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

Программирование

Отчет по курсовой работе
"Игра Реверси (Отелло)"

Работу выполнил:
Бойцов К.С.
Группа:
13501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2016

Содержание

1 Игра Реверси

1.1 Задание

Реализовать настольную игру Реверси.

1.2 Правила игры

В игре используется квадратная доска размером 8x8 (все клетки могут быть одного цвета) и 64 специальные фишки, окрашенные с разных сторон в контрастные цвета, например, в белый и чёрный. Клетки доски нумеруются от верхнего левого угла: вертикали - латинскими буквами, горизонтали - цифрами. Один из игроков играет белыми, другой - чёрными. Делая ход, игрок ставит фишку на клетку доски "своим"цветом вверх.

В начале игры в центр доски выставляются 4 фишки: чёрные на d5 и e4, белые на d4 и e5. Первый ход делают чёрные. Далее игроки ходят по очереди. Делая ход, игрок должен поставить свою фишку на одну из клеток доски таким образом, чтобы между этой поставленной фишкой и одной из имеющихся уже на доске фишек его цвета находился непрерывный ряд фишек соперника, горизонтальный, вертикальный или диагональный (другими словами, чтобы непрерывный ряд фишек соперника оказался "закрыт"фишками игрока с двух сторон). Все фишки соперника, входящие в "закрытый"на этом ходу ряд, переворачиваются на другую сторону (меняют цвет) и переходят к ходившему игроку.

Если в результате одного хода "закрывается"одновременно более одного ряда фишек противника, то переворачиваются все фишки, оказавшиеся на всех "закрытых"рядах.

Игрок вправе выбирать любой из возможных для него ходов. Если игрок имеет возможные ходы, он не может отказаться от хода. Если игрок не имеет допустимых ходов, то ход передаётся сопернику.

Игра прекращается, когда на доску выставлены все фишки или когда ни один из игроков не может сделать хода. По окончании игры проводится подсчёт фишек каждого цвета, и игрок, чьих фишек на доске выставлено больше, объявляется победителем. В случае равенства количества фишек засчитывается ничья.

1.3 Концепция

Готовый проект должен предполагать возможность игры двух игроков.

1.4 Минимально работоспособный продукт

Минимальным работоспособным продуктом должно быть консольное приложение, позволяющее играть двоим игрокам.

1.5 Диаграмма прецедентов использования

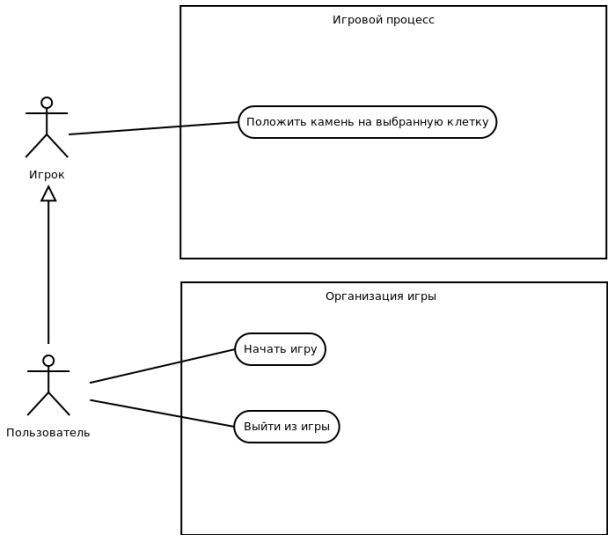

Рис. 1: Диаграмма прецедентов использования

2 Проектрование приложения, реализующего игру Реверси

Программа разделена на 3 подпроекта: app - консольное приложение, core - библиотека, реализующая модель Жизнь, gui - графическое приложение.

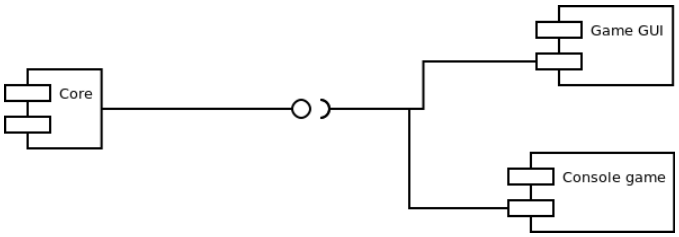

Рис. 2: Диаграмма прецедентов использования

3 Реализация игры Реверси

3.1 Версии программ

Операционная система: Ubuntu 15.10, среда разработки: Qt Creator 3.5.0, компилятор: GCC 4.9.1.

3.2 Библиотека

Основные классы, выделенные в библиотеке:

- Класс Dot. Реализует клетку. Содержит координаты клетки и её состояние. Присутствуют методы, возвращающие и задающие координаты и состояние клетки.
- Класс Field. Класс представляет поле модели. Содержит размер поля и двумерный массив клеток.
- Класс Game. Класс представляет методы, содержащие правила игры.

3.3 Графическое приложение

Графическое приложение позволяет работать с моделью через окна графического приложения.
Классы, выделенные в графическом приложении.

- Класс MainWindow. Главное окно приложения. Присутствуют кнопки «Новая игра» и «Выход».
- Класс GrafField. Строит графическую модель поля.

