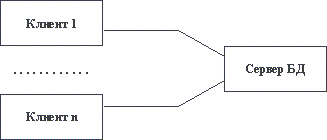
1. Проектирование системы
   1. Выбор и обоснование архитектуры «клиент-сервер»

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются сервера (процессоры, предоставляющие услуги другим процессорам) и клиенты (пользователи, запрашивающие услуги, обеспечиваемые интерфейсом сервера) [7].

Каждая реализация подобной архитектуры определяет свои правила взаимодействия между клиентом и сервером или использует уже существующие. Такие правила называют протоколом обмена или протоколом взаимодействия.

В большинстве современных сетей присутствуют элементы клиент-серверного взаимодействия на основе двух-, трех- и многозвенной архитектуры.

На рисунке 12 изображена двухзвенная архитектура, которая используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, используя при этом только собственные ресурсы, т.е. не вызывая сторонних сетевых приложений и не обращаясь к сторонним ресурсам.

  
Рисунок 12 – Двухзвенная архитектура «клиент-сервер»

Трех- и многозвенная технологии реализуются на основе модели сервера приложений, где сетевое приложение разделено на две и более частей соответственно. Каждая из таких частей может выполняться на отдельном устройстве.

Для обозначения особенностей реализации и возможностей клиента используют понятие «толщины».

«Тонким» называют клиент, который переносит все или большую часть задач по обработке информации на сервер [8]. Ресурсов «тонкого» клиента достаточно лишь для запуска необходимого приложения через WEB-интерфейс. Пользовательский интерфейс формируется средствами статического HTML, а вся прикладная логика выполняется на сервере. Для работы с «тонким» клиентом достаточно обеспечить возможность запуска WEB-браузера, в окне которого осуществляются все действия.

«Толстым», в противовес «тонкому», называют приложение, обеспечивающее расширенную функциональность независимо от центрального сервера. В этом случае сервер является лишь хранилищем данных, а вся работа по обработке и представлению этих данных переносится на машину клиента [9].

* 1. Выбор и обоснование выбора используемого протокола

В качестве протокола обмена данными в современных клиент-серверных приложениях используются протоколы TCP и UDP.

Механизм протокола TCP (Transmission Control Protocol, протокол управления передачей) предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым целостность передаваемых данных и уведомление отправителя о результатах передачи [10].

Протокол UDP (User Datagram Protocol, протокол пользовательских дейтаграмм) использует простую модель передачи, без неявных «рукопожатий» для обеспечения надежности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и дейтаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть. Чувствительные ко времени отклика приложения часто используют протокол UDP, так как для них предпочтительнее сбросить пакеты, чем ждать задержавшиеся. Кроме того, природа UDP как протокола без сохранения состояния полезна для серверов, отвечающих на небольшие запросы от огромного числа клиентов, например, для DNS-серверов [11].

Так как разрабатываемое приложение должно будет поддерживать длительное устойчивое соединение с одним клиентом, протокол TCP лучше подходит для его реализации.

* 1. Структурная схема системы

При проектировании системы используется методология структурного проектирования, в основе которой находится представление системы в виде структурной схемы. Структурная схема является совокупностью элементарных звеньев объекта и связей между ними, представленной в виде графической модели. Под элементарным звеном понимают часть объекта, которая реализует элементарную функцию [12].

Структурная схема разрабатываемой системы представлена на рисунке 13.

Приложение состоит из двух частей:

1. **Клиентская часть**, в состав которой входят:
   * *подсистема взаимодействия с сервером*, которая обеспечивает обмен данными с сервером;
   * *подсистема ведения игры*, в состав которой входят:
   * *подсистема расстановки кораблей,* которая служит для расстановки кораблей на поле боя в ручном или автоматическом режимах;
   * *подсистема совершения хода,* которая выполняет ход и оценивает его результаты;
   * *подсистема ввода/вывода,* которая обрабатывает запросы пользователя и преобразует их и полученные от сервера данные в нужный формат;
   * *подсистема визуализации,* которая отвечает за графическое отображение игрового поля.
2. **Серверная часть**, в состав которой входят:
   * *подсистема конфигурирования параметров игры,* отвечающая за конфигурирование игрового процесса;
   * *подсистема искусственного интеллекта (ИИ),* которая служит для расстановки кораблей и ведения игрового процесса при выборе режима «оффлайн»;
   * *подсистема ведения игры;*
   * *подсистема взаимодействия с клиентом,* которая преобразует информацию в необходимый формат и обменивается данными с клиентом.
   1. Спецификация системы

Спецификация – документ, полностью и в поддающейся проверке форме определяет требования или другие особенности системы, способные определить, были ли выполнены эти условия. Спецификация помогает устранить дублирования и несоответствия, позволяет точно оценить необходимые действия и ресурсы, выступает в качестве согласующего и справочного документов о внесённых изменениях, предоставляет документацию с конфигурацией, и даёт возможность взаимодействия лиц, работающих с основными функциями системного проектирования. Они дают точное представление о решении проблемы, повышая эффективность разработки системы и оценивая стоимость альтернативных путей проектирования. Они служат указанием для испытателей для верификации (качественной оценки) каждого технического требования [13].

Функциональная спецификация не определяет операции, происходящие внутри данной системы и каким образом будет реализована её функция. Вместо этого, она рассматривает взаимодействие с внешними агентами (например, персонал, использующий программное обеспечение; периферийные устройства компьютера или другие компьютеры), которые могут «следить», взаимодействуя с системой. Такое требование описывает взаимодействие внешнего агента (пользователя) и программной системы. Когда пользователь производит ввод в систему, путём нажатия кнопки ОК, программа отвечает (или должна ответить) закрытием окна, содержащего эту кнопку.

Спецификация может быть неформальной, тогда её можно воспринимать как план или руководство пользователя с точки зрения разработчика, или формальной, в таком случае она определяет математические или программные условия. На практике большинство успешных спецификаций пишутся, чтобы облегчить понимание и тонкую настройку приложений, которые уже хорошо спроектированы, но, если речь идёт о жизненно важных программных системах, в таких случаях спецификация тщательно создаётся до начала разработки приложения. Спецификации наиболее важны для внешних интерфейсов, которые должны быть стабильными [11].

