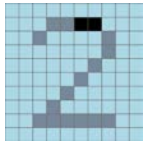
МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
Институт дополнительного образования
Высшая инженерная школа

Выпускная квалификационная работа
По программе профессиональной переподготовки
«Разработчик прикладного программного обеспечения (язык C#)»

«Разработка игры «Морской бой» с расширенными функциями»

Выполнил: Бондарь А.В.
Руководитель: Туральчук К.А.

Санкт-Петербург
2021



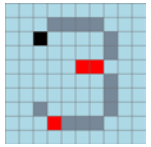
Цели и задачи

Цель ВКР:

Разработать игру «Морской бой» с расширенными функциями настройки.

Задачи:

1. Спроектировать структуру приложения с применением паттерна MVVM;
2. Реализовать приложение с применением технологии WPF;
3. Реализовать алгоритмы логики компьютера с разной сложностью;
4. Выполнить тестирование приложения.

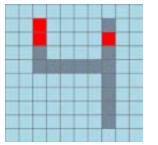


Ожидаемые результаты

В игре должна быть реализована следующая функциональность:

1. Возможность выбора режима игры: человек с человеком, человек с компьютером и компьютер с компьютером;
2. Возможность расстановки кораблей на поле в случайном порядке;
3. Возможность изменения размера игрового поля;
4. Возможность задания количества (плотности расстановки) кораблей на игровом поле;
5. Возможность изменения сложности игры при игре с компьютером;
6. Возможность проведения боёв между двумя разными реализациями логики компьютера для сравнения их результативности.
7. Ведение статистики по ранее сыгранным партиям.

Структура приложения

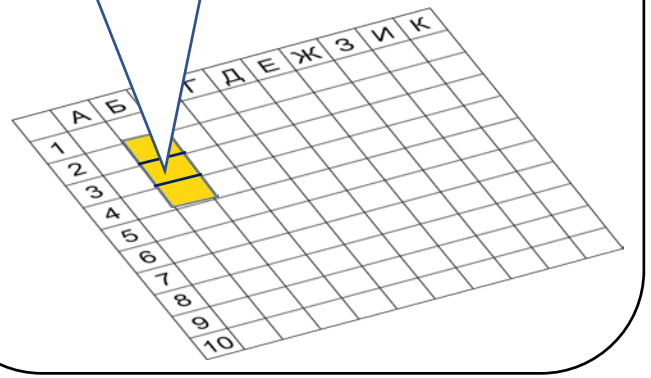


	View	View-Model	Model
Сектор настроек игры	**WPF «NewGameParams»:** Задание параметров для новой игры	**Класс «NewGameParamsViewModel»:** Передача информации о параметрах игры из окна выбора параметров в класс "GameWindowViewMode" с учетом диапазонов из класса "GameParams"	**Класс «GameParams»:** Хранит информацию о параметрах игры и о диапазонах их изменений
Сектор игрового процесса	**WPF-интерфейс:** Основное игровое окно и вспомогательные окна	**Класс «GameWindowViewModel»:** Взаимодействие между классами Model и оконным интерфейсом View, а также между другими классам из View-Model	**Класс «Fleet»:** Структура данных игры **Классы «ComputerLogicLevel_1...3»:** Логика игры компьютера **Класс «GameOrder»:** Регулирует порядок игры
Сектор статистики	**WPF «GameStatistics»:** Отображает статистическую информацию в реальном времени при игре компьютера с компьютером	**Класс «GameStatisticsViewModel»:** Управляет процессом сбора и сохранения статистики во время сбора игры	**Класс «GameStatistics»:** Собирает статистику об игре игроков во время игры **Класс «GameStatisticsUtils»:** Сохраняет и загружает статистическую информацию об игре разных игроков

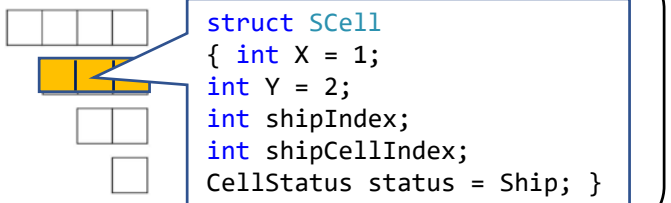
Структура для хранения данных: класс «Fleet»

Игровое поле SCell[,] arrField:

```
struct SCell
{ int X;
int Y;
int shipIndex = 2;
int shipCellIndex = 1;
CellStatus status = Ship; }
```



Игровой флот List<SCell[]> ships:

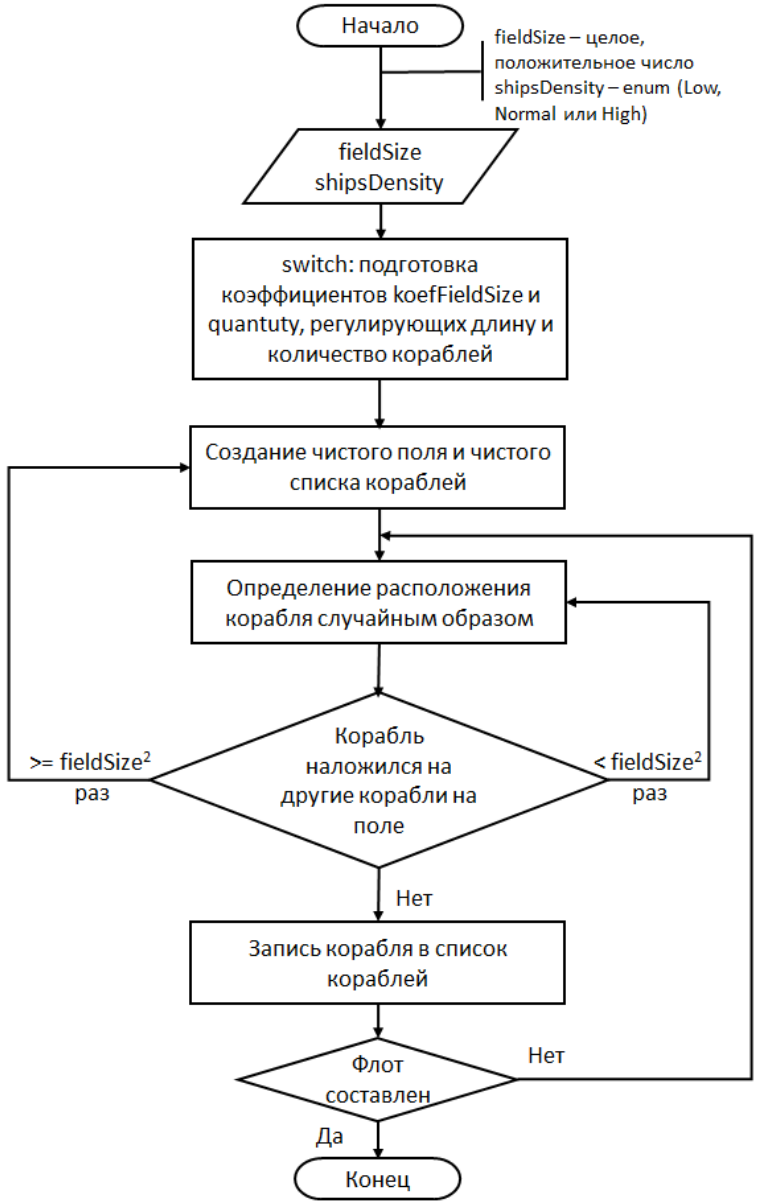


```
struct SCell
{ int X = 1;
int Y = 2;
int shipIndex;
int shipCellIndex;
CellStatus status = Ship; }
```

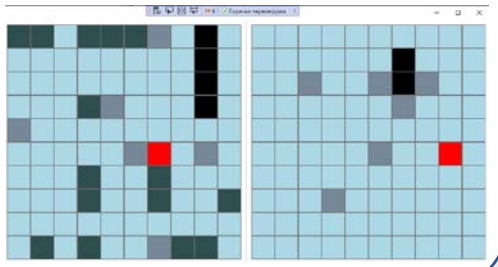
Возможные состояния клеток:

```
enum CellStatus : int
{
AlreadyActivated = -1,
Clear,
Ship,
Hit,
Destroyed,
RestAria,
Miss,
Cover
};
```

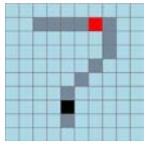
Алгоритм построения кораблей в случайном порядке:



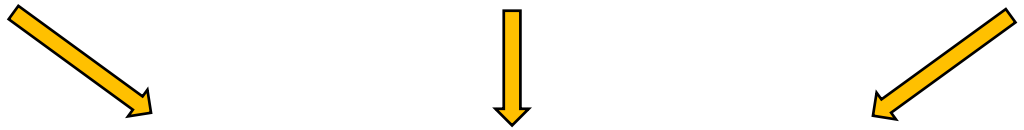
Реализация игрового процесса

	View	View-Model	Model
Сектор игрового процесса	**WPF "MoveTransition":** Информирует игроков о смене хода при игре двух человек **WPF "GameOver":** Информирует об окончании игры и объявляет победителя **WPF "GameWindow":** 	**Класс MoveTransitionViewModel:** Регулирует отображение окна, информирующего игроков о смене хода при игре двух человек **Класс «GameWindowViewModel»:** Взаимодействие между классами Model и оконным интерфейсом View, а также между другими классам из View-Model	**Класс «Fleet»:** Структура данных игры **Классы "ComputerLogicLevel_1...3":** Реализуют логику игры компьютера различной сложности. Анализируют и собирают информацию о поле и состоянии флота противника в процессе игры. Выбирают клетку для следующего удара. **Класс «GameOrder»:** Регулирует порядок игры. Определяет, кто должен делать следующий ход. Определяет окончание игры и победителя

Алгоритмы логики компьютера: Классы «ComputerLogicLevel_1…3»