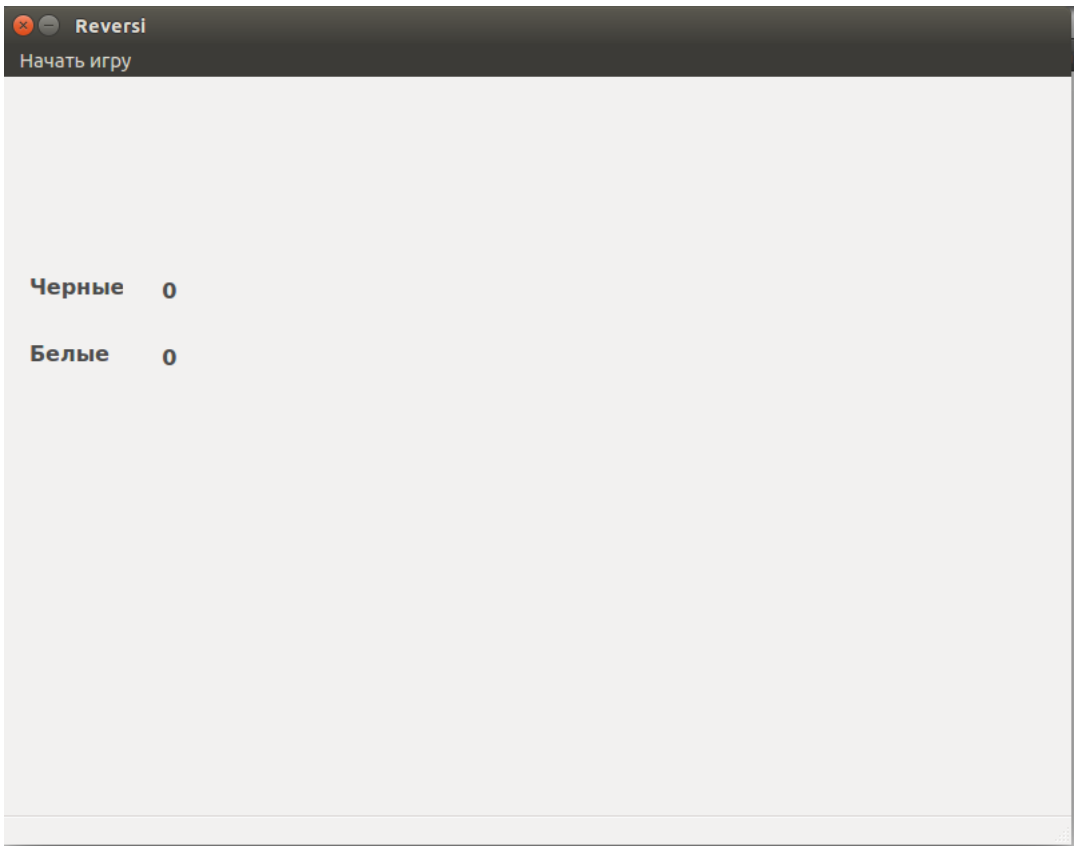

Рис. 3: Главное меню графического приложения

На рис 3 представлено главное окно приложения. В нём пользователю можно начать новую игру.

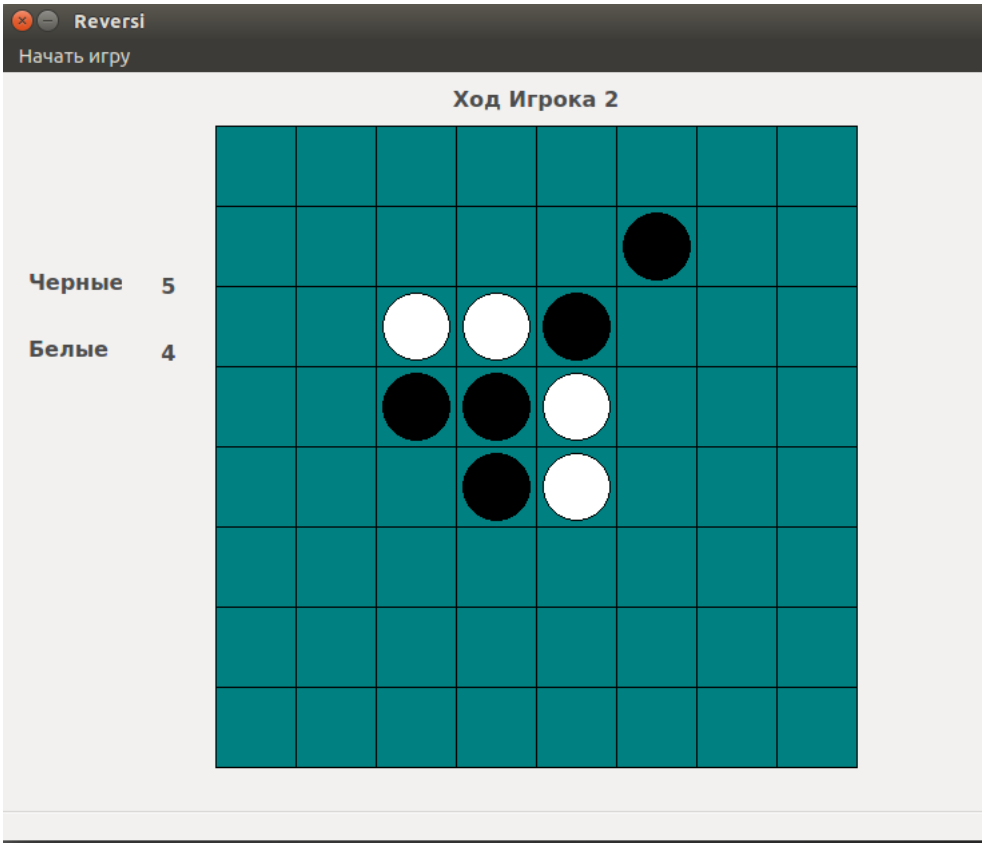

Рис. 4: Игровое поле

На рис 4 представлено игровое поле.

