Файл main.cpp:

#include "mainwindow.h"

#include <QApplication>

int

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

{

QApplication a (argc, argv);

MainWindow menu;

menu.show ();

return a.exec ();

}

Файл bishop.h:

#ifndef BISHOP\_H

#define BISHOP\_H

#include "chesspiece.h"

class Bishop : public ChessPiece

{

public:

Bishop (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void moves ();

};

#endif // BISHOP\_H

Файл bishop.cpp:

#include "bishop.h"

#include "game.h"

#include <QDebug>

extern Game \*game;

Bishop::Bishop (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent) { setImage (); }

void

Bishop::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whitebishop.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackbishop.png"));

}

void

Bishop::moves ()

{

location.clear ();

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

QString team = this->getSide ();

// For upper Left

for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For upper right

for (int i = row - 1, j = col + 1; i >= 0 && j <= 7; i--, j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For downward right

for (int i = row + 1, j = col + 1; i <= 7 && j <= 7; i++, j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For downward left

for (int i = row + 1, j = col - 1; i <= 7 && j >= 0; i++, j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

}

Файл button.h:

#ifndef BUTTON\_H

#define BUTTON\_H

#include <QGraphicsRectItem>

#include <QGraphicsSceneMouseEvent>

class Button : public QObject, public QGraphicsRectItem

{

Q\_OBJECT

public:

Button (QString name, QGraphicsItem \*parent = NULL);

// public methods

void setFont (QFont const textFont);

// events

void mousePressEvent (QGraphicsSceneMouseEvent \*event);

void hoverEnterEvent (QGraphicsSceneHoverEvent \*event);

void hoverLeaveEvent (QGraphicsSceneHoverEvent \*event);

signals:

void clicked ();

private:

QGraphicsTextItem \*text;

};

#endif // BUTTON\_H

Файл button.cpp:

#include "button.h"

#include "qfont.h"

#include <QBrush>

#include <QGraphicsTextItem>

Button::Button (QString name, QGraphicsItem \*parent) : QGraphicsRectItem (parent)

{

// draw the rect

setRect (0, 0, 150, 50);

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (QColor::fromRgb (qRgb (61, 61, 61)));

setBrush (brush);

// draw Text

text = new QGraphicsTextItem (name, this);

int xPos = rect ().width () / 2 - text->boundingRect ().width () / 2 - 15;

int yPos = rect ().height () / 2 - text->boundingRect ().height () / 2;

text->setPos (xPos, yPos);

text->setDefaultTextColor (QColor::fromRgb (qRgb (247, 247, 247)));

// Allow responding to hover

setAcceptHoverEvents (true);

}

void

Button::setFont (QFont const textFont)

{

text->setFont (textFont);

}

void

Button::mousePressEvent (QGraphicsSceneMouseEvent \*event)

{

if (event)

emit clicked ();

}

void

Button::hoverEnterEvent (QGraphicsSceneHoverEvent \*event)

{

// change color

if (event)

{

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (QColor::fromRgb (qRgb (127, 127, 127)));

setBrush (brush);

}

}

void

Button::hoverLeaveEvent (QGraphicsSceneHoverEvent \*event)

{

// change color

if (event)

{

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (QColor::fromRgb (qRgb (61, 61, 61)));

setBrush (brush);

}

}

Файл chessboard.h:

#ifndef CHESSBOARD\_H

#define CHESSBOARD\_H

#include "chesspiece.h"

#include <QGraphicsRectItem>

class ChessBoard

{

public:

ChessBoard ();

// drawing public function

void drawBoxes (int x, int y);

void addChessPiece ();

void setUpWhite ();

void setUpBlack ();

void reset ();

private:

QList<ChessPiece \*> white;

QList<ChessPiece \*> black;

};

#endif // CHESSBOARD\_H

Файл chessboard.cpp:

#include "bishop.h"

#include "chessboard.h"

#include "chessbox.h"

#include "game.h"

#include "king.h"

#include "knight.h"

#include "pawn.h"

#include "queen.h"

#include "rook.h"

extern Game \*game;

ChessBoard::ChessBoard ()

{

setUpBlack ();

setUpWhite ();

}

void

ChessBoard::drawBoxes (int x, int y)

{

int SHIFT = 100;

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

ChessBox \*box = new ChessBox ();

game->collection[i][j] = box;

box->rowLoc = i;

box->colLoc = j;

box->setPos (x + SHIFT \* j, y + SHIFT \* i);

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

box->setOriginalColor (QColor::fromRgb (qRgb (247, 247, 247))); // white

else

box->setOriginalColor (QColor::fromRgb (qRgb (118, 118, 118))); // black

game->addToScene (box);