* + 1. Функциональная спецификация

Функциональная спецификация в системной инженерии и разработке программного обеспечения – это документ, описывающий требуемые характеристики системы (функциональность). Документация описывает необходимые для пользователя системы входные и выходные параметры (например, программная система).

Функциональная спецификация включает в себя перечень всех функций, которые выполняет система, включая внешние и наиболее важные внутренние функции, а также перечень исключительных ситуаций с описанием их обработки.

Функциональная спецификация системы приведена в таблице 3.

* + 1. Спецификация качества

В спецификации качества определяются основные требования к следующим показателям качества системы:

1. эффективность. Под этим термином понимается отношение уровня услуг, предоставляемых системой пользователю при заданных условиях, к объему используемых ресурсов. Эффективность разрабатываемой системы должна быть достигнута за счет использования прямого подключения между игроками через протокол TCP.
2. надежность – способность программной системы выполнить возложенные на нее функции при поступлении требований на их выполнение. Надежность включает в себя:

* защищённость: система должна уметь противостоять преднамеренным или непреднамеренным деструктивным (разрушающим) действиям пользователя. Для обеспечения защищённости будет использоваться валидация входных значений на сервере приложения;
* безопасность: отказы в работе не должны приводить к аварийным ситуациям, таким как потеря данных. Для обеспечения безопасности все данные о пользователях и игре будут храниться в зашифрованном виде;
* отказоустойчивость: возможность восстановления системы и данных в случае отказа в ее работе. Для восстановления соединения с пользователем и обеспечения отказоустойчивости должны быть использованы алгоритмы обработок ошибок, сохранения всех данных в файлы и протокол TCP, который гарантирует целостность передаваемых данных.
  + 1. Перечень исключительных ситуаций

Исключительная ситуация – это ситуация, при которой система не может выполнить возложенных на нее функций или которая может привести к денормализации работы системы [9].

В таблице 4 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

Таблица 4 – Перечень исключительных ситуаций

|  |  |  |
| --- | --- | --- |
| Название  подсистемы | Название исключительной ситуации | Реакция системы |
| 1 Ведения игры | 1.1 Потеря соединения (сервер) | Бэкап сессии;  Попытка восстановления соединения в течении минуты;  Сообщение «Потеря соединения. Попытка переподключения» |
| 1.2 Потеря соединения (Клиент) | Попытка восстановления соединения в течении минуты;  Сообщение «Потеря соединения. Попытка переподключения» |
| 2 Взаимодействия с файлами | 2.1 Невозможно открыть файл сохранений | Сообщение «Файл поврежден» |
| 2.2 Невозможно открыть файл расстановок |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 3 – Перечень функций, выполняемых системой | | | | | |
| Название  подсистемы | Название функции | Информационная среда | | | |
| Входные данные | | Выходные данные | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | Тип, ограничения |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 Ввода и вывода | 1.1 Ввести ключ сервера | Набор допустимых символов | Латинские буквы, цифры и специальные символы | Ключ сервера | Строка от 8 до 20 символов |
| 1.2 Ввести название сохранения | Набор допустимых символов | Латинские буквы, цифры | Имя файла сохранения | Строка от 1 до 10 символов |
| 2 Визуализации | 2.1 Отобразить клетку поля | Координаты клетки | Массив из 2 целых чисел | Визуальное отображение клетки | ⎯ |
| Тип клетки | Перечислимый |
| 2.2 Отобразить попадание в корабль | Координаты клетки | Массив из 2 целых чисел | Визуальное отображение клетки | ⎯ |
| Анимационный файл | Файл, \*gif |
| 3 Взаимодействия с клиентом | 3.1 Создать сервер | ⎯ | ⎯ | Ключ сервера | Строка от 8 до 20 символов |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продолжение таблицы 3 | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 4 Взаимодействие с сервером | 4.1 Подключиться к серверу | Ключ сервера | Строка от 8 до 20 символов | Статус подключения | Логическое |
| 4.2 Отключиться от сервера | ⎯ | ⎯ | Статус подключения | Логическое |
| 4.3 Отправить данные | Ответ сервера | Объект json | Результат выполнения | Логическое |
| 4.4 Получить данные | ⎯ | ⎯ | Запрос на сервер | Объект json |
| 5 Конфигурация параметров игры | 5.1 Выбрать режим игры | Список режимов | * Онлайн * С компьютером | Флаг режима игры | Логическое |
| 5.2 Выбрать режим подключения (для «онлайн») | Список режимов | * Создать лобби * Подключиться к лобби | Флаг сервера | Логическое |
| 5.3 Выбрать длительность хода | Список длительностей хода | * 30 с. * 1 мин. * мин. * 5 мин. | Длительность хода | Целое |
| 5.4 Выбрать способ расстановки кораблей | Список способов расстановки | * Ручной * Случайный * По стратегии | Способ расстановки | Целое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 Конфигурация параметров игры | 5.5 Выбрать сложность ИИ | Список уровней сложности | * Лёгкий * Средний * Сложный | Сложность ИИ | Целое |
| 5.6 Включить/выключить звуковое сопровождение | Статус звукового сопровождения | Логическое | Статус звукового сопровождения | Логическое инверсное |
| 6 Взаимодействие с файловой системой | 6.1 Сохранить игру | Имя файла | Строка от 1 до 10 символов | Файл | \*arr |
| Текущее состояние игры | Объект «Игра» |
| 6.2 Загрузить игру | Имя файла | Строка от 1 до 10 символов | Текущее состояние игры | Объект «Игра» |
| 6.3 Сохранить расстановку кораблей | Имя файла | Строка от 1 до 10 символов | Файл | \*.sb |
| Расстановка кораблей | Объект «Расстановка кораблей» |
| 6.4 Загрузить расстановку кораблей | Имя файла | Строка от 1 до 10 символов | Сохраненная расстановка | Объект «Расстановка кораблей» |
| 6.3 Сохранить расстановку кораблей | Имя файла | Строка от 1 до 10 символов | Файл | \*.sb |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 Расстановки кораблей | 7.1 Очистить поле | Расстановка кораблей | Объект «Расстановка кораблей» | Расстановка кораблей | Объект «Расстановка кораблей» |
| 7.2 Поставить корабль на поле | Координаты на поле | Массив из 2 целых чисел | Id корабля | Целое |
| Расстановка кораблей | Объект «Расстановка кораблей» |
| 7.3 Повернуть корабль | Id корабля | Целое | Координаты на поле | Массив из 2 целых чисел |
| Расстановка кораблей | Объект «Расстановка кораблей» |
| 7.4 Удалить корабль | Id корабля | Целое | Расстановка кораблей | Объект «Расстановка кораблей» |
| 7.5 Расставить автоматически | Способ расстановки | Целое | Расстановка кораблей | Объект «Расстановка кораблей» |
| 7.6 Выбрать тип корабля | Список кораблей | * 4х палубный * 3х палубный * 2х палубный * 1х палубный | Тип корабля | Перечислимое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 Расстановки кораблей | 7.7 Выбрать корабль на поле | Координаты | Массив из 2 целых чисел | Id корабля | Целое |
| 8 Совершения хода | 8.1 Произвести выстрел | Координаты | Массив из 2 целых чисел | Флаг попадания | Логическое |
| 8.2 Завершить ход | ⎯ | ⎯ | Статус игрока | Целое |
| 9 Подсистема ИИ | 9.1 Выбор следующей цели | Тип стратегии | Перечислимое | Координаты | Массив из 2 целых чисел |
| 10. Справочная информация | 10.1 Выдать описание правил игры | Файл справки | Текстовый (\*.HTML) | Визуальное отображение информации |  |
| 10.2 Выдать справочную информацию о программе | Файл справки | Текстовый (\*.HTML) |
| 10.3 Выдать информацию о разработчиках | Сведения о разработчиках системы (ФИО, номер группы) | Текст (МЕМО) |
| Код ошибки | Целое |