4 Вывод

По окончании семестра автор проекта научился делать графический интерфейс с помощью Qt, а также получил опыт работы с большими проектами, содержащими много классов и имеющих как консольное приложение, так и графическое.

5 Приложение 1. Листинги кода

5.1 Консольное приложение

```
1 #include <iostream>
2
3 #include "Board.h"
```

```
4 #include "Game.h"
5
6
7 int main()
8 {
9     Board B;
10    Game Game1(&B);
11    Game1.TheGame();
12    system("pause");
13    return 0;
14 }
```

```h
#pragma once
#include "Board.h"
#include "Dot.h"
class Game
{
private:
    GameState GamS;
    Board *GameBoard;
    static const int Bsize = 8;
public:
    Game(Board *B) {
        GameBoard = B;
        GamS = Player1;
    }
    ~Game() {}
    void TheGame();
    bool CheckAndGo(GameState Player, int x, int y, bool Chk);
    bool Check();
    bool CheckPlayerGo(GameState Player);
};
```

```cpp
#include "stdafx.h"
#include "Game.h"

void Dot::Print()
{
    if (_state == Empty)
        std::cout << "[ ]";
    else if (_state == Black)
        std::cout << "[B]";
    else if (_state == White)
        std::cout << "[W]";
}

void Board::PrintBoard()
{
    std::cout << "  _1__2__3__4__5__6__7__8_" << std::endl;
    for (int i = 0; i < Bsize; i++) {
        std::cout << i + 1 << "|";
        for (int j = 0; j < Bsize; j++)
            dots[j][i].Print();
        std::cout << std::endl;
    }
}

void Game::TheGame()
{
    int x, y;
    if (!Check())
        GamS = EndGame;
    while (GamS!=EndGame)
    {
        switch (GamS)
        {
        case Player1:
        {
            system("cls");
            GameBoard->PrintBoard();
            std::cout << ("Ход игрока 1") << std::endl << std::endl;
            //std::cout << GameBoard->BlackCount();
            //std::cout << GameBoard->WhiteCount();
```

```
41              if (!(CheckPlayerGo(GamS)))
42              {
43                  std::cout << "Нет возможных ходов" << std::endl;
44                  system("pause");
45                  GamS = Player2;
46                  break;
47              }
48              std::cout << ("Введите координату по горизонтали: ") << std
    ↪ ::endl;
49              std::cin >> x;
50              std::cout << ("Введите координату по вертикали: ") << std::
    ↪ endl;
51              std::cin >> y;
52              if (CheckAndGo(Player1, x - 1, y - 1, false))
53                  GamS = Player2;
54              break;
55          }
56
57          case Player2:
58              system("cls");
59              GameBoard->PrintBoard();
60              std::cout << ("Ход игрока 2") << std::endl << std::endl;
61              //std::cout << GameBoard->BlackCount();
62              //std::cout << GameBoard->WhiteCount();
63              if (!(CheckPlayerGo(GamS)))
64              {
65                  std::cout << "Нет возможных ходов" << std::endl;
66                  system("pause");
67                  GamS = Player1;
68                  break;
69              }
70              std::cout << ("Введите координату по горизонтали: ") << std
    ↪ ::endl;
71              std::cin >> x;
72              std::cout << ("Введите координату по вертикали: ") << std::
    ↪ endl;
73              std::cin >> y;
74              if (CheckAndGo(Player2, x - 1, y - 1, false))
75                  GamS = Player1;
76              break;
77          case EndGame:
78              system("cls");
79              if ((GameBoard->BlackCount()) > (GameBoard->WhiteCount
    ↪ ()))
80                  std::cout << "Игрок 1 победил!" << std::endl;
81              else if ((GameBoard->BlackCount()) < (GameBoard->
    ↪ WhiteCount()))
82                  std::cout << "Игрок 2 победил!" << std::endl;
83              else
84                  std::cout << "Ничья!" << std::endl;
85              system("pause");
86              break;
87          default:
88              break;
89          }
90      }
91  }
92
93  bool Game::CheckAndGo(GameState Player,int x, int y, bool Chk)
94  {
95      bool fl = false;
96      bool go = false;
```

```
97      int k, g;
98      int p = 0;
99      int s = 0;
100     State Check = Empty;
101     if (GameBoard->GetState(x, y) != Empty)
102         return go;
103     if (Player == Player1)
104     {
105         Check = Black;
106     }
107     if (Player==Player2)
108     {
109         Check = White;
110     }
111
112     // горизонтали------
113
114     // слева---------- от текущей клетки
115     for ( fl = false, k = y - 1; k >= 0 && Empty != GameBoard->
        ↪ GetState(x,k); --k)
116     {
117         if (GameBoard->GetState(x, k) == Check)
118         {
119             fl = true;
120             break;
121         }
122         else
123             p++;
124     }
125
126     if (fl&&p>0) {
127         s = 0;
128         while ((s <= p) && !(Chk))
129         {
130             GameBoard->SetState(x, y-s, Check);
131             s++;
132         }
133         if (Chk)
134             return true;
135         else
136             go = true;
137     }
138
139     p = 0;
140
141     // справа----------
142
143     for (fl = false, k = y + 1; k < Bsize && Empty != GameBoard->
        ↪ GetState(x, k); ++k)
144     {
145         if (GameBoard->GetState(x, k) == Check)
146         {
147             fl = true;
148             break;
149         }
150         else
151             p++;
152     }
153
154     if (fl&&p>0)
155     {
156         s = 0;
```

```
157             while ((s <= p) && !(Chk))
158             {
159                 GameBoard->SetState(x, y + s, Check);
160                 s++;
161             }
162             if (Chk)
163                 return true;
164             else