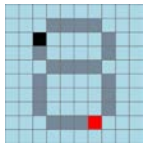


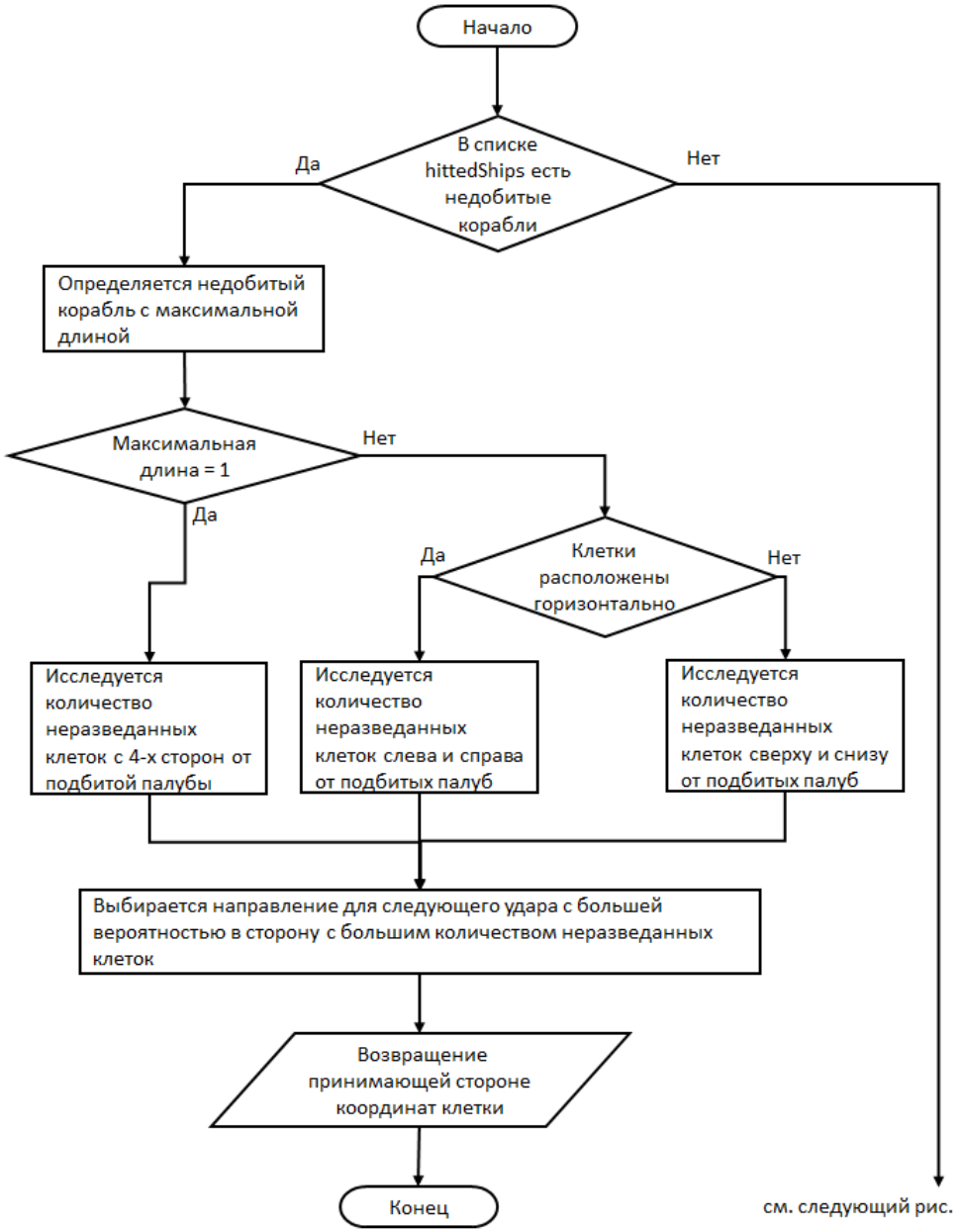
Определение клетки для следующего хода		
Easy	**Normal**	**Advanced**
• Если имеются раненные корабли, то они уничтожаются. • Если обнаруженные корабли подбиты, то клетка для удара определяется в случайном порядке. С вероятностью 1:10 компьютер ударит в пустую клетку рядом с уже подбитым кораблём.	• Если имеются раненные корабли, то направление для следующего удара выбирается с большей вероятностью в сторону с большим количеством неразведанных клеток. • Если обнаруженные корабли подбиты, то следующая клетка для удара определяется в случайном порядке.	• Если имеются раненные корабли, то направление для следующего удара выбирается с большей вероятностью в сторону с большим количеством неразведанных клеток. • Сперва ищется и уничтожается самый длинный корабль, потом корабль поменьше и т.д. • Оставшиеся однопалубные корабли в конце ищутся в случайном порядке.