* + 1. Разработка прототипов интерфейса пользователя

Прототип - это первоначальная версия системы, которая используются для апробирования возможностей дизайна и демонстрирования идей. Прототипы можно использовать на различных фазах разработки. Например, на этапе анализа требований при их нахождении и проверке; на этапе дизайна при исследовании выбора возможностей и планировании пользовательского интерфейса [].

Прототип позволяет также получить обратную связь от будущих пользователей, причем, именно тогда, когда это наиболее необходимо: в начале проекта еще есть возможность исправить ошибки проектирования практически без потерь.

Требования к прототипу изменяются со временем. Сначала наиболее актуальными его свойствами являются скорость создания и простота модификации. Эти свойства позволяют быстро разработать и проверить несколько версий интерфейса, при этом ещё и исправить значительную часть ошибок. Только затем на первый план выходят функциональность и эстетичность.

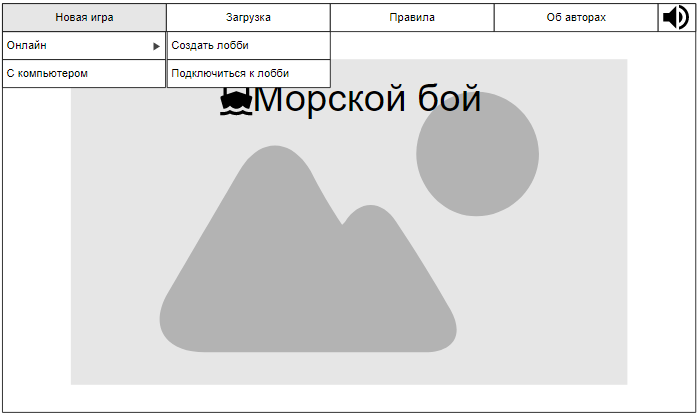
Пользовательский интерфейс должен быть интуитивно понятным, дружественным и удобным. Программист должен поставить себя на место пользователя, который впервые видит программу, и создать такой интерфейс, который не потребует много времени на обучение, понимание и восприятие программы, а также на работу с окнами.

На рисунке 14 приведён прототип начальной формы игры. В меню находятся следующие разделы:

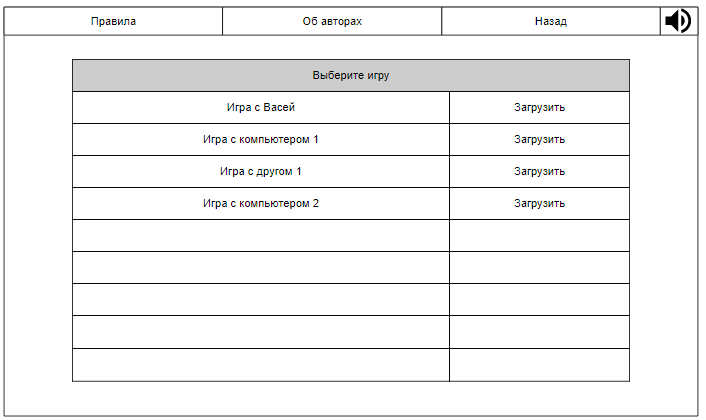
* «Новая игра»;
* «Загрузка»;
* «Правила»;
* «Об авторах».

При нажатии на меню «Новая игра» должно появляться выпадающее меню, позволяющее выбрать режим игры по сети («Онлайн») или с компьютером («С компьютером»). При выборе игры по сети должно появляться выпадающее меню, позволяющее выбрать, создавать ли новую игровую комнату (кнопка «Создать лобби») или подключиться к существующей (кнопка «Подключиться к лобби»).

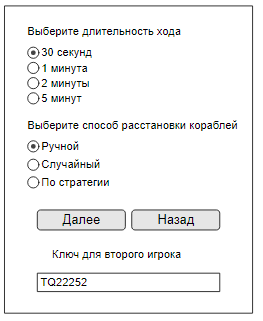
В правом углу меню находится графическая кнопка управления звуком, позволяющая включать и отключать звуковое сопровождение.