165                 go = true;
166         }
167
168         p = 0;
169
170         // —— вертикали
171         // ————————— снизу от клетки
172
173         for (fl = false, k = x - 1; k >= 0 && Empty != GameBoard->
            ↪ GetState(k, y); --k)
174         {
175             if (GameBoard->GetState(k, y) == Check)
176             {
177                 fl = true;
178                 break;
179             }
180             else
181                 p++;
182         }
183
184         if (fl&&p>0) {
185             s = 0;
186             while ((s <= p) && !(Chk))
187             {
188                 GameBoard->SetState(x-s, y , Check);
189                 s++;
190             }
191             if (Chk)
192                 return true;
193             else
194                 go = true;
195         }
196
197         p = 0;
198
199         //————————— сверху
200
201         for (fl = false, k = x + 1; k < Bsize && Empty != GameBoard->
            ↪ GetState(k, y); ++k)
202         {
203             if (GameBoard->GetState(k, y) == Check)
204             {
205                 fl = true;
206                 break;
207             }
208             else
209                 p++;
210         }
211
212         if (fl&&p>0)
213         {
214             s = 0;
215             while ((s <= p) && !(Chk))
216             {
```

```
217                 GameBoard->SetState(x+s, y, Check);
218                 s++;
219             }
220             if (Chk)
221                 return true;
222             else
223                 go = true;
224         }
225
226         p = 0;
227
228         //------ главная диагональ
229         // ---------- слева сверху
230
231         for (fl = false, g = x - 1, k = y - 1; k >= 0 && g >= 0 &&
        ↪ Empty != GameBoard->GetState(g, k); --g, --k)
232         {
233             if (GameBoard->GetState(g, k) == Check)
234             {
235                 fl = true;
236                 break;
237             }
238             else
239                 p++;
240 }
241
242         if (fl&&p>0)
243         {
244             s = 0;
245             while ((s <= p) && !(Chk))
246             {
247                 GameBoard->SetState(x-s, y - s, Check);
248                 s++;
249             }
250             if (Chk)
251                 return true;
252             else
253                 go = true;
254         }
255
256         p = 0;
257
258         //справа---------- снизу
259
260         for (fl = false, g = x + 1, k = y + 1; k < Bsize && g < Bsize
        ↪ && Empty != GameBoard->GetState(g, k); ++g, ++k)
261         {
262             if (GameBoard->GetState(g, k) == Check)
263             {
264                 fl = true;
265                 break;
266             }
267             else
268                 p++;
269         }
270
271         if (fl&&p>0)
272         {
273             s = 0;
274             while ((s <= p) && !(Chk))
275             {
276                 GameBoard->SetState(x + s, y + s, Check);
```

```
277                 s++;
278             }
279             if (Chk)
280                 return true;
281             else
282                 go = true;
283         }
284
285         p = 0;
286
287         // ------ побочная диагональ
288         // ------------ справа сверху
289
290         for (fl = false, g = x - 1, k = y + 1; k < Bsize && g >= 0 &&
        ↪ Empty != GameBoard->GetState(g, k); --g, ++k)
291         {
292             if (GameBoard->GetState(g, k) == Check)
293             {
294                 fl = true;
295                 break;
296             }
297             else
298                 p++;
299         }
300
301         if (fl&&p>0)
302         {
303             s = 0;
304             while ((s <= p) && !(Chk))
305             {
306                 GameBoard->SetState(x-s, y + s, Check);
307                 s++;
308             }
309             if (Chk)
310                 return true;
311             else
312                 go = true;
313         }
314
315         p = 0;
316
317         // ---------------- слева снизу
318
319         for (fl = false, g = x + 1, k = y - 1; k >= 0 && g < Bsize &&
        ↪ Empty != GameBoard->GetState(g, k); ++g, --k)
320         {
321             if (GameBoard->GetState(g, k) == Check)
322             {
323                 fl = true;
324                 break;
325             }
326             else
327                 p++;
328         }
329
330         if (fl&&p>0)
331         {
332             s = 0;
333             while ((s <= p) && !(Chk))
334             {
335                 GameBoard->SetState(x + s, y - s, Check);
336                 s++;
```

```
337             }
338             if(Chk)
339                 return true;
340             else
341                 go = true;
342         }
343
344         p = 0;
345
346         return go;
347     }
348
349     bool Game::Check()
350     {
351         bool emp = false;
352         for (int i = 0; i < Bsize; i++)
353             for (int j = 0; j < Bsize; j++)
354                 if (GameBoard->GetState(i, j) == Empty)
355                     emp = true;
356         if (!emp) return emp;
357         bool pl1 = CheckPlayerGo(Player1);
358         bool pl2 = CheckPlayerGo(Player2);
359         if (pl1 || pl2)
360             return true;
361         if ((!pl1) && (!pl2))
362             return false;
363
364     }
365
366     bool Game::CheckPlayerGo(GameState Player)
367     {
368         bool pl = false;
369         for (int i = 0; i < Bsize; i++)
370             for (int j = 0; j < Bsize; j++)
371                 if (CheckAndGo(Player, i, j, true))
372                     pl = true;
373         return pl;
374     }
```

