeft));
71          cellsLeftNumber->setStyleSheet(QLabelStyle);
72      minesLeftNumber = new QLabel(QString::number(core->minesLeft));
73          minesLeftNumber->setStyleSheet(QLabelStyle);
74      flagsLeftNumber = new QLabel(QString::number(core->flagsLeft));
75          flagsLeftNumber->setStyleSheet(QLabelStyle);
76
77      QHBoxLayout *panelHorizontal1 = new QHBoxLayout;
78      panelHorizontal1->addWidget(fieldHeight);
79      panelHorizontal1->addWidget(fieldHeightNumber);
80      QHBoxLayout *panelHorizontal2 = new QHBoxLayout;
81      panelHorizontal2->addWidget(fieldLength);
82      panelHorizontal2->addWidget(fieldLengthNumber);
83      QHBoxLayout *panelHorizontal3 = new QHBoxLayout;
84      panelHorizontal3->addWidget(allCells);
85      panelHorizontal3->addWidget(allCellsNumber);
86      QHBoxLayout *panelHorizontal4 = new QHBoxLayout;
87      panelHorizontal4->addWidget(cellsLeft);
88      panelHorizontal4->addWidget(cellsLeftNumber);
89      QHBoxLayout *panelHorizontal5 = new QHBoxLayout;
90      panelHorizontal5->addWidget(minesLeft);
91      panelHorizontal5->addWidget(minesLeftNumber);
92      QHBoxLayout *panelHorizontal6 = new QHBoxLayout;
93      panelHorizontal6->addWidget(flagsLeft);
94      panelHorizontal6->addWidget(flagsLeftNumber);
95
96      //timer = new QTimer;
97      //timer->start(10000);
98
99      panelLayout = new QVBoxLayout;
100     mainPanelLayout = new QVBoxLayout;
101     panelBox = new QGroupBox;
102
103     panelLayout->addSpacing(5);
104     panelLayout->addLayout(panelHorizontal1);
105     panelLayout->addLayout(panelHorizontal2);
106     panelLayout->addLayout(panelHorizontal3);
107     panelLayout->addLayout(panelHorizontal4);
108     panelLayout->addLayout(panelHorizontal5);
109     panelLayout->addLayout(panelHorizontal6);
110     panelBox->setLayout(panelLayout);
111
112     mainPanelLayout->addWidget(panelBox);
113     //mainPanelLayout->addWidget(timer);
114     mainPanelLayout->addStretch(1);
115     mainPanelLayout->addWidget(refreshButton);
116     mainPanelLayout->addWidget(backButton);
117     mainPanelLayout->setSpacing(20);
118
119
120
121
122     for (int i = 0; i < core->getSizeX(); i++)
123         for (int j = 0; j < core->getSizeY(); j++)
124         {
125             buttons[i][j] = new QMyPushButton;
126             fieldButtonSize = new QSize(32,32);
127             buttons[i][j]->setStyleSheet(QFieldButtonStyle);
128             buttons[i][j]->setFixedSize(*fieldButtonSize);
```