  
Рисунок 14 – Прототип начальной формы системы

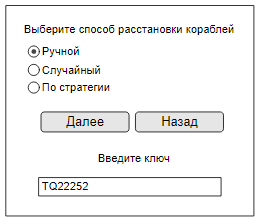
На рисунке 15 приведён прототип формы загрузки. По центру формы должна отображаться таблица, где будет приведён список всех ранее сохранённых незаконченных игр. После нажатия кнопки «Загрузка» откроется форма с игрой, начиная с того момента, на котором она была сохранена. При нажатии кнопки «Назад» должен происходить возврат к начальной форме.

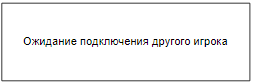
  
Рисунок 15 – Прототип формы загрузки игры

После выбора новой игры в режиме «Онлайн» с созданием новой игровой комнаты должно появляться окно настроек новой игры создателя комнаты (рисунок 16), где пользователь должен выбрать длительность хода для обоих игроков и способ расстановки своих кораблей. Внизу окна должен отображаться ключ, по которому будет подключаться второй игрок. Ключ не должен быть доступен для редактирования. При нажатии кнопки «Далее» и выборе ручной расстановки кораблей должна открываться форма расстановки кораблей (см. рисунок 20), при выборе другого режима должно появляться окно ожидания подключения другого игрока (см. рисунок 18), при нажатии кнопки «Назад» должен происходить возврат в начальную форму.

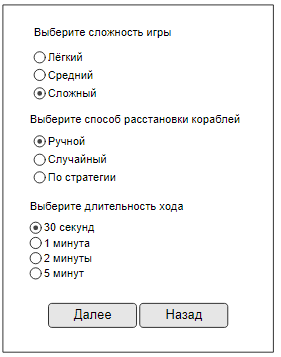
  
Рисунок 16 –Прототип окна настроек новой игры создателя игровой комнаты

После выбора новой игры в режиме «Онлайн» с подключением к существующей комнате, должно появляться окно настроек новой игры   
(рисунок 17), где пользователь должен выбрать способ расстановки своих кораблей. Внизу окна находится поле, в которое пользователь должен ввести ключ существующей игровой комнаты до начала боя. При нажатии кнопки «Далее» и выборе ручной расстановки кораблей должна открываться форма расстановки кораблей (см. рисунок 20), при выборе другого режима должно появляться окно ожидания подключения другого игрока (см. рисунок 18), при нажатии кнопки «Назад» должен происходить возврат в начальную форму.

  
Рисунок 17 – Прототип окна настроек новой игры подключающегося игрока

  
Рисунок 18 – Прототип окна ожидания подключения

После выбора новой игры в режиме «С компьютером» должно появляться окно настроек новой игры (рисунок 19), где пользователь должен выбрать длительность хода для обоих игроков, сложность искусственного интеллекта и способ расстановки своих кораблей. При нажатии кнопки «Далее» и выборе ручной расстановки кораблей открывается форма расстановки кораблей   
(см. рисунок 20), при выборе другого режима появляется форма игры   
(см. рисунок 23), при нажатии кнопки «Назад» происходит возврат в начальную форму.

  
Рисунок 19 – Прототип окна настроек новой игры против компьютера

На рисунке 20 показан прототип формы ручной расстановки кораблей. В меню должны находиться следующие кнопки:

* «Готов»;
* «Сохранить расстановку»;
* «Загрузить расстановку»;
* «Правила»;
* «Об авторах»;
* «Выход».

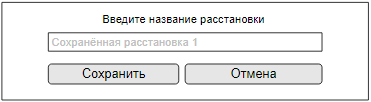
  
Рисунок 20 – Прототип формы расстановки кораблей

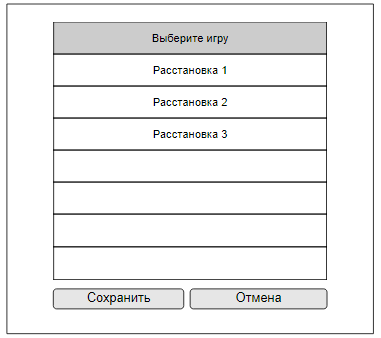
При нажатии кнопки «Сохранить расстановку» должно открываться окно сохранения расстановки (см. рисунок 21). При нажатии кнопки «Загрузить расстановку» должно открываться окно выбора расстановки (см. рисунок 22). При нажатии кнопки «Выход» должно появляться окно подтверждения выхода (см. рисунок 24).

Слева формы будет располагается поле игрока, где должно отображаться расположение кораблей. Справа должны отображаться корабли и их оставшееся количество. Пользователь должен разместить корабли на поле с помощью перетаскивания мышкой изображений кораблей из оставшихся на поле. Должна быть предоставлена возможность перевернуть выбранный корабль на 90 градусов нажатием правой кнопки мыши.

После завершения расстановки, пользователь должен нажать кнопку «Готов» в меню, после чего, если выбрана игра «Онлайн», должно появиться окно ожидания другого пользователя (см. рисунок 18) или, если выбран режим «С компьютером», начнётся бой.

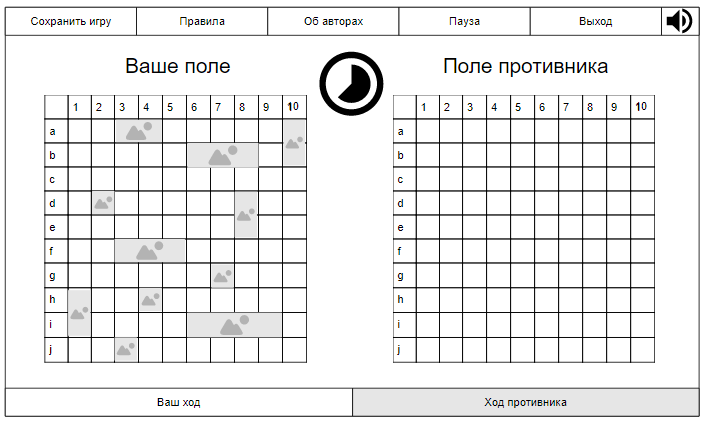
  
Рисунок 20 – Прототип формы расстановки кораблей

  
Рисунок 21 – Прототип формы сохранения расстановки кораблей

  
Рисунок 22 – Прототип формы загрузки расстановки кораблей

На рисунке 23 показан прототип формы игры. В меню должны находиться следующие кнопки:

* «Сохранить игру»;
* «Правила»;
* «Об авторах»;
* «Пауза»;
* «Выход».

