Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

Отчет о проектной работе по курсу

«Основы информатики и программирования»

Разработка приложения «Шахматы»

Выполнил:
студента 1 курса группы 22103

А. А. Каличенко _____
подпись

Петрозаводск — 2021

Содержание

Введение

Цель проекта: Реализовать приложение "Шахматы"с графическим интерфейсом на языке C++.

Задачи проекта:

1. Сделать интерфейс начального экрана
2. Создать стартовый экран игры
3. Внедрить логику передвижения шахматных фигур
4. Сделать интерфейс выхода из приложения при победе

1 Требования к приложению

Итоговое приложение должно:

1. Являться аналогом настольной игры "Шахматы"
2. В эту игру могут играть 2 игрока на одном ПК
3. Иметь начальный(для старта партии) и конечный экран(для выхода из приложения)
4. Реализовано, в первую очередь, для игроков, которые недавно начали играть в шахматы
5. Быть интуитивно для пользователей

2 Проектирование приложения

Основные модули приложения:

main.cpp - запуск приложения; показываем окно; выводим Главное меню

game.h game.cpp - C++ класс, который помогает взаимодействовать с полем(старт, выход, отрисовка фигур).

Основные функции этого класса:

start() - Выполняем расстановку шахмат, Рисуем доску, Убираем ненужные элементы;

displayMainMenu() - выводит интерфейс начального экрана;

displayGameEnd() - выводит интерфейс выхода;

changeTurn() - меняем ход;

eatPiece() - съедаем фигуру;

bishop.cpp bishop.h - C++ класс, который отвечает за логику ходьбы;

Основные функции этого класса:

setImage() - выбираем и ставим изображение фигуры;

moves() - отвечает за передвижение

fdchessBox.h fdchessBox.cpp - C++ класс, который помогает взаимодействовать пользователю с игрой

Основные функции этого класса:

mousePressEvent() - функция, которая реализует взаимодействие пользователя и игрока(Обновляем положения фигуры, Переход хода, Выбираем фигуру, Отмена выбора фигуры, создаём cells)

fdboardChess.h fdboardChess.cpp - C++ класс, который реализует логику визуализации игры

Основные функции этого класса:

ChessBoard(), setUpBlack(), setUpWhite() - функции, которые дают каждой фигуре флаг(1 - Черные; 0 - Белые)

drawBoxes() - Рисуем доску(окрашиваем в разные цвета клетки поочереди)

addChessPiece() - Добавляем фигуры (до 2й строчки - белые; после 5й - чёрные)

3 Реализация приложения

Технологии разработки:

Языки: C++, Qml

Библиотеки:

<QGraphicsView>

<QGraphicsScene>

<QGraphicsRectItem>

<QGraphicsSceneMouseEvent>

<QPixmap>

<QApplication>

<QBrush>

Приведу несколько фрагментов из кода:

Этот код выводит на главный экран название игры "CHESS"и кнопку "PLAY"

```
{
    // Название игры
    QGraphicsTextItem *titleText = new QGraphicsTextItem("CHESS");
    QFont titleFont("Times", 70);
    titleText->setFont(titleFont);
    titleText->setDefaultTextColor(Qt::darkBlue);

    int xPos = width()/2 - titleText->boundingRect().width()/2;
    int yPos = 150;
    titleText->setPos(xPos,yPos);
    addToScene(titleText);
    objs.append(titleText);

    // Кнопка Играть
    Button * playButton = new Button("PLAY");
    playButton->setPos(width()/2 - playButton->boundingRect().width()/2,
                       height()/2 - playButton->boundingRect().height()/2);
    connect(playButton,SIGNAL(clicked()) , this , SLOT(start())); // по нажатии
    //на кнопку вызываем start()

    addToScene(playButton); // Добавляем на сцену
    objs.append(playButton);// Добавляем объекты, чтобы их удалить
}
```

Отмена выбора фигуры(если мы её выбрали), выбор фигуры, съедание фигуры и обновление поожение фигуры:

```
{
    // Отмена выбора фигуры
    if(this == game->pieceToMove){
        game->pieceToMove->getCurrentBox()->resetOriginalColor();
        game->pieceToMove->decolor();
        game->pieceToMove = NULL;
       return;
    }
    ...
    // Выбираем фигуру
    if(!game->pieceToMove){
        game->pieceToMove = this;
        game->pieceToMove->getCurrentBox()->setColor(Qt::lightGray);
        game->pieceToMove->moves();
    }
    else if(this->getSide() != game->pieceToMove->getSide()){
        this->getCurrentBox()->mousePressEvent(event);
    ...
    // Съедаем фигуры
    if(this->getHasChessPiece()){
        this->currentPiece->setCurrentBox(NULL);
        game->eatPiece(this->currentPiece);
    }
    // Обновляем положения фигуры
    game->pieceToMove->getCurrentBox()->setHasChessPiece(false);
    game->pieceToMove->getCurrentBox()->currentPiece = NULL;
    game->pieceToMove->getCurrentBox()->resetOriginalColor();
    placePiece(game->pieceToMove);
    }
}
```

Приведу пример реализации кода для взадимодействия с фигурой "Слон":

1. Ставим изображение

```
void Bishop::setImage()
{
    if(side == "1")
        setPixmap(QPixmap(":Pices/bishop1.png"));
    else
        setPixmap(QPixmap(":Pices/bishop.png"));
}
```

