Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Объектно-ориентированное программирование

К защите допустить:
И.О. Заведующего кафедрой информатики
_____С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ТЕСТИРОВАНИЯ

БГУИР КП 1-40 04 01

Студент Б.Я. Дмитрук

Руководитель Е. В. Тушинская

Нормоконтролер Е. В. Тушинская

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Уже в течение многих тысячелетий, а точнее - более пяти тысяч лет нарды являются наиболее популярной настольной игрой. Кем именно она впервые была придумана никому до сих пор не известно. Это могло произойти где угодно, в Египте, Персии, Греции или в Индии. Корни этой игры уходят в глубину веков, точно известно только то, что зародилась она на Востоке. Историческим доказательством этому служит доска, найденная в Малой Азии, датируемая годом около 5000 лет до нашей эры. Находки, подобные этой, были обнаружены в гробнице Тутанхамона (XV век до нашей эры).

Существует легенда, связанная с игрой в нарды. Согласно ей, однажды индусами персам были посланы шахматы в расчете на то, что персы не смогут понять принцип игры. Персы не только разобрались в шахматах, благодаря их мудрецу Бюзюркмерху, но и сделали ответное послание тоже в виде загадочной игры. Ее принцип индусы не могли разгадать в течение 12 лет.

На этом персы не остановились, все тот же мудрец Бюзюркмерх создал нарды, правда, тогда они назывались нард-тахте, и снова послали игру персам. Персы были народом верующим, и в III веке до нашей эры нарды приобрели для них символическое и мистическое значение, а астрологи научились с их помощью предсказывать судьбу. Происходило это следующим образом: доску, предназначенную для игры, они использовали как небо, а фишки были звездами, отображающими движение настоящих звезд.

Нарды полны символики. Можно предположить, что они произошли от одной из известных тогда игр, так как все имели доски, похожие на доску для нард. Все, что на доске, имеет непосредственную связь со временем. Например, 12 ячеек на каждой стороне символизировали год, состоящий из 12 месяцев; доска разделена на 4 части, что означает 4 времени года; имеющиеся 24 пункта - это 24 часа, сутки; 30 фишек воплощали в себе число лунных и безлунных ночей месяца, а движение фишек по кругу - движение звезд на небе. В процессе игры в нарды бросают кости, сумма очков на них равно семи, а в то время было известно именно семь планет, с которыми связывали все происходящие события.

Для изготовления нард применялся различный материал: доски делали из камня или дерева, а кости - из собственно костей, камня или лепились из глины. Естественно, что нарды со временем совершенствовались, и древние нарды отличались от нынешних. Например, доска, которая была обнаружена в городе Ур, имела всего 12 клеток, размещавшиеся в ящиках по 6 в каждом, а ящики соединены по два.

В каждой стране нарды имели свое название. Испанцы называли их *tablero*, немцы *bretspiel*, греки - *diagramismos*, итальянцы - *tavola reale*, французы - *trick-track*, турки - *tavla*, а англичане - *backgammon*.

Название, данное этой игре персами, переводится как «битва на деревянной доске». Однако, самое древнее имя, которое имели нарды, звучит как «Таблица» или еще - «Королевская таблица». Римское название - «Табула», а греческое - «Таблас», испанское - «Реалес», английское - «*Tables*».

В Западную Европу нарды попали вместе с крестоносцами, возвращавшимися из походов в 12 веке. В Европе игра стала популярной под названием Трик-трак, происхождение которого связано со звуком костей при ударе о доску. В Средние века в Европе нарды являлись игрой королевских особ, которые имели исключительную привилегию играть именно в нарды. Постепенно популярность игры достигла Среднего Востока и Средиземноморья, а в наше время нарды распространены во всем мире.

Правила игры, несмотря на восточное происхождение, были разработаны англичанином Эдмондом Хойлом в 1743 году. Сегодня в нарды играют даже в специализированных любительских клубах, где проходят турниры на международном уровне. Даже на английском канале *Sky* был показан первый фильм, посвященный нардам и чемпионату мира по этой игре, в котором подробно обо всем рассказывалось. Назывался фильм «Нарды. Крупные ставки».