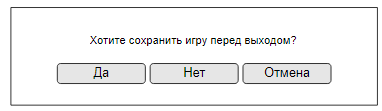
  
Рисунок 23 – Прототип формы игры

При нажатии кнопки «Сохранить игру» должно появляться окно сохранения игры (рисунок 25). При нажатии на кнопку «Пауза» игра будет приостанавливается, а надпись на кнопке изменится «Продолжить игру». При нажатии на кнопки «Выход» должно появиться всплывающее окно подтверждения (рисунок 24).

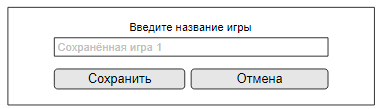
В середине формы должны располагаться два поля (пользователя, где будет отображаться расположение пользовательских кораблей и их текущее состояние, и противника, где расположение кораблей будет скрыто, и будет отображаться только изменение состояния отдельных клеток). Между ними должен располагаться таймер, который будет отображать оставшееся время до окончания хода.

В нижней части формы должна находиться строка, на которой будет отображаться подсвечиванием, кому принадлежит текущий ход. При выборе игроком, которому принадлежит ход, клетки на поле противника, состояние которой ранее не было изменено, её состояние должно измениться, ход должен перейти к другому игроку, должна измениться подсветка нижней строки и сброситься состояние таймера.

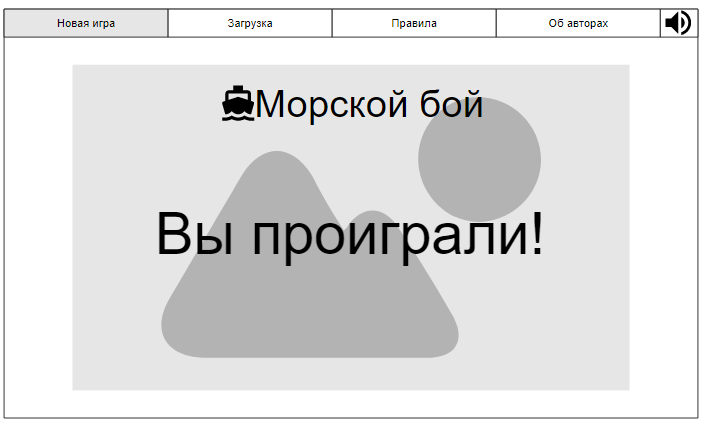
На рисунке 24 показан прототип окна выхода из игры, где пользователю будет предложена возможность сохранить текущую игру. При нажатии кнопки «Да», должно открыться окно сохранения игры (см. рисунок 25), при нажатии кнопки «Нет» текущая игра должна закончиться без подведения итогов, пользователя должно возвратить в начальную форму, при нажатии кнопки «Отмена» должен происходить возврат к форме игры.

  
Рисунок 24 – Прототип окна выхода из игры

На рисунке 25 показан прототип окна сохранения игры, где пользователю будет предложено сохранить текущую игру. Пользователь должен ввести в поле желаемое название игры. После ввода названия и нажатия кнопки «Сохранить», игра будет сохранена и произойдёт возврат пользователя в предыдущее окно, после нажатия кнопки «Отмена» произойдёт возврат к предыдущему окну без сохранения.

  
Рисунок 25 – Прототип диалогового окна «Сохранение игры»

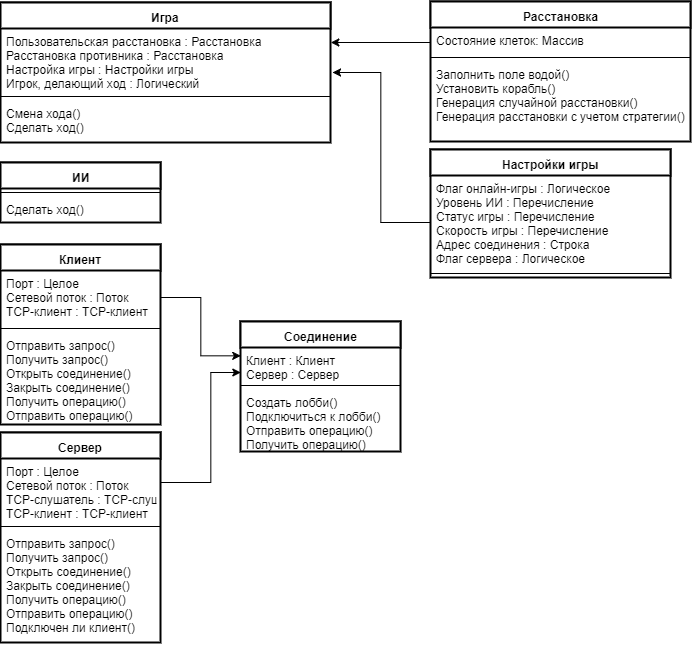
На рисунке 26 показан прототип начальной формы с подведением итогов предыдущей игры. Пользователю должны быть доступны те же действия, что и на начальной форме на рисунке 14.

  
Рисунок 26 – Прототип начальной формы с отображением итога игры

* 1. Разработка структур данных и классов

Диаграмма классов служит для представления статической структурной модели системы в терминологии классов объектно-ориентированного проектирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений. При этом на данной диаграмме не указывается информация о временных аспектах системах.