5.2 Графическое приложение

```
1  #include "mainwindow.h"
2  #include <QApplication>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      MainWindow w;
8      w.setWindowTitle("Reversi");
9      w.show();
10     return a.exec();
11 }
```

```
#ifndef GRAFFIELD_H
#define GRAFFIELD_H

#include <QFrame>
#include <QPainter>
#include <QWidget>
#include "Board.h"
#include <QtGui>

namespace Ui
{
class GrafField;
}

class GrafField : public QFrame
{
    Q_OBJECT
    Board *Field;

public:
    explicit GrafField(QWidget *parent = 0);
    ~GrafField();
    void paintEvent(QPaintEvent *);
    void paintBoard(QPainter &painter);
    void setBoard(Board *B);

private:
    Ui::GrafField *ui;
    static const int Xb=60;
};

#endif // GRAFFIELD_H
```

```
#include "graffield.h"
#include "ui_graffield.h"

void Board::PrintBoard(QPainter &painter)
{
    static const int Xb=60;
    for (int k = 0; k < Bsize; k++) {
        for (int c = 0; c < Bsize; c++){
        if(dots[c][k].GetState()==White)
            painter.setBrush(QBrush(Qt::white, Qt::SolidPattern));
        if(dots[c][k].GetState()==Black)
            painter.setBrush(QBrush(Qt::black, Qt::SolidPattern));
        if(dots[c][k].GetState()!=Empty)
            painter.drawEllipse(c*Xb+5,k*Xb+5,Xb-10,Xb-10);
        }
    }
}

GrafField::GrafField(QWidget *parent) :
    QFrame(parent),
    Field(0),
    ui(new Ui::GrafField)
{
    ui->setupUi(this);
}

GrafField::~GrafField()
{
```

```
29      delete ui;
30  }
31
32  void GrafField::setBoard(Board *B)
33  {
34      Field=B;
35      update();
36  }
37
38  void GrafField::paintEvent(QPaintEvent *)
39  {
40      if(!Field)
41          return;
42      QPainter painter(this);
43      paintBoard(painter);
44      Field->PrintBoard(painter);
45  }
46
47  void GrafField::paintBoard(QPainter &painter)
48  {
49      painter.setPen(Qt::black);
50      painter.fillRect(0,0,Xb*8,Xb*8, Qt::darkCyan);
51      for(int i=0;i<=Xb*8;i+=Xb)
52      {
53          painter.drawLine(i, 0, i, Xb*8);
54          painter.drawLine(0,i,Xb*8,i);
55      }
56  }
```

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "Game.h"
#include "graffield.h"
#include "Board.h"
#include "State.h"
namespace Ui
{
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
    Game *NewGame;
    Board *GameBoard;

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

private slots:
    void on_actionExit_triggered();

    void on_actionNew_Game_triggered();

private:
    Ui::MainWindow *ui;
    bool eventFilter(QObject *, QEvent *);
    static const int Xb=60;
    int x,y;
    void TheGame();
    GameState GamS;
};

#endif // MAINWINDOW_H
```

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QMessageBox"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    GameBoard(0),
    NewGame(0),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->frame->installEventFilter(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

bool MainWindow::eventFilter(QObject *obj, QEvent *evt)
{
    if((GameBoard)&&evt->type()==QEvent::MouseButtonPress)
```

```
22      {
23          qDebug()<<"Click";
24          QMouseEvent *me = (QMouseEvent*)evt;
25          if(me->button()==Qt::LeftButton)
26          {
27              x=me->x()/Xb;
28              y=me->y()/Xb;
29              qDebug()<<x<<y;
30              qDebug()<<"Clack";
31              TheGame();
32          }
33      }
34      return QMainWindow::eventFilter(obj,evt);
35  }
36
37  void MainWindow::TheGame()
38  {
39      qDebug()<<"Game";
40      if (NewGame->Check()==0)
41          GamS = EndGame;
42
43      switch (GamS)
44      {
45      case Player1:
46      {
47          qDebug()<<"Player1";
48          ui->frame->update();
49          if (!(NewGame->CheckPlayerGo(GamS)))
50          {
51              QMessageBox::information(0, "Смена игрока", "Нет
    ↪ Возможных ходов!");
52              GamS = Player2;
53              ui->label->setText("Ход Игрока 2");
54              break;
55          }
56          if (NewGame->CheckAndGo(Player1, x , y , false))
57          {
58              GamS = Player2;
59              ui->label->setText("Ход Игрока 2");
60          }
61          ui->label_3->setNum(GameBoard->BlackCount());
62          ui->label_5->setNum(GameBoard->WhiteCount());
63          break;
64      }
65
66      case Player2:
67          qDebug()<<"Player2";
68          ui->frame->update();
69          if (!(NewGame->CheckPlayerGo(GamS)))
70          {
71              QMessageBox::information(0, "Смена игрока", "Нет
    ↪ Возможных ходов!");
72              GamS = Player1;
73              ui->label->setText("Ход Игрока 1");
74              break;
75          }
76          if (NewGame->CheckAndGo(Player2, x, y, false))
77          {
78              ui->label->setText("Ход Игрока 1");
79              GamS = Player1;
80              ui->label_3->setNum(GameBoard->BlackCount());
81              ui->label_5->setNum(GameBoard->WhiteCount());
```

```
82          }
83          break;
84      case EndGame:
85          if ((GameBoard->BlackCount()) > (GameBoard->WhiteCount()))
86              QMessageBox::information(0, "Игра окончена.", "Игрок 1
    ↪ победил!");
87          else if ((GameBoard->BlackCount()) < (GameBoard->WhiteCount
    ↪ ()))
88              QMessageBox::information(0, "Игра окончена.", "Игрок 2
    ↪ победил!");
89          else
90              QMessageBox::information(0, "Игра окончена.", "Ничья!");
91          break;
92      default:
93          break;
94      }
95  }
96
97  void MainWindow::on_actionExit_triggered()
98  {
99       QCoreApplication::exit(1);
100 }
101
102 void MainWindow::on_actionNew_Game_triggered()
103 {
104     delete GameBoard;
105     GameBoard= new Board;
106     delete NewGame;
107     NewGame =new Game(GameBoard);
108     ui->frame->setBoard(GameBoard);
109     GamS=Player1;
110     ui->label->setText("Ход Игрока 1");
111     ui->label_3->setNum(GameBoard->BlackCount());
112     ui->label_5->setNum(GameBoard->WhiteCount());
113 }
```