```
129             //buttons[i][j]->setText("Do it!");
130             fieldLayout->addWidget(buttons[i][j], i, j, 1, 1);
131             buttons[i][j]->setProperty("coordinates", i * 1000 + j);
132             connect(buttons[i][j], SIGNAL(pressed()), this, SLOT(clickedLeft()));
133             connect(buttons[i][j], SIGNAL(rClicked()), this, SLOT(setFlag()));
134             //QMyPushButton *but = new QMyPushButton;
135             //connect(but, SIGNAL())
136
137         }
138     //mainFieldLayout->addSpacing(15);
139     fieldLayout->setSpacing(0);
140     mainFieldLayout->addLayout(fieldLayout);
141
142     //mainLayout->addSpacing(15);
143     mainLayout->addLayout(mainFieldLayout);
144     mainLayout->addSpacing(25);
145     mainLayout->addLayout(mainPanelLayout);
146     //mainLayout->addSpacing(15);
147
148     QWidget *centralWidget = new QWidget;
149     centralWidget->setLayout(mainLayout);
150     this->setCentralWidget(centralWidget);
151     this->setWindowTitle("Minesweeper TOP Game");
152     this->setWindowIcon(QIcon(":/resources/images/icon.png"));
153
154     connect(this, SIGNAL(allCellsOpen()), this, SLOT(winGame()));
155     connect(this, SIGNAL(allah_BABAH()), this, SLOT(loseGame()));
156
157
158 }
159
160 GameWindow::~GameWindow()
161 {
162     delete ui;
163 }
164
165 void GameWindow::clickedLeft()
166 {
167 QMyPushButton *button = qobject_cast<QMyPushButton*>(sender());
168 QVariant index = button->property("coordinates");
169 int coordinates = index.toInt();
170 int x = coordinates / 1000;
171 int y = coordinates - x * 1000;
172 if (core->getPieceOfField(x,y).isFlag() == 0)
173 {
174     core->open(x,y);
175     repaint();
176     if (core->getPieceOfField(x,y).isMine())
177     {
178         core->openAllCells();
179         repaint();
180         emit allah_BABAH();
181     }
182     if (core->minesLeft == 0)
183         emit allCellsOpen();
184
185 }
186 }
187 void GameWindow::repaint()
188 {
189     for (int i = 0; i < core->getSizeX(); i++)
190         for (int j = 0; j < core->getSizeY(); j++)
191         {
192             //buttons[i][j]->setStyleSheet(QFieldButtonStyle);
193
194             if (core->getPieceOfField(i,j).isOpen() && core->getPieceOfField(i,j).isMine() ==
    ↪ 0)
195             {
196                 buttons[i][j]->setStyleSheet(OpenButtonStyle);
197                 if (core->getPieceOfField(i,j).getValue())
198                     buttons[i][j]->setText(QString::number(core->getPieceOfField(i,j).getValue
    ↪ ()));
199
200             }
201             if (core->getPieceOfField(i,j).isOpen() && core->getPieceOfField(i,j).isMine())
202             {
```

```
203                 buttons[i][j]->setStyleSheet(MinesButtonStyle);
204                 //buttons[i][j]->setIcon(QIcon(":/resources/images/mine.png"));
205                 //button->setText("Лох))00)");
206             }
207             if (core->getPieceOfField(i,j).isFlag())
208             {
209                 buttons[i][j]->setStyleSheet(FlagButtonStyle);
210             }
211         }
212     cellsLeftNumber->setText(QString::number(core->cellsLeft));
213     flagsLeftNumber->setText(QString::number(core->flagsLeft));
214     minesLeftNumber->setText(QString::number(core->minesLeft));
215 }
216
217 void GameWindow::backToMenu()
218 {
219     MainWindow *mainWnd = new MainWindow;
220     mainWnd->show();
221     this->close();
222 }
223
224 void GameWindow::setFlag()
225 {
226     QMyPushButton *button = qobject_cast<QMyPushButton*>(sender());
227     QVariant index = button->property("coordinates");
228     int coordinates = index.toInt();
229     int x = coordinates / 1000;
230     int y = coordinates - x * 1000;
231     if (core->getPieceOfField(x,y).isOpen() == 0 || core->getPieceOfField(x,y).isFlag())
232         core->setFlag(x,y);
233     if (core->getPieceOfField(x,y).isFlag())
234         button->setStyleSheet(FlagButtonStyle);
235     if (core->getPieceOfField(x,y).isFlag() == 0)
236         button->setStyleSheet(QFieldButtonStyle);
237     repaint();
238
239     if (core->minesLeft == 0)
240         emit allCellsOpen();
241 }
242
243 void GameWindow::loseGame()
244 {
245     QMessageBox msgBox;
246     msgBox.setWindowIcon(QIcon(":/resources/images/icon.png"));
247     msgBox.setWindowTitle("Поражение!");
248     msgBox.setText("Поздравляем!");
249     msgBox.setInformativeText("Ты␣проиграл!␣Ухади!");
250
251     msgBox.exec();
252 }
253
254 void GameWindow::winGame()
255 {
256     QMessageBox msgBox;
257     msgBox.setWindowIcon(QIcon(":/resources/images/icon.png"));
258     msgBox.setWindowTitle("Победа!");
259     msgBox.setText("Поздравляем!");
260     msgBox.setInformativeText("Ты␣победил!␣Ухади!");
261
262     msgBox.exec();
263 }
264
265 void GameWindow::refreshGame()
266 {
267     GameWindow *newWindow = new GameWindow(sizeOfFieldX, sizeOfFieldY, minesInTheField);
268     newWindow->show();
269     this->close();
270 }
```

//