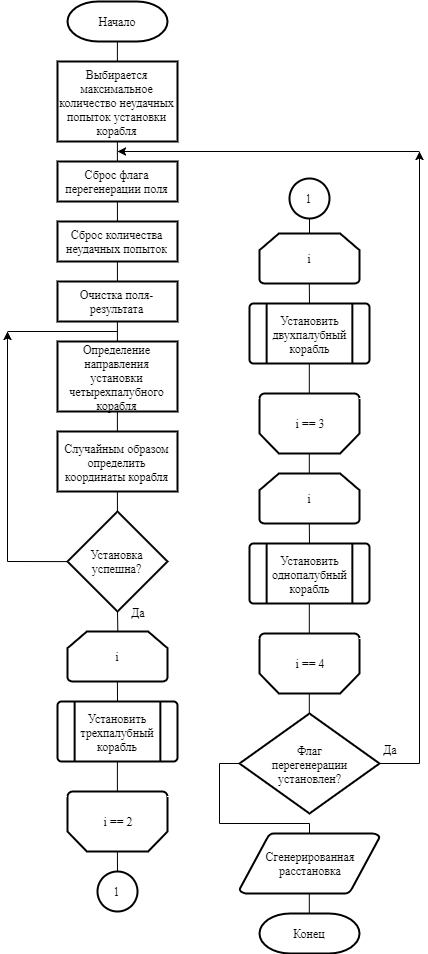
Класс в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Описание класса состоит в определении атрибутов и методов [14].

  
Рисунок 27 – Диаграмма классов системы

* 1. Выбор и обоснование алгоритмов обработки данных

Алгоритм – это конечный набор предписаний для получения решения задачи посредством конечного количества операций [16]. Многие задачи могут быть решены с использованием различных путей; в таком случае наиболее подходящее решение выбирается исходя из различных критериев.

На рисунке 28 представлен алгоритм расстановки кораблей на поле случайным образом.

  
Рисунок 28 – Схема алгоритма расстановки кораблей случайным образом

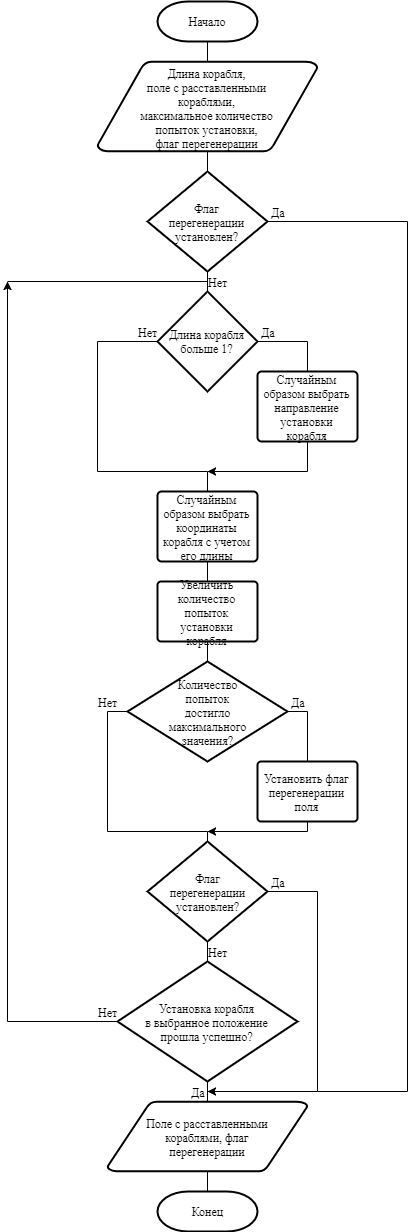
Для упрощения алгоритма в отдельную подфункцию выделена установка отдельного корабля. Алгоритм установки корабля по случайным координатам представлен на рисунке 29.

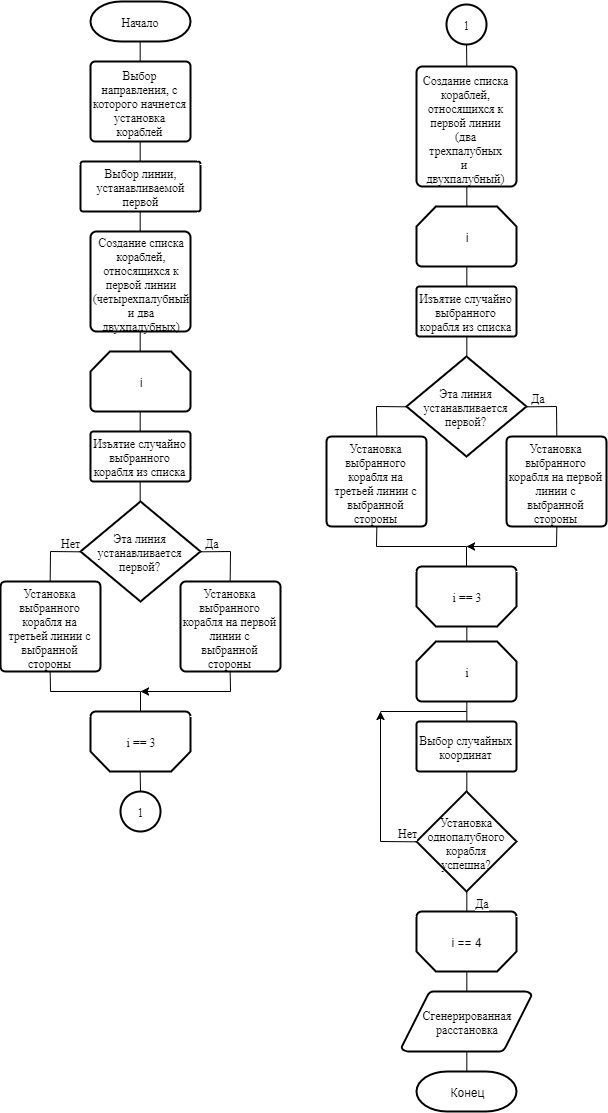
Помимо расстановки кораблей на поле случайным образом будет существовать возможность расстановки коралей в соответствии со стратегией, разработанной Я.И. Перельманом. Алгоритм генерации такой расстановки приведен на рисунке 30.