5.3 Библиотека

```
#pragma once
#include "Board.h"
#include "Dot.h"
class Game
{
private:
    GameState GamS;
    Board *GameBoard;
    static const int Bsize = 8;
public:
    Game(Board *B) {
        GameBoard = B;
        GamS = Player1;
    }
    ~Game() {}
    void TheGame();
    bool CheckAndGo(GameState Player, int x, int y, bool Chk);
    bool Check();
    bool CheckPlayerGo(GameState Player);
};
```

```
#include "stdafx.h"
#include "Game.h"

void Dot::Print()
{
    if (_state == Empty)
        std::cout << "[ ]";
    else if (_state == Black)
        std::cout << "[B]";
    else if (_state == White)
        std::cout << "[W]";
}

void Board::PrintBoard()
{
    std::cout << "  _1__2__3__4__5__6__7__8_" << std::endl;
    for (int i = 0; i < Bsize; i++) {
        std::cout << i + 1 << "|";
        for (int j = 0; j < Bsize; j++)
            dots[j][i].Print();
        std::cout << std::endl;
    }
}

void Game::TheGame()
{
    int x, y;
    if (!Check())
        GamS = EndGame;
    while (GamS!=EndGame)
    {
        switch (GamS)
        {
        case Player1:
        {
            system("cls");
            GameBoard->PrintBoard();
            std::cout << ("[U+FFFD][U+FFFD][U+FFFD][U+FFFD]1") << std::
    ↪ endl << std::endl;
```

```
39              //std::cout << GameBoard->BlackCount();
40              //std::cout << GameBoard->WhiteCount();
41              if (!(CheckPlayerGo(GamS)))
42              {
43                  std::cout << "[U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD]<< std::endl;
44 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣system("pause");
45 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣GamS = Player2;
46 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣break;
47 ␣␣␣␣␣␣␣␣␣␣␣␣}
48 ␣␣␣␣␣␣␣␣␣␣␣␣std::cout << ("
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] ") << std::endl;
49 ␣␣␣␣␣␣␣␣␣␣␣␣std::cin >> x;
50 ␣␣␣␣␣␣␣␣␣␣␣␣std::cout << ("
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD] ") << std::endl;
51 ␣␣␣␣␣␣␣␣␣␣␣␣std::cin >> y;
52 ␣␣␣␣␣␣␣␣␣␣␣␣if (CheckAndGo(Player1, x - 1, y - 1, false))
53 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣GamS = Player2;
54 ␣␣␣␣␣␣␣␣␣␣␣␣break;
55 ␣␣␣␣␣␣␣␣}
56
57 ␣␣␣␣␣␣␣␣case Player2:
58 ␣␣␣␣␣␣␣␣␣␣␣␣system("cls");
59 ␣␣␣␣␣␣␣␣␣␣␣␣GameBoard->PrintBoard();
60 ␣␣␣␣␣␣␣␣␣␣␣␣std::cout << ("[U+FFFD][U+FFFD][U+FFFD][U+FFFD]2") << std::
    ↪ endl << std::endl;
61 ␣␣␣␣␣␣␣␣␣␣␣␣//std::cout << GameBoard->BlackCount();
62 ␣␣␣␣␣␣␣␣␣␣␣␣//std::cout << GameBoard->WhiteCount();
63 ␣␣␣␣␣␣␣␣␣␣␣␣if (!(CheckPlayerGo(GamS)))
64 ␣␣␣␣␣␣␣␣␣␣␣␣{
65 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣std::cout << "[U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD]<< std::endl;
66                  system("pause");
67                  GamS = Player1;
68                  break;
69              }
70              std::cout << ("
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] ") << std::endl;
71              std::cin >> x;
72              std::cout << ("
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
    ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD] ") << std::endl;
73              std::cin >> y;
74              if (CheckAndGo(Player2, x - 1, y - 1, false))
75                  GamS = Player1;
76              break;
77          case EndGame:
78              system("cls");
79              if ((GameBoard->BlackCount()) > (GameBoard->WhiteCount
    ↪ ()))
80                  std::cout << "[U+FFFD][U+FFFD]1 [U+FFFD][U+FFFD][U+FFFD]
    ↪ << std::endl;
81 ␣␣␣␣␣␣␣␣␣␣␣␣else if ((GameBoard->BlackCount()) < (GameBoard->
    ↪ WhiteCount()))
82 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣std::cout << "[U+FFFD][U+FFFD]2 [U+FFFD][U+FFFD][U+FFFD]
    ↪ << std::endl;
83              else
84                  std::cout << "[U+FFFD]!" << std::endl;
85              system("pause");
```

```
86                  break;
87              default:
88                  break;
89              }
90      }
91  }
92
93  bool Game::CheckAndGo(GameState Player,int x, int y, bool Chk)
94  {
95      bool fl = false;
96      bool go = false;
97      int k, g;
98      int p = 0;
99      int s = 0;
100     State Check = Empty;
101     if (GameBoard->GetState(x, y) != Empty)
102         return go;
103     if (Player == Player1)
104     {
105         Check = Black;
106     }
107     if (Player==Player2)
108     {
109         Check = White;
110     }
111
112     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD]————
113     //
        ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]————————————
        ↪   for ( fl = false, k = y - 1; k >= 0 && Empty != GameBoard
        ↪ ->GetState(x,k); --k)
114     {
115         if (GameBoard->GetState(x, k) == Check)
116         {
117             fl = true;
118             break;
119         }