}

}

}

void

ChessBoard::addChessPiece ()

{

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

ChessBox \*box = game->collection[i][j];

if (i < 2)

{

static int k;

box->placePiece (black[k]);

game->alivePiece.append (black[k]);

game->addToScene (black[k++]);

}

if (i > 5)

{

static int h;

box->placePiece (white[h]);

game->alivePiece.append (white[h]);

game->addToScene (white[h++]);

}

}

}

}

void

ChessBoard::setUpWhite ()

{

ChessPiece \*piece;

for (int i = 0; i < 8; i++)

{

piece = new Pawn ("WHITE");

white.append (piece);

}

piece = new Rook ("WHITE");

white.append (piece);

piece = new Knight ("WHITE");

white.append (piece);

piece = new Bishop ("WHITE");

white.append (piece);

piece = new Queen ("WHITE");

white.append (piece);

piece = new King ("WHITE");

white.append (piece);

piece = new Bishop ("WHITE");

white.append (piece);

piece = new Knight ("WHITE");

white.append (piece);

piece = new Rook ("WHITE");

white.append (piece);

}

void

ChessBoard::setUpBlack ()

{

ChessPiece \*piece;

piece = new Rook ("BLACK");

black.append (piece);

piece = new Knight ("BLACK");

black.append (piece);

piece = new Bishop ("BLACK");

black.append (piece);

piece = new Queen ("BLACK");

black.append (piece);

piece = new King ("BLACK");

black.append (piece);

piece = new Bishop ("BLACK");

black.append (piece);

piece = new Knight ("BLACK");

black.append (piece);

piece = new Rook ("BLACK");

black.append (piece);

for (int i = 0; i < 8; i++)

{

piece = new Pawn ("BLACK");

black.append (piece);

}

}

void

ChessBoard::reset ()

{

int k = 0;

int h = 0;

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

ChessBox \*box = game->collection[i][j];

box->setHasChessPiece (false);

box->setChessPieceColor ("NONE");

box->currentPiece = NULL;

if (i < 2)

{

box->placePiece (black[k]);

black[k]->setIsPlaced (true);

black[k]->firstMove = true;

game->alivePiece.append (black[k++]);

}

if (i > 5)

{

box->placePiece (white[h]);

white[h]->setIsPlaced (true);

white[h]->firstMove = true;

game->alivePiece.append (white[h++]);

}

}

}

}

Файл chessbox.h :

#ifndef CHESSBOX\_H

#define CHESSBOX\_H

#include "chesspiece.h"

#include <QBrush>

#include <QGraphicsRectItem>

#include <QGraphicsSceneMouseEvent>

class ChessPiece;

class ChessBox : public QGraphicsRectItem

{

public:

// Constructor

ChessBox (QGraphicsItem \*parent = 0);

~ChessBox ();

// public member function

void mousePressEvent (QGraphicsSceneMouseEvent \*event);

void setColor (QColor color);

void placePiece (ChessPiece \*piece);

void resetOriginalColor ();

void setOriginalColor (QColor value);

bool getHasChessPiece ();

void setHasChessPiece (bool value, ChessPiece \*piece = 0);

void checkForCheck ();

QString getChessPieceColor ();

void setChessPieceColor (QString value);

int rowLoc;

int colLoc;

ChessPiece \*currentPiece;

private:

QBrush brush;

QColor originalColor;

bool hasChessPiece;

QString chessPieceColor;

};

#endif // CHESSBOX\_H

Файл chessbox.cpp:

#include "chessbox.h"

#include "game.h"

#include "king.h"

#include <QDebug>

extern Game \*game;

ChessBox::ChessBox (QGraphicsItem \*parent) : QGraphicsRectItem (parent)

{

// making the Square CHess Box

setRect (0, 0, 100, 100);

brush.setStyle (Qt::SolidPattern);

setZValue (-1);

setHasChessPiece (false);

setChessPieceColor ("NONE");

currentPiece = NULL;

}

ChessBox::~ChessBox () { delete this; }

void

ChessBox::mousePressEvent (QGraphicsSceneMouseEvent \*event)

{

// Deselecting counter part of ChessPiece

if (currentPiece == game->pieceToMove && currentPiece)

{

currentPiece->mousePressEvent (event);

return;

}

// if selected

if (game->pieceToMove)

{

// if same team

if (this->getChessPieceColor () == game->pieceToMove->getSide ())

return;

// removing the eaten piece

QList<ChessBox \*> movLoc = game->pieceToMove->moveLocation ();