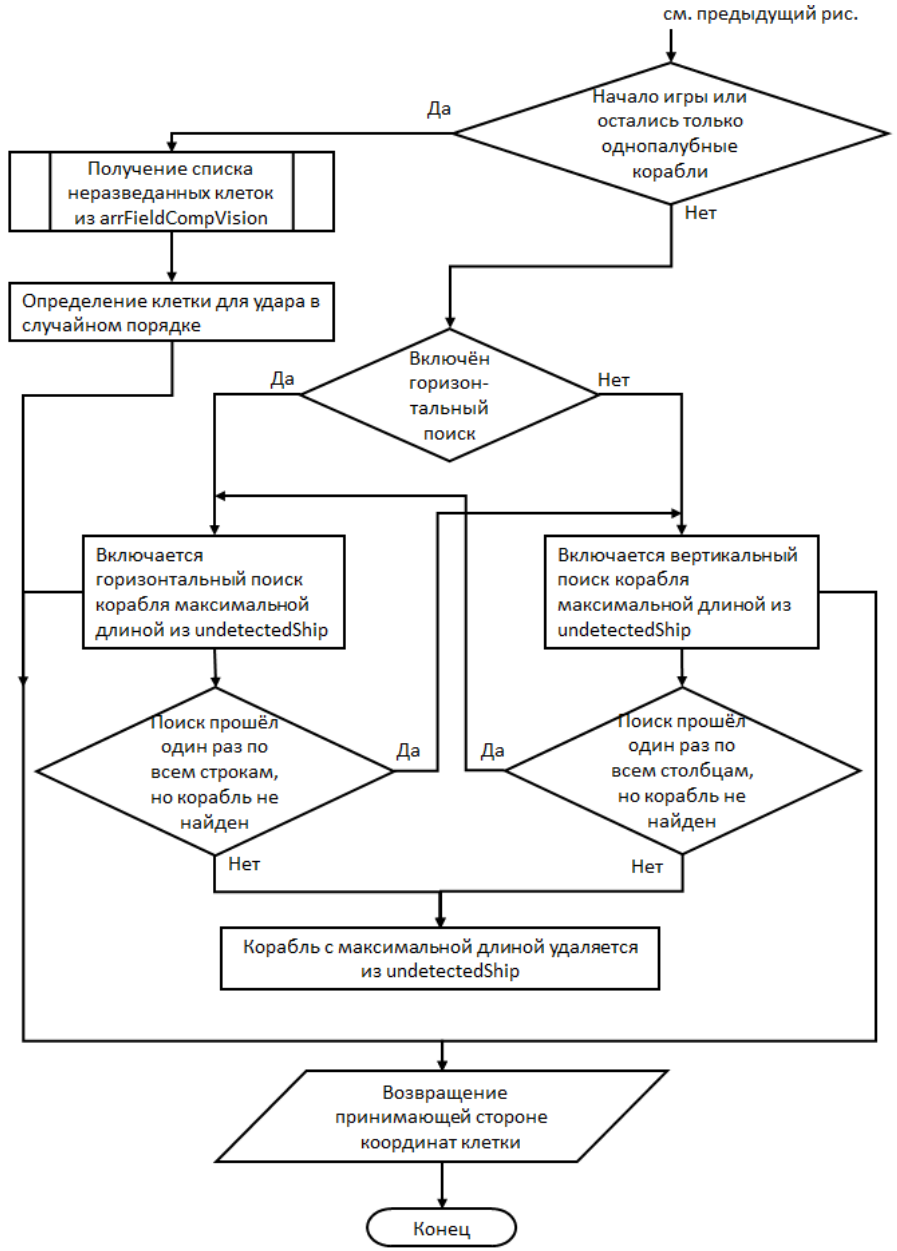


Корректировка представления о поле противника на основе сделанного хода

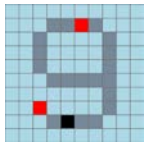
Упрощенный алгоритм метода GetNextCellForTurn класса ComputerLogicLevel_3 :