120         else
121             p++;
122     }
123
124     if (fl&&p>0) {
125         s = 0;
126         while ((s <= p) && !(Chk))
127         {
128             GameBoard->SetState(x, y-s, Check);
129             s++;
130         }
131         if (Chk)
132             return true;
133         else
134             go = true;
135     }
136
137     p = 0;
138
139     // [U+FFFD][U+FFFD]————————————————————
140     for (fl = false, k = y + 1; k < Bsize && Empty != GameBoard->
        ↪ GetState(x, k); ++k)
141     {
142         if (GameBoard->GetState(x, k) == Check)
143         {
```

```
144                 fl = true;
145                 break;
146             }
147             else
148                 p++;
149         }
150 
151         if (fl&&p>0)
152         {
153             s = 0;
154             while ((s <= p) && !(Chk))
155             {
156                 GameBoard->SetState(x, y + s, Check);
157                 s++;
158             }
159             if (Chk)
160                 return true;
161             else
162                 go = true;
163         }
164 
165         p = 0;
166 
167         // ---- [U+FFFD][U+FFFD][U+FFFD] // ---------------- [U+FFFD][U+FFFD]
        ↪ [U+FFFD][U+FFFD]
168         for (fl = false, k = x - 1; k >= 0 && Empty != GameBoard->
        ↪ GetState(k, y); --k)
169         {
170             if (GameBoard->GetState(k, y) == Check)
171             {
172                 fl = true;
173                 break;
174             }
175             else
176                 p++;
177         }
178 
179         if (fl&&p>0) {
180             s = 0;
181             while ((s <= p) && !(Chk))
182             {
183                 GameBoard->SetState(x-s, y , Check);
184                 s++;
185             }
186             if (Chk)
187                 return true;
188             else
189                 go = true;
190         }
191 
192         p = 0;
193 
194         //---------------- [U+FFFD][U+FFFD]for (fl = false, k = x + 1; k <
        ↪ Bsize && Empty != GameBoard->GetState(k, y); ++k)
195         {
196             if (GameBoard->GetState(k, y) == Check)
197             {
198                 fl = true;
199                 break;
200             }
201             else
202                 p++;
```

```
203         }
204
205         if (fl&&p>0)
206         {
207             s = 0;
208             while ((s <= p) && !(Chk))
209             {
210                 GameBoard->SetState(x+s, y, Check);
211                 s++;
212             }
213             if (Chk)
214                 return true;
215             else
216                 go = true;
217         }
218
219         p = 0;
220
221         //------- [U+FFFD][U+FFFD]  [U+FFFD][U+FFFD][U+FFFD]
222         // ---------- [U+FFFD][U+FFFD][U+FFFD][U+FFFD]for (fl = false, g =
            ↪ x - 1, k = y - 1; k >= 0 && g >= 0 && Empty != GameBoard->
            ↪ GetState(g, k); --g, --k)
223         {
224             if (GameBoard->GetState(g, k) == Check)
225             {
226                 fl = true;
227                 break;
228             }
229             else
230                 p++;
231     }
232
233         if (fl&&p>0)
234         {
235             s = 0;
236             while ((s <= p) && !(Chk))
237             {
238                 GameBoard->SetState(x-s, y - s, Check);
239                 s++;
240             }
241             if (Chk)
242                 return true;
243             else
244                 go = true;
245         }
246
247         p = 0;
248
249         //[U+FFFD][U+FFFD][U+FFFD][U+FFFD]----------for (fl = false, g = x +
            ↪  1, k = y + 1; k < Bsize && g < Bsize && Empty != GameBoard
            ↪ ->GetState(g, k); ++g, ++k)
250         {
251             if (GameBoard->GetState(g, k) == Check)
252             {
253                 fl = true;
254                 break;
255             }
256             else
257                 p++;
258         }
259
260         if (fl&&p>0)
```

```
261     {
262         s = 0;
263         while ((s <= p) && !(Chk))
264         {
265             GameBoard->SetState(x + s, y + s, Check);
266             s++;
267         }
268         if (Chk)
269             return true;
270         else
271             go = true;
272     }
273
274     p = 0;
275
276     // ------ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
277     // ------------ [U+FFFD][U+FFFD][U+FFFD][U+FFFD]for (fl = false, g =
        ↪ x - 1, k = y + 1; k < Bsize && g >= 0 && Empty != GameBoard
        ↪ ->GetState(g, k); --g, ++k)
278     {
279         if (GameBoard->GetState(g, k) == Check)
280         {
281             fl = true;
282             break;
283         }
284         else
285             p++;
286     }
287
288     if (fl&&p>0)
289     {
290         s = 0;
291         while ((s <= p) && !(Chk))