// TO make sure the selected box is in move zone

int check = 0;

for (size\_t i = 0, n = movLoc.size (); i < n; i++)

{

if (movLoc[i] == this)

{

check++;

}

}

// if not prsent return

if (check == 0)

return;

// change the color back to normal

game->pieceToMove->decolor ();

// make the first move false applicable for pawn only

game->pieceToMove->firstMove = false;

// this is to eat or consume the enemy present inn the movable region

if (this->getHasChessPiece ())

{

this->currentPiece->setIsPlaced (false);

this->currentPiece->setCurrentBox (NULL);

game->placeInDeadPlace (this->currentPiece);

}

// changing the new stat and resetting the previous left region

game->pieceToMove->getCurrentBox ()->setHasChessPiece (false);

game->pieceToMove->getCurrentBox ()->currentPiece = NULL;

game->pieceToMove->getCurrentBox ()->resetOriginalColor ();

placePiece (game->pieceToMove);

game->pieceToMove = NULL;

// changing turn

game->changeTurn ();

checkForCheck ();

}

// Selecting couterpart of the chessPiece

else if (this->getHasChessPiece ())

{

this->currentPiece->mousePressEvent (event);

}

}

void

ChessBox::setColor (QColor color)

{

brush.setColor (color);

setBrush (color);

}

void

ChessBox::placePiece (ChessPiece \*piece)

{

piece->setPos (x () + 50 - piece->pixmap ().width () / 2,

y () + 50 - piece->pixmap ().width () / 2);

piece->setCurrentBox (this);

setHasChessPiece (true, piece);

currentPiece = piece;

}

void

ChessBox::resetOriginalColor ()

{

setColor (originalColor);

}

void

ChessBox::setOriginalColor (QColor value)

{

originalColor = value;

setColor (originalColor);

}

bool

ChessBox::getHasChessPiece ()

{

return hasChessPiece;

}

void

ChessBox::setHasChessPiece (bool value, ChessPiece \*piece)

{

hasChessPiece = value;

if (value)

setChessPieceColor (piece->getSide ());

else

setChessPieceColor ("NONE");

}

void

ChessBox::checkForCheck ()

{

int c = 0;

QList<ChessPiece \*> pList = game->alivePiece;

for (size\_t i = 0, n = pList.size (); i < n; i++)

{

King \*p = dynamic\_cast<King \*> (pList[i]);

if (p)

{

continue;

}

pList[i]->moves ();

pList[i]->decolor ();

QList<ChessBox \*> bList = pList[i]->moveLocation ();

for (size\_t j = 0, n = bList.size (); j < n; j++)

{

King \*p = dynamic\_cast<King \*> (bList[j]->currentPiece);

if (p)

{

if (p->getSide () == pList[i]->getSide ())

continue;

bList[j]->setColor (Qt::darkMagenta);

// if(pList[i]->getCurrentBox()->getChessPieceColor() == "WHITE")

// emit game->whiteWon();

// else

// emit game->blackWon();

// game->gameOver();

// pList[i]->getCurrentBox()->setColor(Qt::darkYellow);

if (!game->check->isVisible ())

game->check->setVisible (true);

else

{

bList[j]->resetOriginalColor ();

pList[i]->getCurrentBox ()->resetOriginalColor ();

// game->gameOver();

}

c++;

}

}

}

if (!c)

{

game->check->setVisible (false);

for (size\_t i = 0, n = pList.size (); i < n; i++)

pList[i]->getCurrentBox ()->resetOriginalColor ();

}

}

QString

ChessBox::getChessPieceColor ()

{

return chessPieceColor;

}

void

ChessBox::setChessPieceColor (QString value)

{

chessPieceColor = value;

}

Файл chesspiece.h:

#ifndef CHESSPIECE\_H

#define CHESSPIECE\_H

#include "chessbox.h"

#include <QGraphicsPixmapItem>

#include <QGraphicsSceneMouseEvent>

class ChessBox;

class ChessPiece : public QGraphicsPixmapItem

{

public:

ChessPiece (QString team = "", QGraphicsItem \*parent = 0);

void mousePressEvent (QGraphicsSceneMouseEvent \*event);

void setCurrentBox (ChessBox \*box);

ChessBox \*getCurrentBox ();

QString getSide ();

void setSide (QString value);

virtual void setImage () = 0;

bool getIsPlaced ();

void setIsPlaced (bool value);

QList<ChessBox \*> moveLocation ();

virtual void moves () = 0;

void decolor ();

bool firstMove;

bool boxSetting (ChessBox \*box);

protected:

ChessBox \*currentBox;