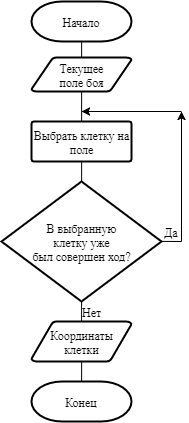
Подсистема ИИ должна содержать несколько реалиаций, соответствующих трем различным уровням сложности. Алгоритмы совершения хода ИИ легкой, средней и высокой сложности представлены соответственно на рисунках 31, 32, 33. Алгоритмы ИИ средней и высокой сложности используют подпрограмму добивания кораблей, алгоритм которой приведен на рисунках 34, 35.

* 1. Выбор и обоснование комплекса программных средств
     1. Выбор языка программирования и среды разработки

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов — это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов [18].

  
Рисунок 29 – Схема алгоритма установки корабля по случайным координатам

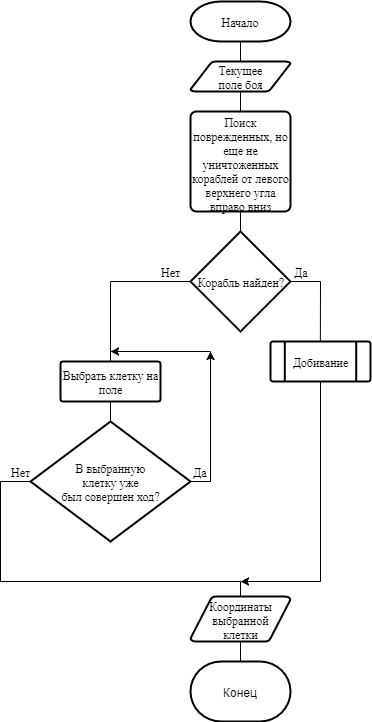
  
Рисунок 30 – Схема алгоритма расстановки кораблей в соответствии со стратегией.