```
1 #ifndef OPTIONSWINDOW_H
2 #define OPTIONSWINDOW_H
3
4 #include <QMainWindow>
5
6 namespace Ui {
```

```
7  class OptionsWindow;
8  }
9
10 class OptionsWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     explicit OptionsWindow(QWidget *parent = 0);
16     ~OptionsWindow();
17
18 signals:
19
20 public slots:
21     void backToMenu();
22     void saveChanges();
23
24 private:
25     Ui::OptionsWindow *ui;
26
27     const QSize SCREEN_SIZE{850, 600};
28
29     QString QPushButtonStyle =
30                 "QPushButton "
31                 "{"
32                     " border: 1px solid #000000;"
33                     " border-image: url(:/resources/images/button.jpg);"
34                     " padding: 7.5px 15px;"
35                     " border-radius: 10px;"
36                     " color: #000000;"
37                     " font-size: 18px;"
38                 "}"
39                 "QPushButton:flat "
40                 "{"
41                     " border: none;"
42                 "}"
43                 "QPushButton:hover "
44                 "{"
45                     " border-image: url(:/resources/images/button_hover);"
46                 "}"
47                 "QPushButton:pressed "
48                 "{"
49                     " border-image: url(:/resources/images/button_background.jpg);"
50                     " border-color #00ffff;"
51                 "}";
52 };
53
54 #endif // OPTIONSWINDOW_H
```

```
1  #include "optionswindow.h"
2  #include "ui_optionswindow.h"
3  #include "mainwindow.h"
4
5  OptionsWindow::OptionsWindow(QWidget *parent) :
6      QMainWindow(parent),
7      ui(new Ui::OptionsWindow)
8  {
9      ui->setupUi(this);
10
11     this->setFixedSize(SCREEN_SIZE);
12
13     QPixmap background(":/resources/images/options_background.jpg");
14         QPalette qPalette;
15         qPalette.setBrush(this->backgroundRole(),QBrush(background));
16         this->setPalette(qPalette);
17
18     this->setWindowTitle("Minesweeper TOP Game");
19     this->setWindowIcon(QIcon(":/resources/images/icon.png"));
20
21     ui->backButton->setStyleSheet(QPushButtonStyle);
22     ui->saveButton->setStyleSheet(QPushButtonStyle);
23
24     connect(ui->backButton, SIGNAL(clicked(bool)), this, SLOT(backToMenu()));
25     connect(ui->saveButton, SIGNAL(clicked(bool)), this, SLOT(saveChanges()));
26 }
27
```

```
28 OptionsWindow::~OptionsWindow()
29 {
30     delete ui;
31 }
32
33 void OptionsWindow::backToMenu()
34 {
35     MainWindow *mainWnd = new MainWindow;
36     mainWnd->show();
37     this->close();
38 }
39
40 void OptionsWindow::saveChanges()
41 {
42
43 }
```

//

```
1  #ifndef APPLICATION_H
2  #define APPLICATION_H
3
4  #include "field.h"
5  class Application
6  {
7  public:
8      Application();
9
10     /**
11      * @brief viewMainMenu Отьпажает главное меню
12      * @return 0 в случае отработки
13      */
14     int viewMainMenu() const;
15
16     /**
17      * @brief startGame Запускает игру
18      */
19     void startGame();
20
21     int enterDifficulty();
22
23     /**
24      * @brief paintField Рисует в консоли поле
25      * @param field Поле, которое надо отрисовать
26      * @return 0 в случае успеха
27      */
28     int paintField(Field field);
29
30     /**
31      * @brief open Открывает ячейку  все соседние пустые в радиусе 1
32      * @param x 1 координата
33      * @param y 2 координата
34      */
35     void open(int x, int y);
36
37     /**
38      * @brief setFlag Устанавливает флаг по заданным координатам
39      * @param x 1 координата
40      * @param y 2 координата
41      */
42     void setFlag(int x, int y);
43
44     /**
45      * @brief checkLose Проверка на конец игры поражение()
46      */
47     void checkLose();
48
49     /**
50      * @brief checkWin Проверка на конец игры победа()
51      */
52     void checkWin();
53
54 protected:
55     /**
56      * @brief calculateFlagsLeft Высчитывает количетво оставшихся флагов
57      * @return Количество флагов
58      */
```

```
59      int calculateFlagsLeft();
60
61      /**
62       * @brief calculateMinesLeft Высчитывает количетво оставшихся мин
63       * @return Количество мин
64       */
65      int calculateMinesLeft();
66
67      /**
68       * @brief help Выводит доступные команды и справку по игре
69       */
70      void help();
71
72      /**
73       * @brief enterCommands Ввод команд
74       */
75      void enterCommands();
76      void noManyFlags();
77
78      int difficulty;
79      Field* appField;
80      int x;
81      int y;
82      int minesLeft;
83      int facticalMinesNumber;
84      int flagsLeft;
85      string command;
86      bool isWin;
87      bool gameActiv;
88      int sizeX;
89      int sizeY;
90      int mines;
91  };
92
93  #endif // APPLICATION_H
```

```
1   #include "application.h"
2   #include <iostream>
3   Application::Application()
4   {
5   isWin = 0;
6   }
7   int Application::enterDifficulty()
8   {
9       std::cin  >> Application::difficulty;
10      return difficulty;
11  }
12  int Application::paintField(Field field)
13  {
14      cout << "␣␣␣␣+0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+10␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+20␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣+30" << endl;
15      cout << "␣␣␣␣";
16      for (int i = 0; i < 32; i++)
17          cout << i % 10 << "␣";
18
19      cout << endl;
20      cout << endl;
21
22      for (int i = 0; i < field.getSizeX(); i++)
23      {
24          if (i < 10)
25              cout << i << "␣␣␣";
26          else cout << i << "␣␣";
27          for (int j = 0; j < field.getSizeY(); j++)
28          {
29              if (field.getPieceOfField(i, j).isFlag())
30                  std::cout << "F" << "␣";
31              else if (field.getPieceOfField(i, j).isOpen())
32              {
33                  if (field.getPieceOfField(i, j).isMine())
34                      std::cout << "M" << "␣";
35                  else
36                      std::cout << field.getPieceOfField(i, j).getValue() << "␣";
37              }
38              else
39                  std::cout << "_␣";
40          }
```

```
41          std::cout << std::endl;
42      }
43      std::cout << "Mines left: " << minesLeft << endl;
44      std::cout << "Flags left: " << flagsLeft << endl;
45      //std::cout << endl;
46
47
48      return 0;
49  }
50  int Application::viewMainMenu() const
51  {
52      std::cout << "Good day! It's a MINESWEEPER game." << std::endl << std::endl;
53  //    std::cout << "Choice your level: (1 - newbie, 2 - gamer, 3 - professional)" << std::endl
        ↪ ;
54  //    enterDifficulty();
55  //    std::cout << "HaHa! Level of difficulty , which you have entered , has no effect!" << std
        ↪ ::endl;
56      std::cout << "The program itself will decide on which field you play!!" << std::endl;
57      std::cout << "Lets play a game))00))" << std::endl << std::endl;
58      system("pause");
59      std::cout << endl;
60      std::cout << endl;
61      return 0;
62  }
63  void Application::startGame()
64  {
65      sizeX = 16;
66      sizeY = 32;
67      mines = 50;
68      isWin = 0;
69      appField = new Field(sizeX, sizeY, mines);
70      facticalMinesNumber = calculateMinesLeft();
71      flagsLeft = facticalMinesNumber;
72      while (appField->isGameActive())
73      {
74          paintField(*appField);
75          enterCommands();
76      }
77      std::cout << "Would you like try again? Y/N" << std::endl;
78      cin >> command;
79      if (command != "y" && command != "Y")
80          exit(0);
81  }
82  void Application::open(int x, int y)
83  {
84      appField->open(x,y);
85      checkLose();
86      calculateMinesLeft();
87  }
88  int Application::calculateMinesLeft()
89  {
90      minesLeft = 0;
91      for (int i = 0; i < appField->getSizeX(); i++)
92          for (int j = 0; j < appField->getSizeY(); j++)
93              if (appField->getPieceOfField(i, j).isMine() && appField->getPieceOfField(i, j).
        ↪ isFlag() == 0)
94                  minesLeft++;
95      return minesLeft;
96
97  }
98  int Application::calculateFlagsLeft()
99  {
100     flagsLeft = facticalMinesNumber;
101     return flagsLeft;
102
103 }
104 void Application::setFlag(int x, int y)
105 {
106     if (flagsLeft > 0)
107     {
108     appField->setFlag(x, y);
109     if (appField->getPieceOfField(x, y).isFlag())
110         flagsLeft--;
111     else flagsLeft++;
112     }
113     else noManyFlags();
```

```
114     calculateMinesLeft();
115     checkWin();
116 }
117 void Application::help()
118 {
119     std::cout << "   Command list: " << std::endl;
120     std::cout << "1. open - for open cell with coordinates X and Y. Press /enter/ between
    ↪ entering first and second coordinates" << std::endl;
121     std::cout << "2. flag - for set flag in cell with coordinates X and Y. Press /enter/
    ↪ between entering first and second coordinates" << std::endl;
122     std::cout << "3. exit - for close app" << std::endl;
123     system("pause");
124 //    std::cout << "" << std::endl;
125 //    std::cout <<  << std::endl;
126 //    std::cout <<  << std::endl;
127
128 }
129 void Application::noManyFlags()
130 {
131     std::cout << "You haven't flags more" << std::endl;
132 }
133 void Application::enterCommands()
134 {
135     cout << "Enter command" << endl;
136     cin >> command;
137     SWITCH (command)
138     {
139     CASE ("exit"):
140         cout << "Game over" << endl;
141         exit(0);
142         break;
143     CASE ("open"):
144         cout << "Enter coordinates of cell, which must be opened" << endl;
145         cin >> y >> x;
146         open(x, y);
147         break;
148     CASE ("openall"):
149         appField->openAllCells();
150         break;
151     CASE ("flag"):
152         cout << "Enter coordinates of cell, where you want set flag" << endl;
153         cin >> y >> x;
154         setFlag(x, y);
155         break;
156     CASE ("help"):
157          help();
158         break;
159     DEFAULT:
160         help();
161     }
162 }
163 void Application::checkLose()
164 {
165     std::string command;
166     if (appField->isLose())
167     {
168         appField->openAllCells();
169         paintField(*appField);
170         cout << endl;
171         cout << "!!!You lose!!!" << endl;
172         appField->gameActive = 0;
173
174     }
175 }
176 void Application::checkWin()
177 {
178     if (minesLeft < 1)
179         isWin = 1;
180     if (isWin)
181     {
182         std::cout << "GRATULATIONS!!! YOU WIN" << std::endl;
183         appField->gameActive = 0;
184     }
185 }
```

//