QString side;

bool isPlaced;

QList<ChessBox \*> location;

};

#endif // CHESSPIECE\_H

Файл chesspiece.cpp:

#include "chesspiece.h"

#include "game.h"

#include "king.h"

#include <QDebug>

extern Game \*game;

ChessPiece::ChessPiece (QString team, QGraphicsItem \*parent) : QGraphicsPixmapItem (parent)

{

side = team;

isPlaced = true;

firstMove = true;

}

void

ChessPiece::mousePressEvent (QGraphicsSceneMouseEvent \*event)

{

// Deselect

if (this == game->pieceToMove)

{

game->pieceToMove->getCurrentBox ()->resetOriginalColor ();

game->pieceToMove->decolor ();

game->pieceToMove = NULL;

return;

}

// if it is already consumed or not the respective color's turn

if ((!getIsPlaced ()) || ((game->getTurn () != this->getSide ()) && (!game->pieceToMove)))

return;

// selecting

if (!game->pieceToMove)

{

game->pieceToMove = this;

game->pieceToMove->getCurrentBox ()->setColor (Qt::darkGreen);

game->pieceToMove->moves ();

}

// Consuming counterPart of the CHessBox

else if (this->getSide () != game->pieceToMove->getSide ())

{

this->getCurrentBox ()->mousePressEvent (event);

}

}

void

ChessPiece::setCurrentBox (ChessBox \*box)

{

currentBox = box;

}

ChessBox \*

ChessPiece::getCurrentBox ()

{

return currentBox;

}

QString

ChessPiece::getSide ()

{

return side;

}

void

ChessPiece::setSide (QString value)

{

side = value;

}

bool

ChessPiece::getIsPlaced ()

{

return isPlaced;

}

void

ChessPiece::setIsPlaced (bool value)

{

isPlaced = value;

}

QList<ChessBox \*>

ChessPiece::moveLocation ()

{

return location;

}

void

ChessPiece::decolor ()

{

for (size\_t i = 0, n = location.size (); i < n; i++)

{

location[i]->resetOriginalColor ();

}

}

bool

ChessPiece::boxSetting (ChessBox \*box)

{

if (box->getHasChessPiece ())

{

King \*q = dynamic\_cast<King \*> (location.last ()->currentPiece);

if (q)

{

box->setColor (Qt::darkMagenta);

}

else

box->setColor (Qt::yellow);

return true;

}

else

location.last ()->setColor (Qt::green);

return false;

}

Файл game.h:

#ifndef GAME\_H

#define GAME\_H

#include "chesspiece.h"

#include <QGraphicsScene>

#include <QGraphicsView>

#include <chessboard.h>

class Game : public QGraphicsView

{

Q\_OBJECT

public:

// Constructors

Game (QWidget \*parent = 0);

// public Methods

void drawDeadHolder (int x, int y, QColor color);

void drawChessBoard ();

void displayDeadWhite ();

void displayDeadBlack ();

void placeInDeadPlace (ChessPiece \*piece);

void drawPawnMenu ();

// Scene Related

void addToScene (QGraphicsItem \*item);

void removeFromScene (QGraphicsItem \*item);

// getters/setters

ChessPiece \*pieceToMove;

QString getTurn ();

void setTurn (QString value);

void changeTurn ();

ChessBox \*collection[8][8];

QGraphicsTextItem \*check;

QList<ChessPiece \*> alivePiece;

void displayMainMenu ();

void gameOver ();

void removeAll ();

signals:

void GameOver ();

void whiteWon ();

void blackWon ();

public slots:

void start ();

private:

QGraphicsScene \*gameScene;

QList<ChessPiece \*> whiteDead;

QList<ChessPiece \*> blackDead;

QGraphicsRectItem \*deadHolder;

QGraphicsRectItem \*pawnMenu;

QString turn;

ChessBoard \*chess;

QList<QGraphicsItem \*> listG;

QGraphicsTextItem \*turnDisplay;

QGraphicsTextItem \*choosePawnText;

};

#endif // GAME\_H

Файл game.cpp:

#include "button.h"

#include "game.h"

#include "king.h"

#include <QDebug>

#include <QPixmap>

Game::Game (QWidget \*parent) : QGraphicsView (parent)

{

// Making the Scene

gameScene = new QGraphicsScene ();

gameScene->setSceneRect (0, 0, 1400, 900);

// Making the view

setFixedSize (1400, 900);

setHorizontalScrollBarPolicy (Qt::ScrollBarAlwaysOff);

setVerticalScrollBarPolicy (Qt::ScrollBarAlwaysOff);

setScene (gameScene);