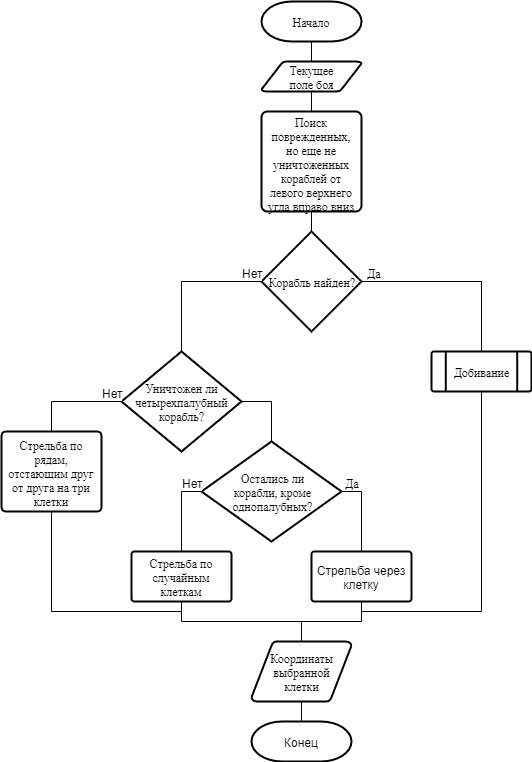
  
Рисунок 31 – Схема алгоритма совершения хода ИИ легкой сложности

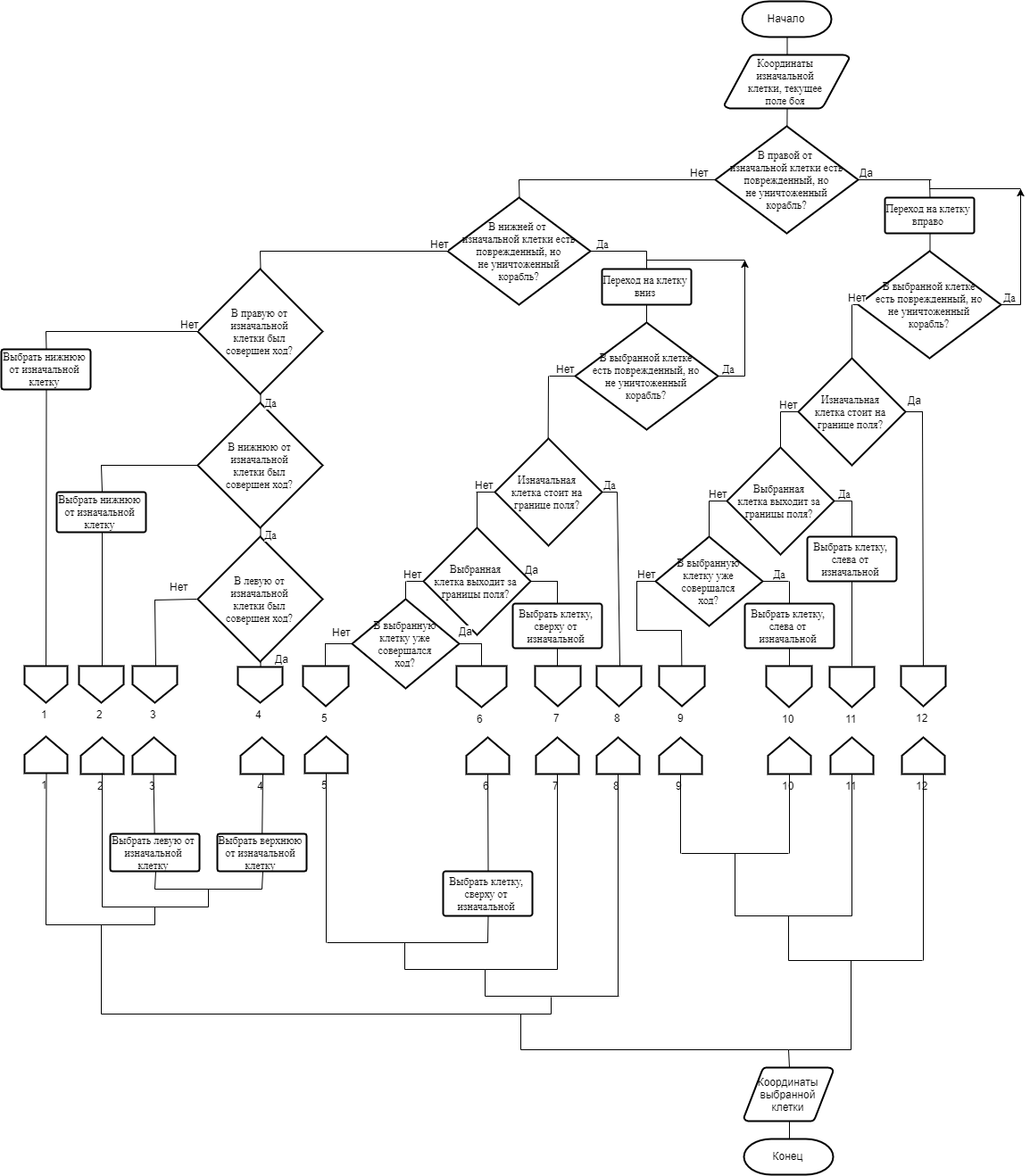
* + 1. Выбор операционной системы

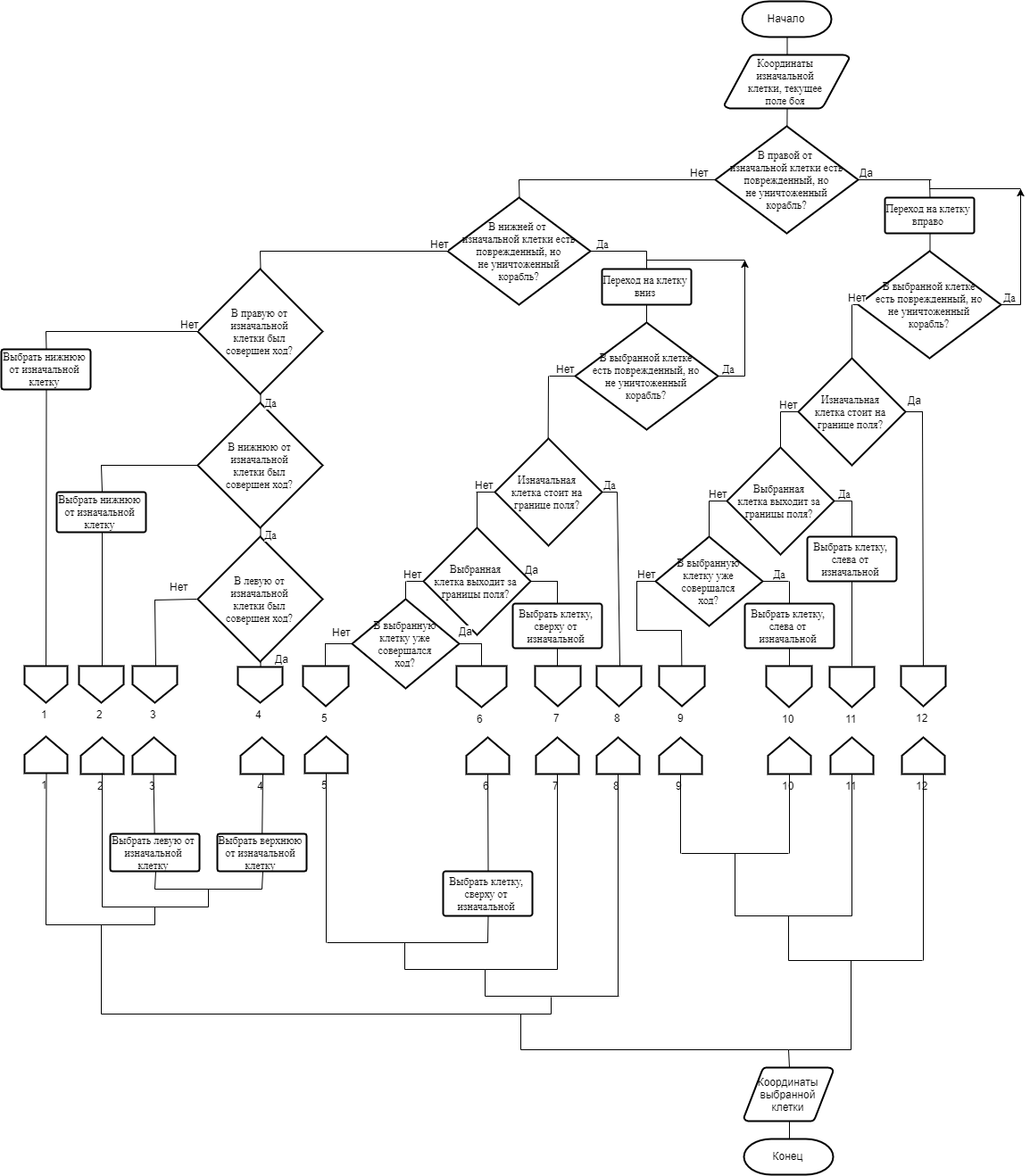
В качестве операционной системы была выбрана система семейства MS Windows – Windows 10. В ней вы найдете несколько приятных новшеств, которые были очень грамотно перенесены из предыдущих версий. Она предоставляет программистам все необходимые средства для создания пользовательского интерфейса.

Линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WPF. Visual Studio 2017 содержит множество улучшений, которые повышают производительность основных используемых функций. Заметно уменьшится время загрузки и объем используемой памяти, а также повысится скорость отклика во всем цикле разработки [19].

  
Рисунок 32 – Схема алгоритма совершения хода ИИ средней сложности

  
Рисунок 33 – Схема алгоритма совершения хода ИИ высокой сложности

  
Рисунок 34 – Схема алгоритма добивания корабля

  
Рисунок 35 – Продолжение схемы алгоритма добивания корабля

1. Реализация системы
   1. Разработка и описание интерфейса пользователя

При разработке интерфейса необходимо следовать следующим принципам: