**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное

учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направление информационные системы и технологии

Отчет

По курсовой работе

по дисциплине

**«ПРОГРАММИРОВАНИЕ ЧАСТЬ 2 »**

**разработка игры на объектно-ориентированных языках программирования.**

Выполнил:

Студент группы 8И23 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Изов

Проверил:

Доцент ОИТ ИШИТР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Иванов

Томск 2024

Оглавление

[Задание 3](#_Toc155802576)

[Введение 4](#_Toc155802577)

[Архитектура 5](#_Toc155802578)

[Результаты 7](#_Toc155802579)

[Выводы 11](#_Toc155802580)

[Использованная литература 12](#_Toc155802581)

# Задание

Разработать игру на объектно-ориентированном языке программирования.

Технические требования:

* Программа должна использовать архитектуру: MVC, MVVM или др.
* Программа должна использовать хотя бы 1 шаблон (паттерн): порождающие паттерны, паттерны поведения, структурные паттерны.
* Для разработки необходимо использовать любой ООП язык программирования: С#, java, C++ и т.д
* Запрещается использовать любые игровые движки: Unity, Cocos, Unreal, Godot и т.д.

# Введение

Конечным результатом курсовой работы является игра крестики-нолики. Приложение будет реализовывать все правила настольной игры, без изменений игрового процесса. Играть можно будет только против компьютера, у которого есть 3 режима сложности. В игре будет реализована система подсказок действий. А также, возможность менять цветовую гамму и статистика побед/поражений для каждого уровня сложности.

Цель работы: создать приложение «Компьютерная игра: крестики-нолики».

Задачи:

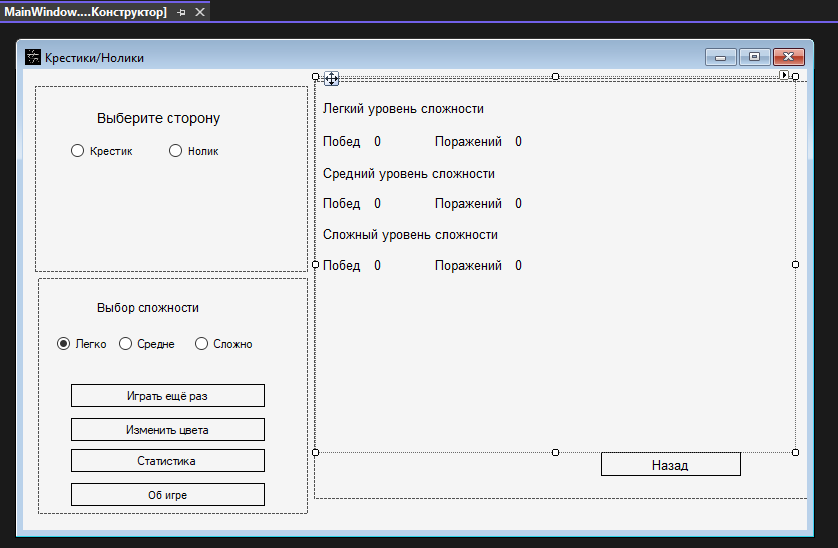
1. Обзор возможных инструментов разработки
2. Обзор возможных архитектур приложения
3. Разработка модели приложения
4. Программирование приложения

* Создание интерфейса
* Программирование алгоритмов хода компьютера
* Программирование действий игрока
* Программирование

1. Тестирование
2. Доработка приложения
3. Выпуск приложения

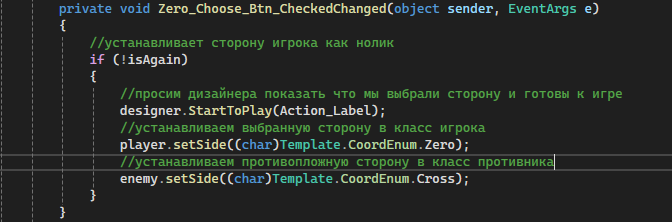
Архитектура

Для разработки приложения Я выбрал архитектуру MVC. Так как она удобна для приложения на фреймворке WinForms. Эта архитектура подразумевает три компонента. Model – логика приложения. View – вид приложения. Controller – взаимодействие логики с внешним видом приложения. В моем приложении за View отвечает файл MainWindows.cs [конструктор] и StyleWindow[Конструктор] . В режиме конструктора можно добавлять компоненты в приложение без написания программного кода.



*Рис.1. View MVC*

Файл MainWindow.cs, который отвечает за обработку событий является Controller. Так как он связывает интерфейс программы с классами реализующими логику. Внутри обработчиков событий, мы можем обращаться к классу представляющему Model. И на основе его ответа изменять интерфейс. Передача данных происходит через аргументы у вызванных методов.



*Рис.2. Пример Controller*

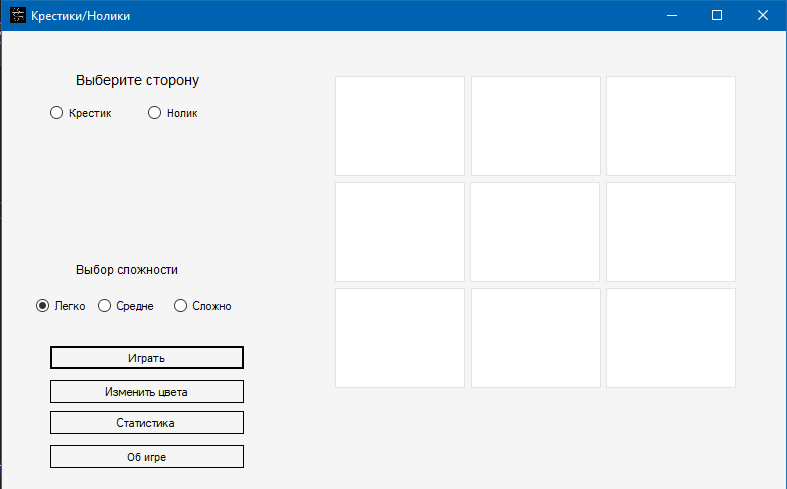
Model – его реализует сразу несколько классов. Designer – класс, выполненный по паттерну Singleton делегирует на себя обязанности по внесению изменений в интерфейс. Класс template, реализующий паттерн Template Pattern является родительским классом для Player и Enemy. Класс Player реализует действия игрока, а класс Enemy действия компьютера соответственно.

Выбор шаблонного паттерна охарактеризован схожестью функционала классов Player и Enemy. Для каждого класса игровой ход содержит в себе выбор подходящей клетки на поле и проверка, является ли эта клетка победной, или же приведет к ничьей.

Класс Designer реализует паттерн одиночка, потому что мы стараемся избежать двойственных указаний по изменению внешнего вида. Это может случиться из-за того, что у нас есть 2 окна в приложении. И каждое окно требует определенных изменений внешнего вида.

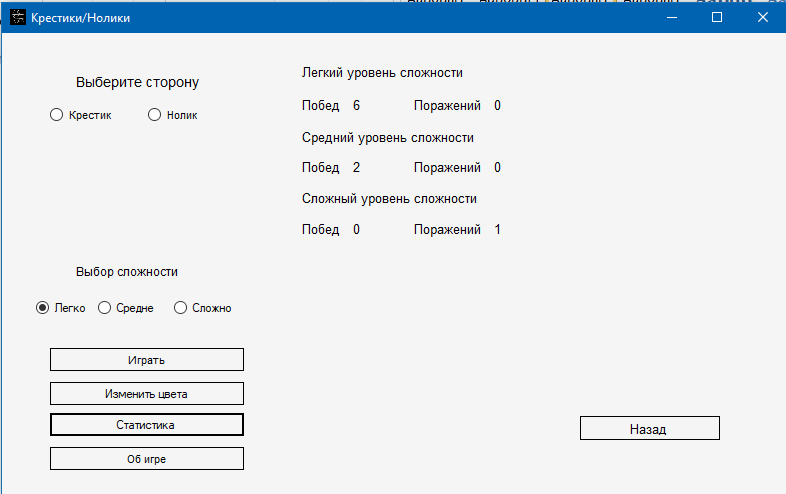
# Результаты

Сразу после запуска нас встречает интуитивно понятное окно.



*Рис.3. Результат работы программы*

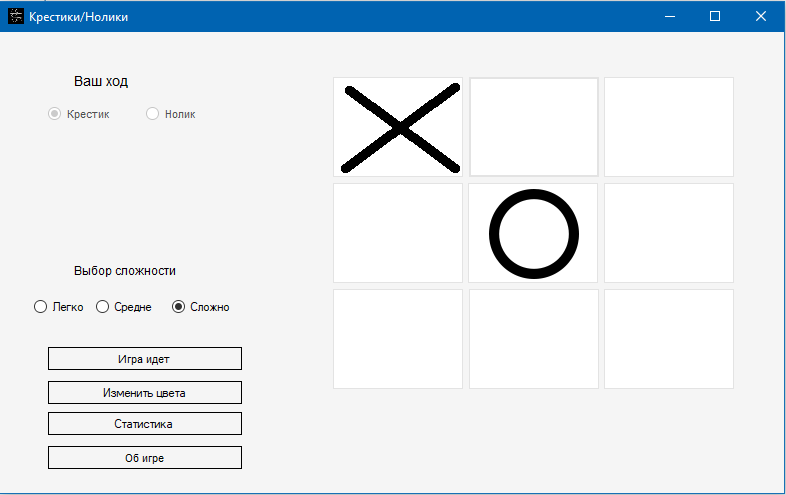
Можно посмотреть правила игры по нажатию кнопки «Об игре». Можно увидеть статистику, соответственно после нажатия кнопки «Статистика». После закрытия приложения данные по количеству побед и поражений не пропадают, а сохраняются.