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (QColor::fromRgb (qRgb (61, 61, 61)));

setBackgroundBrush (brush);

pieceToMove = NULL;

// display turn

turnDisplay = new QGraphicsTextItem ();

turnDisplay->setPos (width () / 2 - 100, 10);

turnDisplay->setZValue (1);

turnDisplay->setDefaultTextColor (QColor::fromRgb (qRgb (247, 247, 247)));

QFont fontTurn;

fontTurn.setFamily (QString::fromUtf8 ("Courier"));

fontTurn.setPointSize (18);

fontTurn.setBold (true);

fontTurn.setKerning (true);

turnDisplay->setFont (fontTurn);

turnDisplay->setPlainText ("Turn : WHITE");

// display Check

check = new QGraphicsTextItem ();

check->setPos (width () / 2 - 100, 860);

check->setZValue (4);

check->setDefaultTextColor (Qt::red);

QFont fontCheck;

fontCheck.setFamily (QString::fromUtf8 ("Courier"));

fontCheck.setPointSize (18);

fontCheck.setBold (true);

fontCheck.setKerning (true);

check->setFont (fontCheck);

check->setPlainText ("CHECK!");

check->setVisible (false);

setTurn ("WHITE");

}

void

Game::drawChessBoard ()

{

chess = new ChessBoard ();

drawDeadHolder (0, 0, QColor::fromRgb (qRgb (247, 247, 247)));

drawDeadHolder (1100, 0, QColor::fromRgb (qRgb (247, 247, 247)));

chess->drawBoxes (width () / 2 - 400, 50);

}

void

Game::displayDeadWhite ()

{

int SHIFT = 50;

int j = 0;

int k = 0;

for (size\_t i = 0, n = whiteDead.size (); i < n; i++)

{

if (j == 4)

{

k++;

j = 0;

}

whiteDead[i]->setPos (40 + SHIFT \* j++, 100 + SHIFT \* 2 \* k);

}

}

void

Game::displayDeadBlack ()

{

int SHIFT = 50;

int j = 0;

int k = 0;

for (size\_t i = 0, n = blackDead.size (); i < n; i++)

{

if (j == 4)

{

k++;

j = 0;

}

blackDead[i]->setPos (1140 + SHIFT \* j++, 100 + SHIFT \* 2 \* k);

}

}

void

Game::placeInDeadPlace (ChessPiece \*piece)

{

if (piece->getSide () == "WHITE")

{

whiteDead.append (piece);

King \*g = dynamic\_cast<King \*> (piece);

if (g)

{

check->setPlainText ("Black Won");

emit blackWon ();

gameOver ();

}

displayDeadWhite ();

}

else

{

blackDead.append (piece);

King \*g = dynamic\_cast<King \*> (piece);

if (g)

{

check->setPlainText ("White Won");

emit whiteWon ();

gameOver ();

}

displayDeadBlack ();

}

alivePiece.removeAll (piece);

}

void

Game::addToScene (QGraphicsItem \*item)

{

gameScene->addItem (item);

}

void

Game::removeFromScene (QGraphicsItem \*item)

{

gameScene->removeItem (item);

}

QString

Game::getTurn ()

{

return turn;

}

void

Game::setTurn (QString value)

{

turn = value;

}

void

Game::changeTurn ()

{

if (getTurn () == "WHITE")

setTurn ("BLACK");

else

setTurn ("WHITE");

turnDisplay->setPlainText ("Turn : " + getTurn ());

}

void

Game::start ()

{

drawChessBoard ();

for (size\_t i = 0, n = listG.size (); i < n; i++)

removeFromScene (listG[i]);

addToScene (turnDisplay);

QGraphicsTextItem \*whitePiece = new QGraphicsTextItem ();

whitePiece->setPos (70, 10);

whitePiece->setZValue (1);

whitePiece->setDefaultTextColor (QColor::fromRgb (qRgb (61, 61, 61)));

QFont fontWhitePiece;

fontWhitePiece.setFamily (QString::fromUtf8 ("Courier"));

fontWhitePiece.setPointSize (20);

fontWhitePiece.setBold (true);

fontWhitePiece.setKerning (true);

whitePiece->setFont (fontWhitePiece);

whitePiece->setPlainText ("WHITE PIECE");

addToScene (whitePiece);

QGraphicsTextItem \*blackPiece = new QGraphicsTextItem ();

blackPiece->setPos (1170, 10);

blackPiece->setZValue (1);

blackPiece->setDefaultTextColor (QColor::fromRgb (qRgb (61, 61, 61)));

