Федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра «Информационные системы и технологии»

Направление подготовки/ специальность: 09.03.02 «Информационные системы и технологии»

ОТЧЕТ

по проектной практике

Студент: Кривоносова Варвара Владимировна_____ Группа: __241-337____

Место прохождения практики: Московский Политех, кафедра «ИиИТ»

Отчет принят с оценкой _____ Дата _____

Руководитель практики: Меньшикова Наталия Павловна_____

Москва 2025

Отчет по вариативной части проектной (учебной) практике

Вариативная часть задания включает в себя:

1. Практическая реализация технологии:

- Выполнить все задачи базовой части.
- Выбрать любую технологию (Build a C#/WPF RPG). Проведите исследование: изучите, как создать выбранную технологию с нуля, воспроизведите практическую часть.
- Создать подробное описание в формате Markdown, включающее:
    - Последовательность действий по исследованию предметной области и созданию технологии.
    - Напишите техническое руководство по созданию этой технологии, ориентированное на начинающих.
    - Включите в руководство:
    - Пошаговые инструкции.
    - Примеры кода.
- Создание технического руководства по созданию проекта на выбранную тему;
- Создание видеопрезентации выполненной работы;
- Написание документации проекта в формате Markdown;
- Подготовка финального отчёта.

Финальный отчет по разработке игры "Сапер" на WPF (C#)

Разработка игры "Сапер" на платформе WPF (Windows Presentation Foundation) с использованием языка C# представляла собой комплексный процесс, включавший несколько последовательных этапов.

На подготовительном этапе проводился детальный анализ оригинальной игры для понимания ключевых механик игрового процесса. Особое внимание уделялось изучению алгоритмов генерации случайного расположения мин на поле, методов подсчета количества мин в соседних клетках и принципов рекурсивного открытия пустых областей. Параллельно разрабатывалась архитектура будущего приложения, где центральное место заняла система обработки пользовательских действий через механизмы событий мыши.

Основной этап разработки начался с создания многослойного графического интерфейса, в котором в качестве фундамента игрового поля был выбран элемент UniformGrid. Этот выбор обусловлен его способностью автоматически поддерживать равномерное распределение клеток, что значительно упростило процесс визуализации и последующей обработки

игровых событий. Каждая клетка игрового поля была реализована как элемент управления Button, что не только обеспечило естественную обработку пользовательских действий, но и позволило легко интегрировать сложную систему визуальной обратной связи. Для визуального отображения состояния клеток применялась сложная система оформления, включающая различные цвета для цифр, специальные символы для мин и флагов.

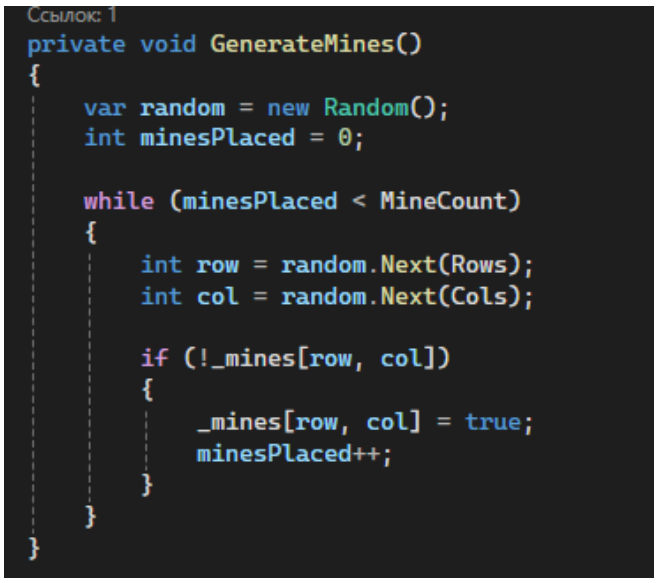
```
private void UpdateButtonAppearance(int row, int col)
{
    Button button = _buttons[row, col];

    if (_flagged[row, col])
    {
        button.Content = "🚩";
        button.Background = Brushes.LightGray;
    }
    else if (!_revealed[row, col])
    {
        button.Content = "";
        button.Background = Brushes.LightGray;
    }
    else if (_mines[row, col])
    {
        button.Content = "💣";
        button.Background = Brushes.Red;
    }
    else
    {
        int count = _adjacentMines[row, col];
        button.Content = count > 0 ? count.ToString() : "";
        button.Background = Brushes.White;

        // Разные цвета для цифр
        switch (count)
        {
            case 1: button.Foreground = Brushes.Blue; break;
            case 2: button.Foreground = Brushes.Green; break;
            case 3: button.Foreground = Brushes.Red; break;
            case 4: button.Foreground = Brushes.DarkBlue; break;
            case 5: button.Foreground = Brushes.DarkRed; break;
            case 6: button.Foreground = Brushes.Teal; break;
            case 7: button.Foreground = Brushes.Black; break;
            case 8: button.Foreground = Brushes.Gray; break;
            default: button.Foreground = Brushes.Black; break;
        }
```

Рисунок 1 Фрагмент кода метода UpdateButtonAppearance для визуализации состояния клеток в игре 'Сапер'

Важнейшим компонентом игры стал механизм генерации мин, основанный на использовании криптографически стойкого генератора случайных чисел. Этот алгоритм гарантирует равномерное распределение мин по полю без образования скоплений. Не менее значимой оказалась реализация системы подсчета соседних мин, где для каждой клетки анализируются восемь соседних позиций с учетом граничных условий.



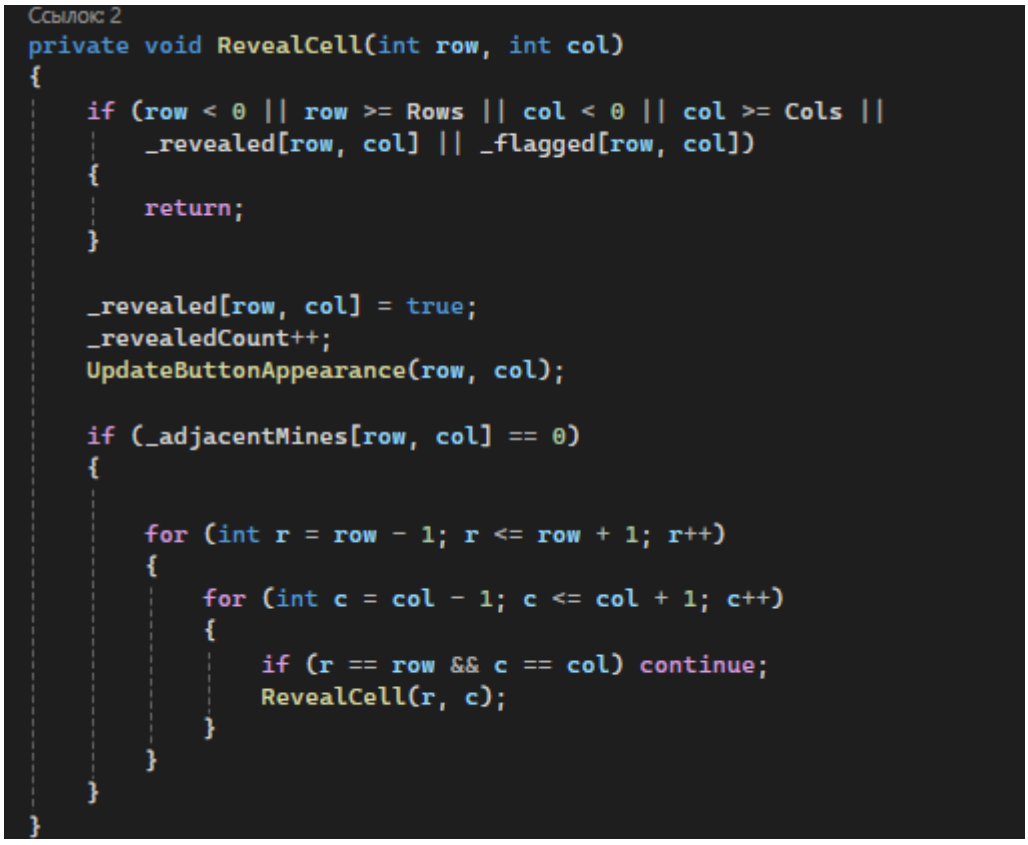
```
Ссылок: 1
private void GenerateMines()
{
    var random = new Random();
    int minesPlaced = 0;

    while (minesPlaced < MineCount)
    {
        int row = random.Next(Rows);
        int col = random.Next(Cols);

        if (!_mines[row, col])
        {
            _mines[row, col] = true;
            minesPlaced++;
        }
    }
}
```

Рисунок 2 Алгоритм генерации мин и подсчета соседних клеток в игре 'Сапер'

В обработчике таймера MainEventTimer была реализована основная игровая Особую сложность представляла реализация алгоритма рекурсивного открытия пустых областей. Этот механизм должен корректно обрабатывать все возможные сценарии, включая открытие клеток у границ поля и предотвращение бесконечной рекурсии.



```
Ссылок: 2
private void RevealCell(int row, int col)
{
    if (row < 0 || row >= Rows || col < 0 || col >= Cols ||
        _revealed[row, col] || _flagged[row, col])
    {
        return;
    }

    _revealed[row, col] = true;
    _revealedCount++;
    UpdateButtonAppearance(row, col);

    if (_adjacentMines[row, col] == 0)
    {

        for (int r = row - 1; r <= row + 1; r++)
        {
            for (int c = col - 1; c <= col + 1; c++)
            {
                if (r == row && c == col) continue;
                RevealCell(r, c);
            }
        }
    }
}
```

Рисунок 3 Рекурсивный алгоритм открытия клеток в игре 'Сапер' (метод RevealCell)

Для управления игровым процессом использовалась система обработки событий мыши, где левый клик отвечает за открытие клетки, а правый - за установку или снятие флага.