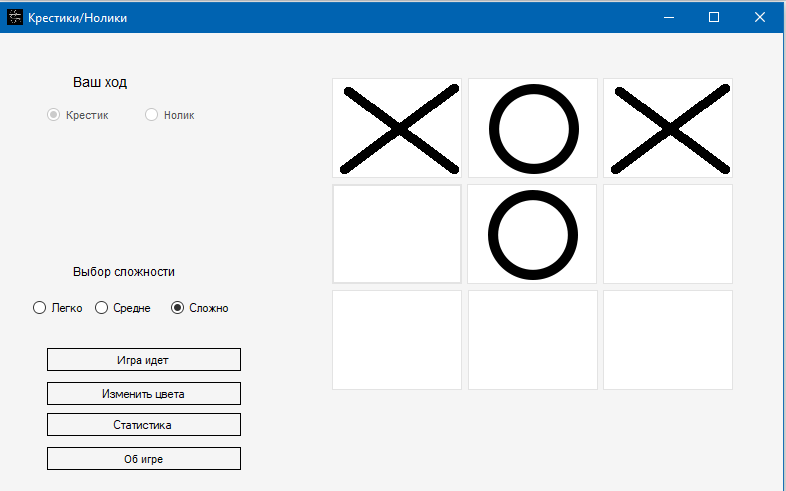
*Рис.4. Окно статистики*

Можно выбрать сторону, за которую хочет играть пользователь. Так же, он может выбрать уровень сложности. По умолчанию стоит легкий уровень сложности.

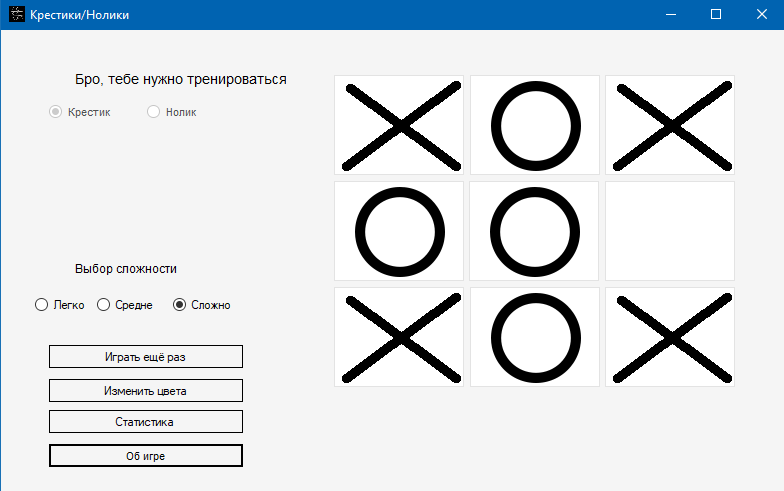
Выбрав, например, сложный уровень, игрок столкнется с алгоритмом, который ищет победные ходы противника и мешает ему их выполнить. Это можно увидеть на следующих скриншотах игры. Если игрок сходит в правый верхний угол (где стоит курсор). То у него 1 ход до победы. Алгоритм найдет это поле и помешает игроку победить так просто.



*Рис.5.1. Сложный режим игры*



*Рис.5.2. Сложный режим игры*



*Рис.5.3 Сложный режим игры*

Что же, мне не удалось победить, получаем совет от компьютера и можем попытать удачу ещё раз.

В окне статистики рис.4. мы можем отследить наше К/Д. Данные хранятся в файле настроек. Доступ к ним можно получить в любой точке программы используя класс Properties.Settings.Default. Так же, через класс Properties можно обратиться к ресурсному файлу Resources, в котором удобно хранить изображения и другие файлы.

# Выводы

В ходе выполнения курсовой работы были изучены паттерны проектирования. Из 23 существующих паттернов было выбрано 2. Паттерн Singleton и паттерн Template method. Паттерн одиночки позволяет нам быть уверенным, что мы работаем всегда с одним и тем же экземпляром класса. Он необходим для замены статического класса. Шаблонный метод необходим для упрощения и уменьшения кода в том случае, если у нас есть классы со схожим функционалом. Это достигается механизмом наследования.

Была изучена архитектура MVC. Она заключается в разделении кода логики и кода интерфейса. Взаимодействие их осуществляется в контроллере.

Был изучен фреймворк WinForms. Были изучены его возможности по работе с настройками приложения, ресурсными файлами.

Было изучено расширение Microsoft Visual Studio Installer Project 2022. Оно позволяет создать приложение установщик для нашего приложения крестики-нолики.

Был разработан алгоритм поиска победных ходов противника с целью помешать игроку победить. Алгоритм включал в себя общий член бесконечного ряда. Он был необходим для генерации на определенном шаге итерации цикла определенное число. Такой метод необходим для оптимизации поиска необходимой ячейки игрового поля.

# Использованная литература

1. Паттерны проектирования: [Электронный ресурс]. URL: <https://metanit.com/sharp/patterns/> (Дата обращения 10.11.2023).
2. Рукводство по WindowsForms: [Электронный ресурс]. URL: <https://metanit.com/sharp/windowsforms/> (Дата обращения 15.11.2023).
3. Настройки приложения: [Электронный ресурс]. URL: <https://youtu.be/n1t6qogQx4Q?si=ZubVcgohX731q3tY> (Дата обращения 27.11.2023).
4. Создание установочного файла: [Электронный ресурс]. URL: <https://youtu.be/aYcYamTI6ys?si=cfXRmt2qwcV6llCC> (Дата обращения 02.12.2023).