QFont fontBlackPiece;

fontBlackPiece.setFamily (QString::fromUtf8 ("Courier"));

fontBlackPiece.setPointSize (20);

fontBlackPiece.setBold (true);

fontBlackPiece.setKerning (true);

blackPiece->setFont (fontBlackPiece);

blackPiece->setPlainText ("BLACK PIECE");

addToScene (blackPiece);

addToScene (check);

chess->addChessPiece ();

// drawPawnMenu();

}

void

Game::drawDeadHolder (int x, int y, QColor color)

{

deadHolder = new QGraphicsRectItem (x, y, 300, 900);

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (color);

deadHolder->setBrush (brush);

addToScene (deadHolder);

}

void

Game::drawPawnMenu ()

{

pawnMenu = new QGraphicsRectItem (300, 400, 800, 200);

QBrush brush;

brush.setStyle (Qt::SolidPattern);

brush.setColor (QColor::fromRgb (qRgb (247, 247, 247)));

pawnMenu->setBrush (brush);

addToScene (pawnMenu);

choosePawnText = new QGraphicsTextItem ();

turnDisplay->setPos (450, 420);

turnDisplay->setZValue (1);

turnDisplay->setDefaultTextColor (QColor::fromRgb (qRgb (61, 61, 61)));

QFont font;

font.setFamily (QString::fromUtf8 ("Courier"));

font.setPointSize (18);

font.setBold (true);

font.setKerning (true);

turnDisplay->setFont (font);

turnDisplay->setPlainText ("Choose a piece instead of a pawn");

Button \*bishopButton = new Button ("bishop");

Button \*knightButton = new Button ("knight");

Button \*queenButton = new Button ("queen");

Button \*rookButton = new Button ("rook");

int pxPos = 350;

int pyPos = 500;

bishopButton->setPos (pxPos, pyPos);

knightButton->setPos (pxPos + 175, pyPos);

queenButton->setPos (pxPos + 350, pyPos);

rookButton->setPos (pxPos + 525, pyPos);

font.setPointSize (12);

bishopButton->setFont (font);

knightButton->setFont (font);

queenButton->setFont (font);

rookButton->setFont (font);

connect (bishopButton, SIGNAL (clicked ()), this, SLOT (start ()));

addToScene (bishopButton);

addToScene (knightButton);

addToScene (queenButton);

addToScene (rookButton);

listG.append (bishopButton);

listG.append (knightButton);

listG.append (queenButton);

listG.append (rookButton);

}

void

Game::gameOver ()

{

// removeAll();

// setTurn("WHITE");

// alivePiece.clear();

// chess->reset();

this->close ();

emit GameOver ();

}

void

Game::removeAll ()

{

QList<QGraphicsItem \*> itemsList = gameScene->items ();

for (size\_t i = 0, n = itemsList.size (); i < n; i++)

{

if (itemsList[i] != check)

removeFromScene (itemsList[i]);

}

}

Файл king.h:

#ifndef KING\_H

#define KING\_H

#include "chesspiece.h"

class King : public ChessPiece

{

public:

King (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void findUnSafeLocation ();

void moves ();

};

#endif // KING\_H

Файл king.cpp:

#include "game.h"

#include "king.h"

#include "pawn.h"

extern Game \*game;

King::King (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent) { setImage (); }

void

King::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whiteking.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackking.png"));

}

void

King::moves ()

{

location.clear ();

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

QString team = this->getSide ();

if (col > 0 && row > 0 && !(game->collection[row - 1][col - 1]->getChessPieceColor () == team))

{ // up left

location.append (game->collection[row - 1][col - 1]);

game->collection[row - 1][col - 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (col < 7 && row > 0 && !(game->collection[row - 1][col + 1]->getChessPieceColor () == team))

{ // up right

location.append (game->collection[row - 1][col + 1]);

game->collection[row - 1][col + 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (row > 0 && !(game->collection[row - 1][col]->getChessPieceColor () == team))

{ // up

location.append (game->collection[row - 1][col]);

game->collection[row - 1][col]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (row < 7 && !(game->collection[row + 1][col]->getChessPieceColor () == team))

{ // down

location.append (game->collection[row + 1][col]);

game->collection[row + 1][col]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (col > 0 && !(game->collection[row][col - 1]->getChessPieceColor () == team))

{ // left

location.append (game->collection[row][col - 1]);

game->collection[row][col - 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (col < 7 && !(game->collection[row][col + 1]->getChessPieceColor () == team))

{ // right

location.append (game->collection[row][col + 1]);

game->collection[row][col + 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (col > 0 && row < 7 && !(game->collection[row + 1][col - 1]->getChessPieceColor () == team))