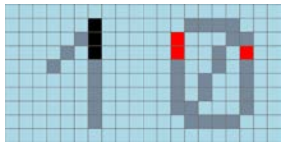


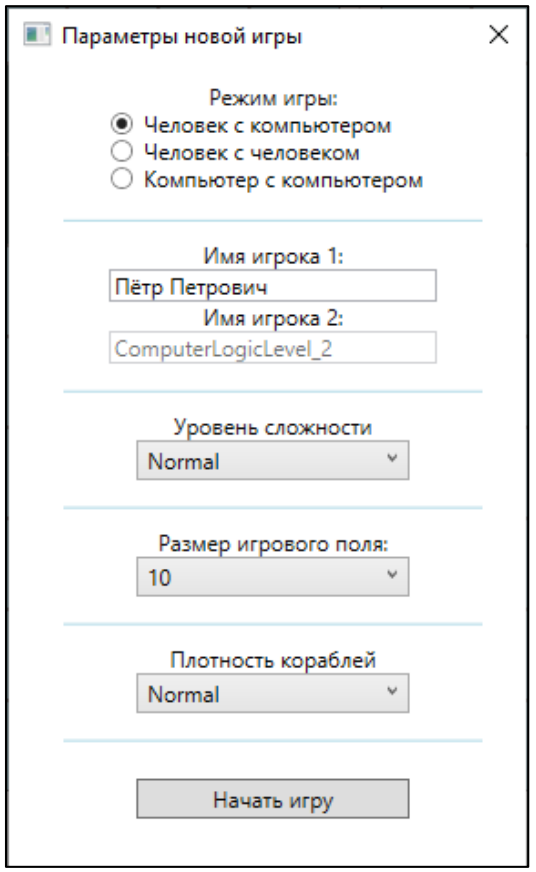
Управление параметрами игры

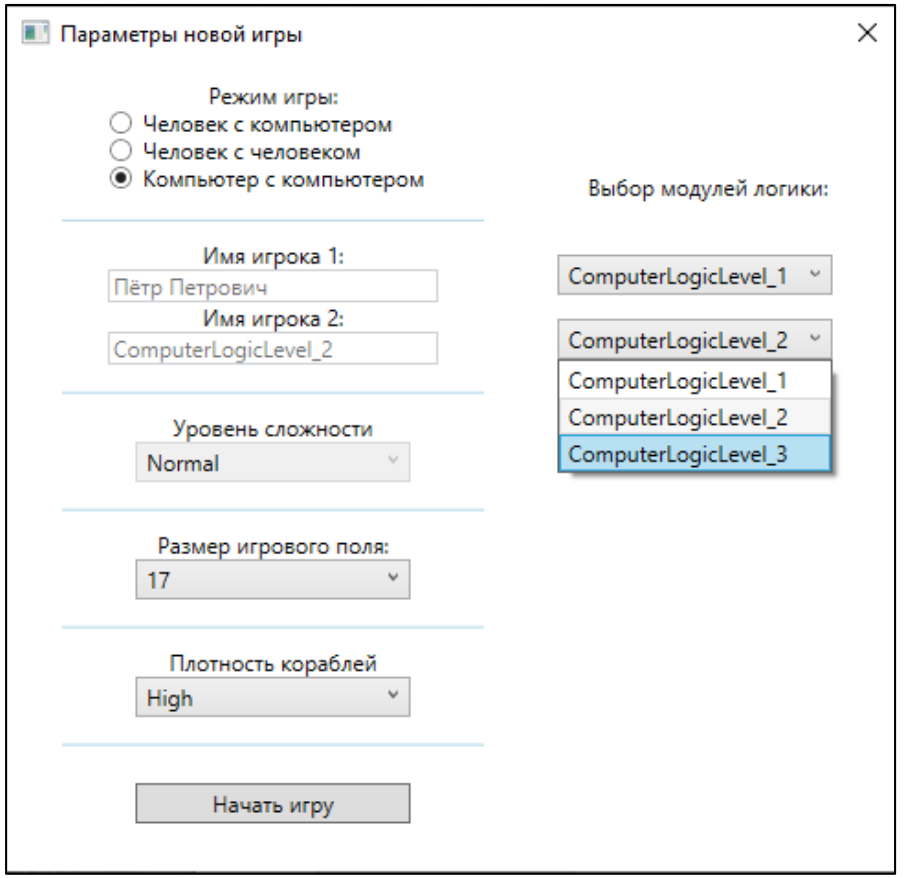


	View	View-Model	Model
Сектор настроек игры	**WPF «NewGameParams»:** Задание параметров для новой игры	**Класс «NewGameParamsViewModel»:** Передача информации о параметрах игры из окна выбора параметров в класс "GameWindowViewMode" с учетом диапазонов из класса "GameParams" Связь с сектором игрового процесса	**Класс «GameParams»:** Хранит информацию о параметрах игры и о диапазонах их изменений