292         {
293             GameBoard->SetState(x-s, y + s, Check);
294             s++;
295         }
296         if (Chk)
297             return true;
298         else
299             go = true;
300     }
301
302     p = 0;
303
304     // ---------------- [U+FFFD][U+FFFD][U+FFFD][U+FFFD]for (fl = false, g
        ↪  = x + 1, k = y - 1; k >= 0 && g < Bsize && Empty !=
        ↪ GameBoard->GetState(g, k); ++g, --k)
305     {
306         if (GameBoard->GetState(g, k) == Check)
307         {
308             fl = true;
309             break;
310         }
311         else
312             p++;
313     }
314
315     if (fl&&p>0)
316     {
317         s = 0;
318         while ((s <= p) && !(Chk))
```

```
319             {
320                 GameBoard->SetState(x + s, y - s, Check);
321                 s++;
322             }
323             if(Chk)
324                 return true;
325             else
326                 go = true;
327         }
328 
329         p = 0;
330 
331         return go;
332 }
333 
334 bool Game::Check()
335 {
336         bool emp = false;
337         for (int i = 0; i < Bsize; i++)
338             for (int j = 0; j < Bsize; j++)
339                 if (GameBoard->GetState(i, j) == Empty)
340                     emp = true;
341         if (!emp) return emp;
342         bool pl1 = CheckPlayerGo(Player1);
343         bool pl2 = CheckPlayerGo(Player2);
344         if (pl1 || pl2)
345             return true;
346         if ((!pl1) && (!pl2))
347             return false;
348 
349 }
350 
351 bool Game::CheckPlayerGo(GameState Player)
352 {
353         bool pl = false;
354         for (int i = 0; i < Bsize; i++)
355             for (int j = 0; j < Bsize; j++)
356                 if (CheckAndGo(Player, i, j, true))
357                     pl = true;
358         return pl;
359 }
```

```
1  #pragma once
2  #include "State.h"
3  #include <iostream>
4  class Dot
5  {
6  private:
7
8      int row, colum;           // [U+FFFD][U+FFFD][U+FFFD]
       ↪ [U+FFFD][U+FFFD]     State _state;    //
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]public:
9      Dot(int x = 0, int y = 0, State state = Empty)
10     {
11         colum = x;
12         row = y;
13         _state = state;
14     }
15     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] x
       ↪ [U+FFFD][U+FFFD][U+FFFD]- [U+FFFD][U+FFFD]void SetCol(int x);
16     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] y
       ↪ [U+FFFD][U+FFFD][U+FFFD]- [U+FFFD][U+FFFD]void SetRow(int y);
17     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]void SetState(State state);
18     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
       ↪ State GetState();
19     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] x
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]  int GetCol();
20     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD] y
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]  int GetRow();
21     // [U+FFFD][U+FFFD][U+FFFD][U+FFFD] [U+FFFD][U+FFFD][U+FFFD][U+FFFD]
       ↪ [U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
       ↪ void Print();
22 };
```

```
1
2  #include "Dot.h"
3
4  void Dot::SetCol(int x)
5  {
6      colum = x;
7  }
8
9  void Dot::SetRow(int y)
10 {
11     row = y;
12 }
13
14 void Dot::SetState(State state)
15 {
16     _state = state;
17 }
18
19 State Dot::GetState()
20 {
21     return _state;
22 }
23
24 int Dot::GetCol()
25 {
26     return colum;
27 }
```

```
28
29 int Dot::GetRow()
30 {
31     return row;
32 }
```

```
#pragma once
#include "State.h"
#include "Dot.h"

class Board
{
private:
    int b = 0;
    int w = 0;
    static const int Bsize = 8;
    Dot dots[Bsize][Bsize];
    GameState Gstate;
public:
    Board();
    ~Board() {}
    void PrintBoard();
    int BlackCount();
    int WhiteCount();
    State GetState(int x, int y);
    void SetState(int x, int y, State dotstate);
};
```

```

#include "Board.h"
#include <iostream>


Board::Board()
{
    for (int i = 0; i < Bsize; i++)
        for (int j = 0; j < Bsize; j++)
            dots[i][j].SetState(Empty);
    dots[3][3].SetState(White);
    dots[4][4].SetState(White);
    dots[4][3].SetState(Black);
    dots[3][4].SetState(Black);

}



int Board::BlackCount()
{
    b = 0;
    for (int i = 0; i < Bsize; i++)
        for (int j = 0; j < Bsize; j++)
            if (Black==dots[i][j].GetState())
                b++;
    return b;
}


int Board::WhiteCount()
{
    w = 0;
    for (int i = 0; i < Bsize; i++)
        for (int j = 0; j < Bsize; j++)
            if (White == dots[i][j].GetState())
                ++w;
    return w;
}
```

```
40
41 State Board::GetState(int x, int y)
42 {
43     return dots[x][y].GetState();
44 }
45
46 void Board::SetState(int x, int y, State DotState)
47 {
48     dots[x][y].SetState(DotState);
49 }
```