2. Логика ходьбы (ходит по диагонали) (здесь только для диагонали Верх Лево)

```
void Bishop::moves()
{
    location.clear();
    int row = this->getCurrentBox()->rowLoc;
    int col = this->getCurrentBox()->colLoc;
    QString place = this->getSide();

    /// Для каждого свободной клетки по диагонали

    //Верх Лево
    for(int i = row - 1, j = col - 1; i >= 0 && j >=0; i--,j--) {
        if(game->collection[i][j]->getChessPieceColor() == place)
            break;
        else{
            location.append(game->collection[i][j]);
            if(boxSetting(location.last()))
                break;
        }
    }
}
```

Заключение

Разработано приложение "Шахматы"

Код опубликован на Github: `https://github.com/Skand21/Chess_Qt`

Были получены знания и опыт разработки приложения

Реализованы основные идеи:

1. Сделать интерфейс начального экрана (рис. 1)
2. Создать стартовый экран игры (рис. 2)
3. Внедрить логику передвижения шахматных фигур (рис. 3)
4. Сделать интерфейс выхода из приложения при победе (рис. 4)

Рис. 1: Начальный экран игры

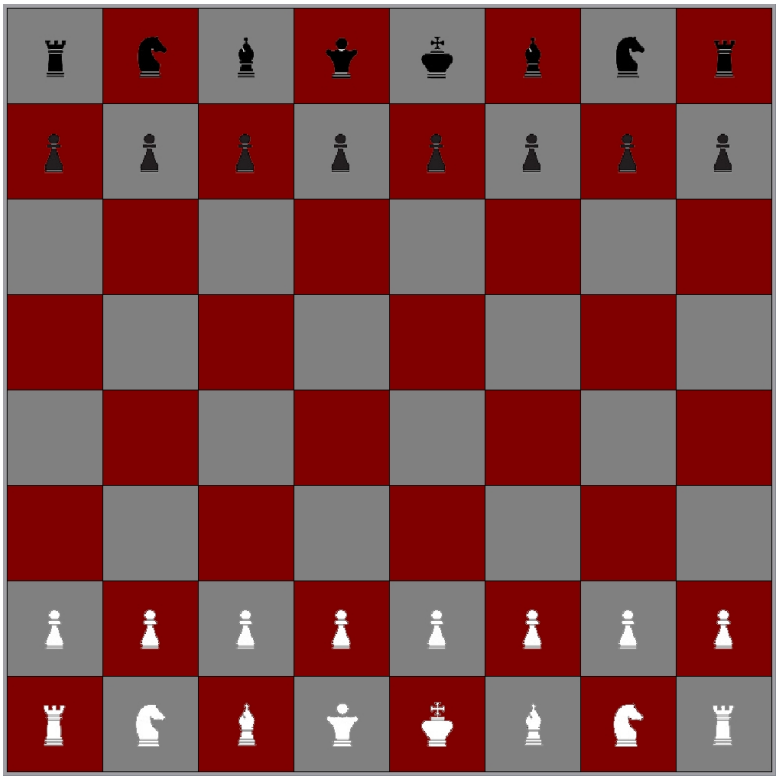

Рис. 2: Интерфейс начального экрана

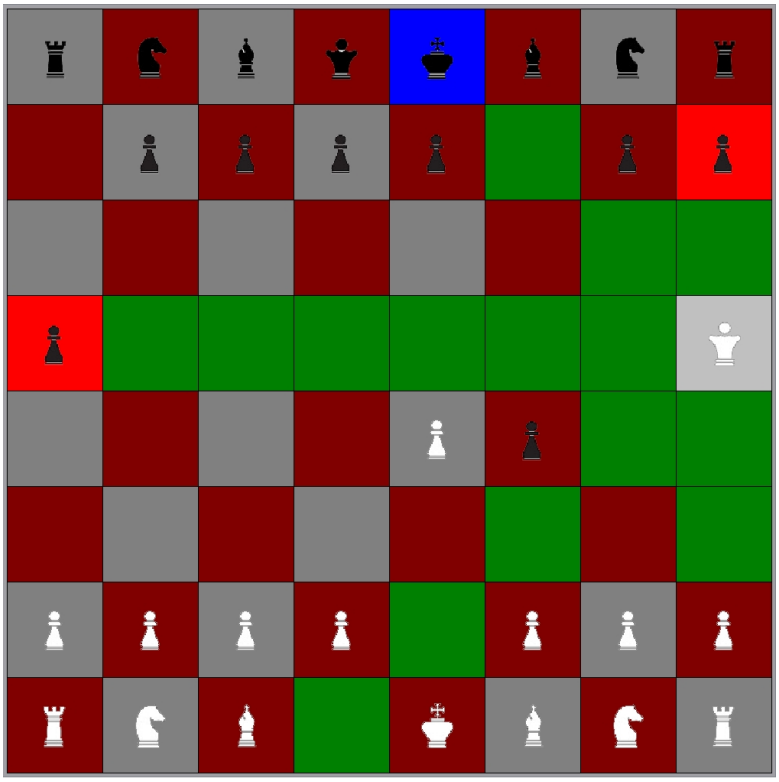

Рис. 3: Передвижение шахматных фигур

Рис. 4: Интерфейс выхода из приложения при победе одной из сторон