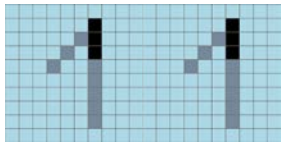
Управление параметрами игры





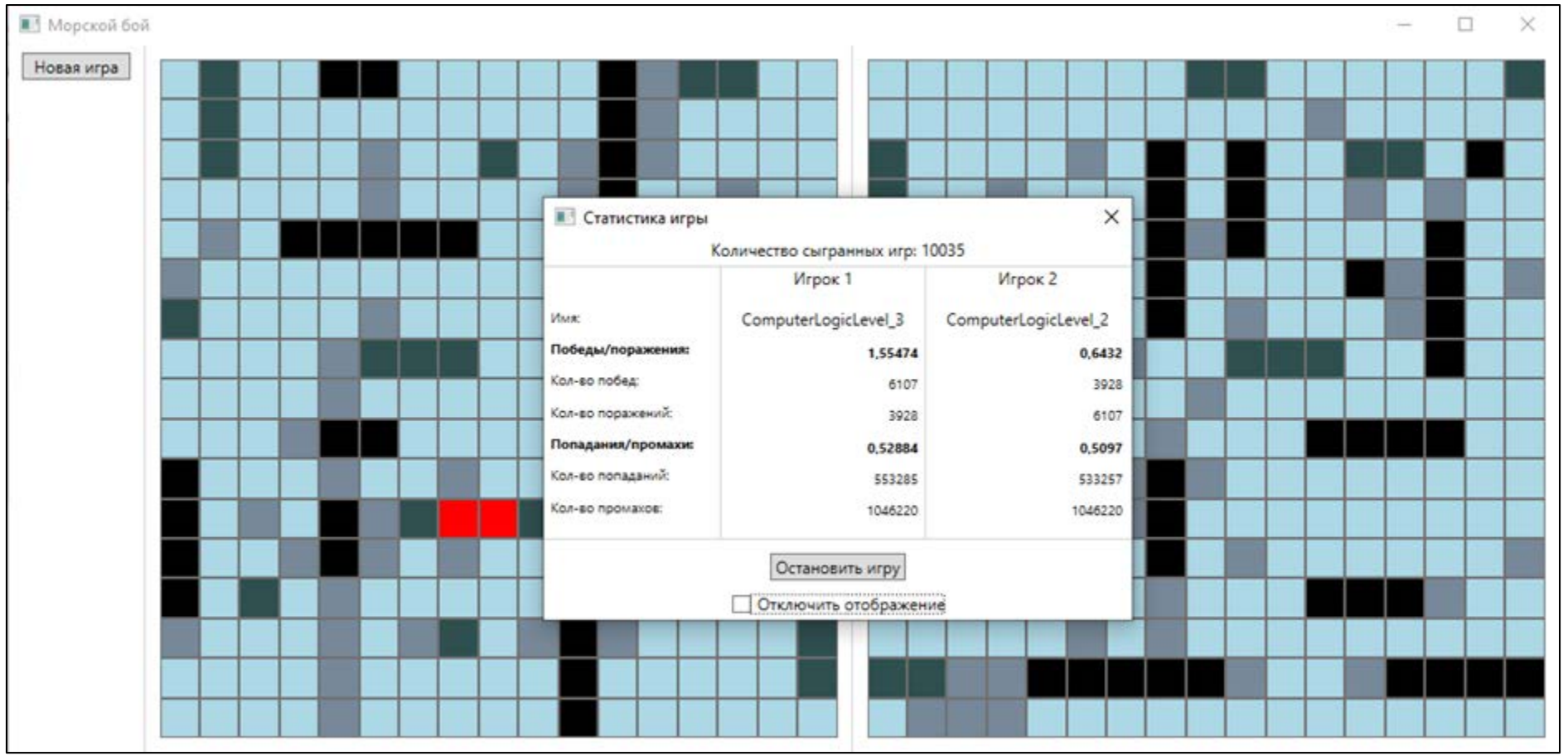


Работа со статистикой

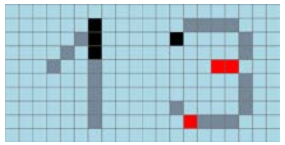


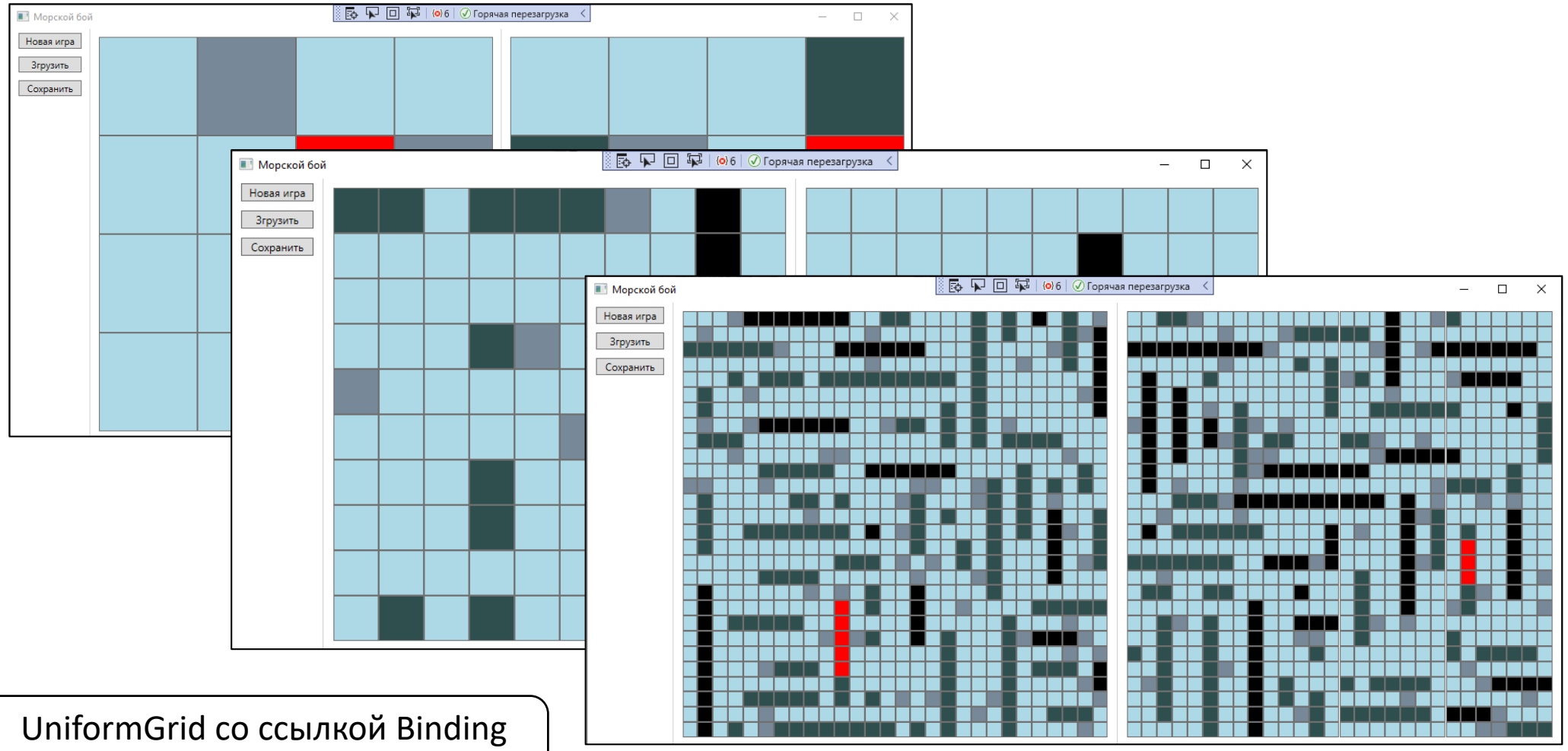
	View	View-Model	Model
Сектор статистики	**WPF «GameStatistics»:** Отображает статистическую информацию в реальном времени при игре компьютера с компьютером	Связь с сектором игрового процесса **Класс «GameStatisticsViewModel»:** Управляет процессом сбора и сохранения статистики во время сбора игры	**Класс «GameStatistics»:** Собирает статистику об игре игроков во время игры **Класс «GameStatisticsUtils»:** Сохраняет и загружает статистическую информацию об игре разных игроков