На этапе тестирования проводилась комплексная проверка всех игровых механик. Особое внимание уделялось пограничным случаям, таким как открытие клетки с миной в начале игры или установка флага на уже открытую клетку. Проверялась корректность работы таймера и системы подсчета оставшихся мин. Тестирование на различных конфигурациях оборудования подтвердило стабильность работы приложения.

Визуальная составляющая игры была тщательно проработана для создания удобного интерфейса. Использование различных цветов для цифр помогает игроку быстрее оценивать ситуацию на поле. Специальные символы в виде эмодзи делают игру более наглядной и привлекательной. Система подсчета очков и отображения времени игры реализована в верхней части окна, что соответствует принципам удобства пользовательского интерфейса.

Рисунок 4 Игровой интерфейс 'Сапера' с отображением поражения

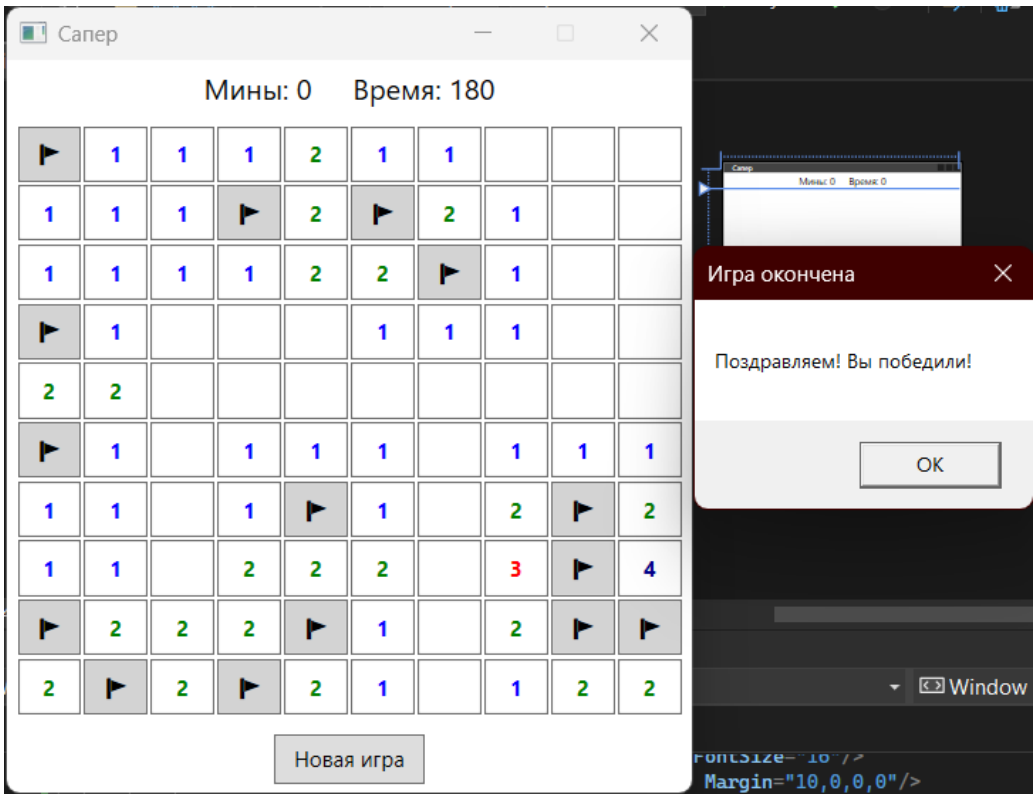


Рисунок 5 Игровой интерфейс 'Сапера' с отображением победы

В результате проведенной работы была создана полнофункциональная версия игры "Сапер", не только соответствующая всем канонам классической реализации, но и предлагающая дополнительные удобства для игроков. Приложение демонстрирует стабильную работу во всех тестовых сценариях, точное выполнение игровых правил и приятный визуальный стиль, соответствующий современным стандартам пользовательских интерфейсов.

В процессе разработки были успешно преодолены многочисленные технические сложности, связанные с особенностями работы WPF, обработкой пользовательского ввода и реализацией нетривиальной игровой логики. Полученный опыт наглядно демонстрирует эффективность выбранного технологического стека для создания сложных логических игр и интерактивных приложений.