{ // down left

location.append (game->collection[row + 1][col - 1]);

game->collection[row + 1][col - 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

if (col < 7 && row < 7 && !(game->collection[row + 1][col + 1]->getChessPieceColor () == team))

{ // down right

location.append (game->collection[row + 1][col + 1]);

game->collection[row + 1][col + 1]->setColor (Qt::green);

if (location.last ()->getHasChessPiece ())

{

location.last ()->setColor (Qt::yellow);

}

}

// findUnSafeLocation();

}

void

King::findUnSafeLocation ()

{

QList<ChessPiece \*> pList = game->alivePiece;

for (size\_t i = 0, n = pList.size (); i < n; i++)

{

if (pList[i]->getSide () != this->getSide ())

{

QList<ChessBox \*> bList = pList[i]->moveLocation ();

for (size\_t j = 0, n = bList.size (); j < n; j++)

{

Pawn \*c = dynamic\_cast<Pawn \*> (pList[i]);

if (c)

continue;

for (size\_t k = 0, n = location.size (); k < n; k++)

{

if (bList[j] == location[k])

{

location[k]->setColor (Qt::blue);

}

}

}

}

}

}

Файл knight.h:

#ifndef KNIGHT\_H

#define KNIGHT\_H

#include "chesspiece.h"

class Knight : public ChessPiece

{

public:

Knight (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void moves ();

};

#endif // KNIGHT\_H

Файл knight.cpp:

#include "game.h"

#include "knight.h"

extern Game \*game;

Knight::Knight (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent) { setImage (); }

void

Knight::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whiteknight.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackknight.png"));

}

void

Knight::moves ()

{

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

int i, j;

QString team = this->getSide ();

// There are total 8 places where a night can move

// 1st up up left

i = row - 2;

j = col - 1;

if (i >= 0 && j >= 0 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 2nd up up right

i = row - 2;

j = col + 1;

if (i >= 0 && j <= 7 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 3rd down down left

i = row + 2;

j = col - 1;

if (i <= 7 && j >= 0 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 4th down down right

i = row + 2;

j = col + 1;

if (i <= 7 && j <= 7 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 5th left left up

i = row - 1;

j = col - 2;

if (i >= 0 && j >= 0 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 6th left left down

i = row + 1;

j = col - 2;

if (i <= 7 && j >= 0 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 7th right right up

i = row - 1;

j = col + 2;

if (i >= 0 && j <= 7 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

// 8th right right down

i = row + 1;

j = col + 2;

if (i <= 7 && j <= 7 && (game->collection[i][j]->getChessPieceColor () != team))

{

location.append (game->collection[i][j]);

if (location.last ()->getHasChessPiece ())

location.last ()->setColor (Qt::yellow);

else

location.last ()->setColor (Qt::green);

}

}

Файл mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui

{

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow (QWidget \*parent = nullptr);

~MainWindow ();

private slots:

void on\_buttonEXIT\_clicked ();

void on\_buttonPVP\_clicked ();

public slots:

void whiteWON ();

void blackWON ();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

Файл mainwindow.cpp

#include "game.h"

#include "mainwindow.h"

#include "ui\_mainwindow.h"

Game \*game;

MainWindow::MainWindow (QWidget \*parent) : QMainWindow (parent), ui (new Ui::MainWindow)

{

ui->setupUi (this);

}

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

void

MainWindow::on\_buttonEXIT\_clicked ()

{

this->close ();

}

void

MainWindow::whiteWON ()

{

this->setStyleSheet (QString::fromUtf8 ("background-image:url(E:/kursovaya3sem/myProject/media/"

"whitewon.png); background-position: center; "));

}

void

MainWindow::blackWON ()

{

this->setStyleSheet (QString::fromUtf8 ("background-image:url(E:/kursovaya3sem/myProject/media/"

"blackwon.png); background-position: center; "));

}

void

MainWindow::on\_buttonPVP\_clicked ()

{

if (game != nullptr)

{

game->close ();

}

this->ui->buttonPVP->hide ();

this->hide ();

game = new Game ();

game->show ();

game->start ();

QObject::connect (game, SIGNAL (GameOver ()), this, SLOT (show ()));

QObject::connect (game, SIGNAL (whiteWon ()), this, SLOT (whiteWON ()));

QObject::connect (game, SIGNAL (blackWon ()), this, SLOT (blackWON ()));

}

Файл pawn.h:

#ifndef PAWN\_H

#define PAWN\_H

#include "chesspiece.h"

class Pawn : public ChessPiece

{

public:

Pawn (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void moves ();

private:

};

#endif // PAWN\_H

Файл pawn.cpp:

#include "game.h"

#include "king.h"

#include "pawn.h"

#include <QDebug>

#include <typeinfo>

extern Game \*game;

Pawn::Pawn (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent)

{

setImage ();

firstMove = true;

}

void

Pawn::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whitepawn.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackpawn.png"));

}

void

Pawn::moves ()

{

location.clear ();

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

if (this->getSide () == "WHITE")

{

if (col > 0 && row > 0

&& game->collection[row - 1][col - 1]->getChessPieceColor () == "BLACK")

{

location.append (game->collection[row - 1][col - 1]);

boxSetting (location.last ());

}

if (col < 7 && row > 0

&& game->collection[row - 1][col + 1]->getChessPieceColor () == "BLACK")

{

location.append (game->collection[row - 1][col + 1]);

boxSetting (location.last ());

}

if (row > 0 && (!game->collection[row - 1][col]->getHasChessPiece ()))

{

location.append (game->collection[row - 1][col]);

boxSetting (location.last ());

if (firstMove && !game->collection[row - 2][col]->getHasChessPiece ())

{

location.append (game->collection[row - 2][col]);

boxSetting (location.last ());

}

}

}

else

{

if (col > 0 && row < 7

&& game->collection[row + 1][col - 1]->getChessPieceColor () == "WHITE")

{ // left

location.append (game->collection[row + 1][col - 1]);

boxSetting (location.last ());

}

if (col < 7 && row < 7

&& game->collection[row + 1][col + 1]->getChessPieceColor () == "WHITE")

{ // right

location.append (game->collection[row + 1][col + 1]);

boxSetting (location.last ());

}

if (row < 7 && (!game->collection[row + 1][col]->getHasChessPiece ()))

{

location.append (game->collection[row + 1][col]);

boxSetting (location.last ());

if (firstMove && !game->collection[row + 2][col]->getHasChessPiece ())

{

location.append (game->collection[row + 2][col]);

boxSetting (location.last ());

}

}

}

}

Файл queen.h:

#ifndef QUEEN\_H

#define QUEEN\_H

#include "chesspiece.h"

class Queen : public ChessPiece

{

public:

Queen (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void moves ();

};

#endif // QUEEN\_H

Файл queen.cpp:

#include "game.h"

#include "queen.h"

extern Game \*game;

Queen::Queen (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent) { setImage (); }

void

Queen::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whitequeen.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackqueen.png"));

}

void

Queen::moves ()

{

location.clear ();

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

QString team = this->getSide ();

// For up

for (int i = row - 1, j = col; i >= 0; i--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

// For Down

for (int i = row + 1, j = col; i <= 7; i++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For left

for (int i = row, j = col - 1; j >= 0; j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

// For Right

for (int i = row, j = col + 1; j <= 7; j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

// For upper Left

for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For upper right

for (int i = row - 1, j = col + 1; i >= 0 && j <= 7; i--, j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For downward right

for (int i = row + 1, j = col + 1; i <= 7 && j <= 7; i++, j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For downward left

for (int i = row + 1, j = col - 1; i <= 7 && j >= 0; i++, j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

}

Файл rook.h:

#ifndef ROOK\_H

#define ROOK\_H

#include "chesspiece.h"

class Rook : public ChessPiece

{

public:

Rook (QString team, QGraphicsItem \*parent = 0);

void setImage ();

void moves ();

};

#endif // ROOK\_H

Файл rook.cpp:

#include "game.h"

#include "rook.h"

extern Game \*game;

Rook::Rook (QString team, QGraphicsItem \*parent) : ChessPiece (team, parent) { setImage (); }

void

Rook::setImage ()

{

if (side == "WHITE")

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/whiterook.png"));

else

setPixmap (QPixmap ("E:/kursovaya3sem/myProject/media/blackrook.png"));

}

void

Rook::moves ()

{

location.clear ();

int row = this->getCurrentBox ()->rowLoc;

int col = this->getCurrentBox ()->colLoc;

QString team = this->getSide ();

// For up

for (int i = row - 1, j = col; i >= 0; i--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

// For Down

for (int i = row + 1, j = col; i <= 7; i++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

{

break;

}

}

}

// For left

for (int i = row, j = col - 1; j >= 0; j--)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

// For Right

for (int i = row, j = col + 1; j <= 7; j++)

{

if (game->collection[i][j]->getChessPieceColor () == team)

{

break;

}

else

{

location.append (game->collection[i][j]);

if (boxSetting (location.last ()))

break;

}

}

}