Работа со статистикой



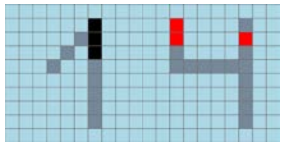
Игровое окно

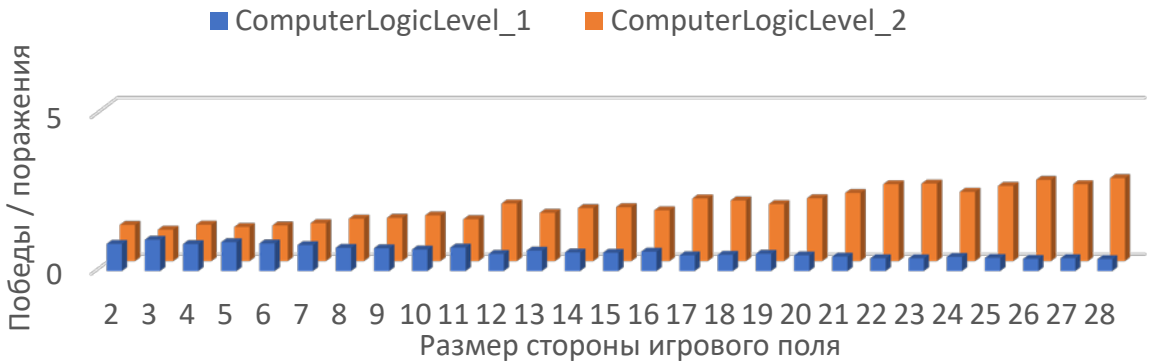




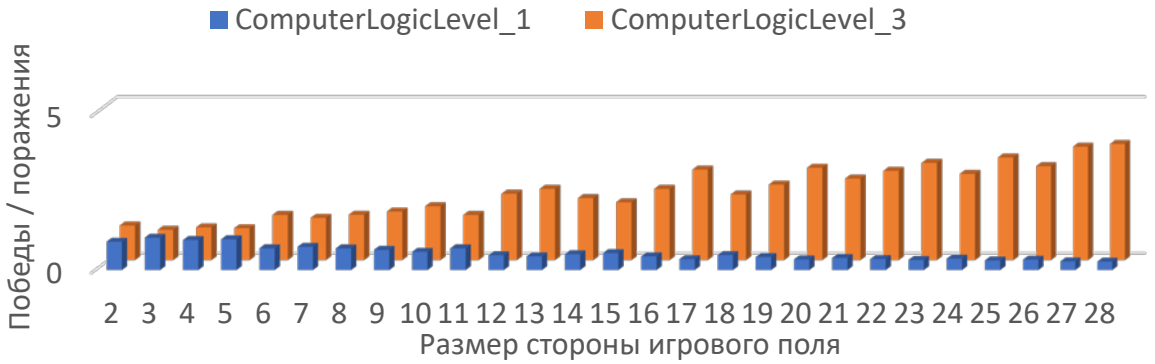
UniformGrid со ссылкой Binding
на ObservableCollection<Button>

Тестирование

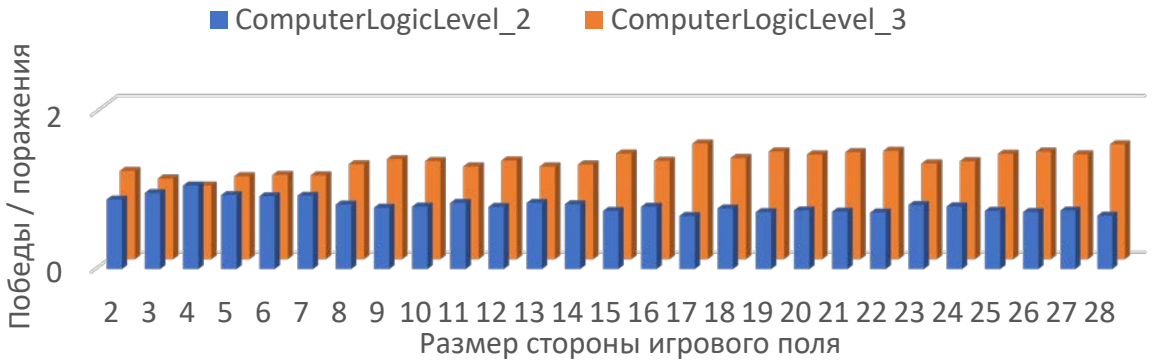




- Отношение количества побед к количеству поражений при игре класса ComputerLogicLevel_1 с ComputerLogicLevel_2 при плотности кораблей 10%

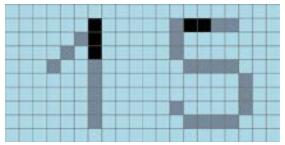


- Отношение количества побед к количеству поражений при игре класса ComputerLogicLevel_1 с ComputerLogicLevel_3 при плотности кораблей 10%

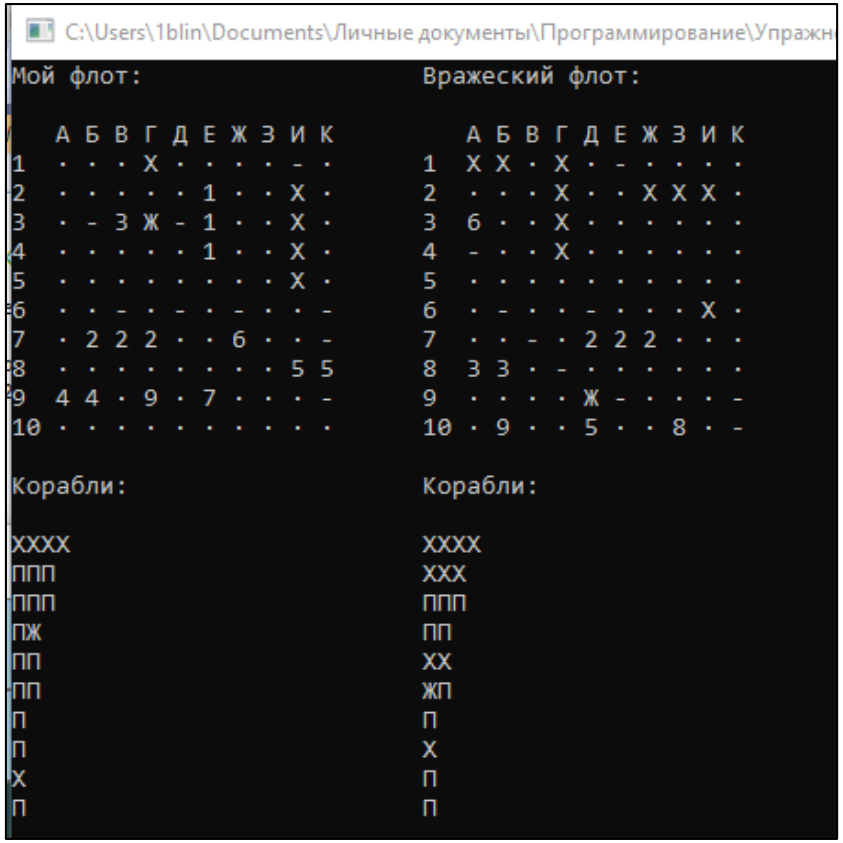


- Отношение количества побед к количеству поражений при игре класса ComputerLogicLevel_2 с ComputerLogicLevel_3 при плотности кораблей 10%

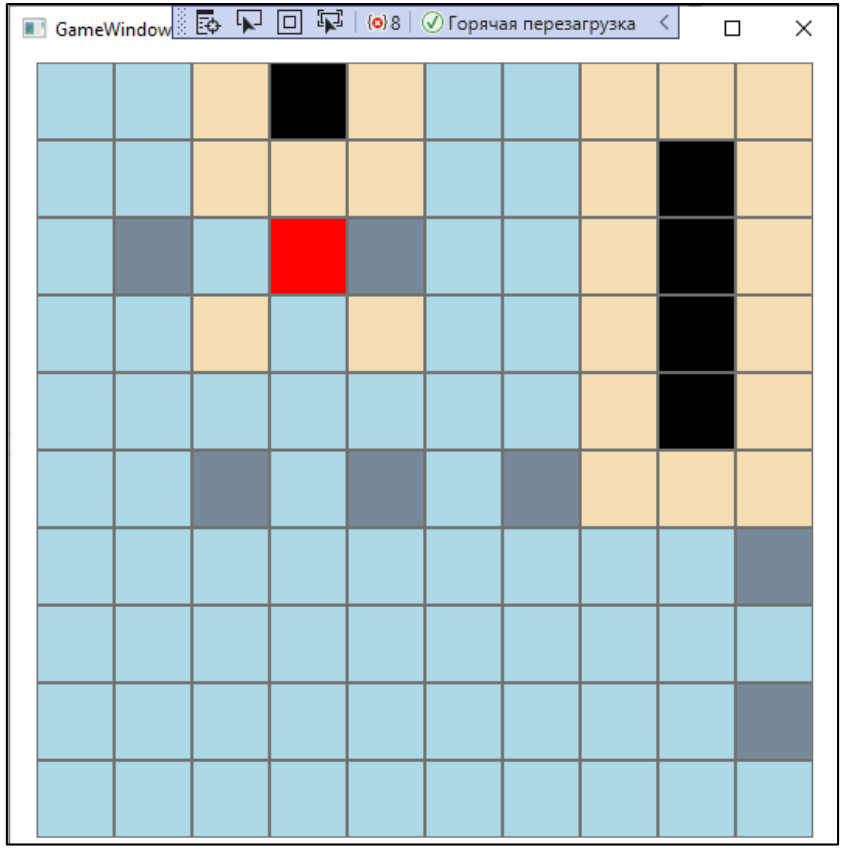
Отладка



Консольный интерфейс



Визуализация представления компьютера об игровом поле противника



Заключение

Были выполнены следующие поставленные задачи:

1. Спроектирована архитектура приложения с применением паттерна MVVM;
2. Реализовано приложение с применением технологии WPF с расширенными функциями:
   - изменение размера поля в широких диапазонах;
   - регулирование плотности расстановки кораблей;
   - выбор сложности игры при игре с компьютером;
   - выбор режима игры: любые сочетания с участием человека и компьютера;
   - ведение и сохранение статистики игры различных игроков;
3. Реализованы алгоритмы логики компьютера с разной сложностью;
4. Выполнено тестирование приложения в режиме игры компьютера с компьютером.