Самый эффективный способ заполнить досуг, а также весело и с пользой для воспитания воли, тренировки концентрации внимания и развития стойкости характера провести время - настольные игры. В них можно играть всей семьей и это является отличным развлечением и прекрасным отдыхом от будничных забот.

Кроме историко-культурного аспекта, ключевыми факторами при выборе темы курсовой работы стали возможность изучить новые инструменты и средства программирования из разных областей, а также необходимость творческого подхода при создании графического интерфейса программного продукта.

Кроме того, нарды по сегодняшний день остаются актуальной игрой как в аналоговом, так и в цифровом формате, имеющие немалое количество десктопных, мобильных и браузерных реализаций. Исходя из этого подобная видеоигра вполне могла бы стать полноценным коммерческим проектом.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Правила игры

В сравнении с другими распространенными настольными настольными играми, такими как шашки или шахматы, нарды, за неимением долгое время единых стандартных правил игры, успели обзавестись массой различных вариаций. Две основные из них – длинные нарды и короткие нарды.

Короткие нарды – игра для двух игроков, на доске, состоящей из двадцати четырех узких треугольников, называемых пунктами. Треугольники чередуются по цвету и объединены в четыре группы по шесть треугольников в каждой. Эти группы называются - дом, двор, дом противника, двор противника. Дом и двор разделены между собой планкой, которая выступает над игровым полем и называется бар (рисунок 1.1).

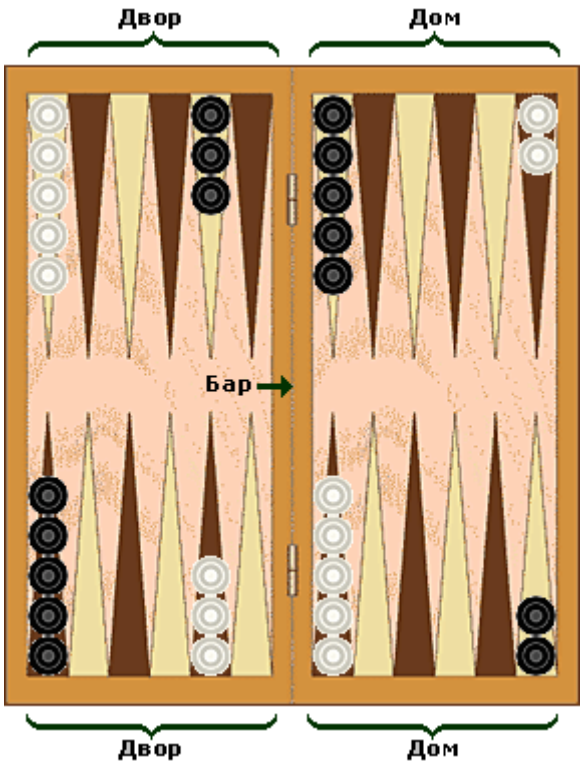

Рисунок 1.1 – Доска с шашками в начальной позиции. Возможна расстановка, зеркально симметричная к той, что приведена на рисунке. Дом в ней располагается слева, двор –справа.

Пункты нумеруются для каждого игрока отдельно, начиная с дома данного игрока. Самый дальний пункт является 24-м пунктом, он также

является первым пунктом для оппонента. У каждого игрока имеется 15 шашек. Начальная расстановка шашек такова: у каждого из игроков по две шашки в двадцать четвертом пункте, пять в тринадцатом, три в восьмом и пять в шестом.

Цель игры - перевести все свои шашки в свой дом и затем снять их с доски. Первый игрок, который снял все свои шашки, выигрывает партию.

Движение шашек

Игроки поочередно бросают по две кости и выполняют ходы.

Число на каждой кости показывают, на сколько пунктов, или шагов, игрок должен передвинуть свои шашки. Шашки всегда движутся только в одном направлении – от пунктов с большими номерами к пунктам с меньшими (рисунок 1.2).

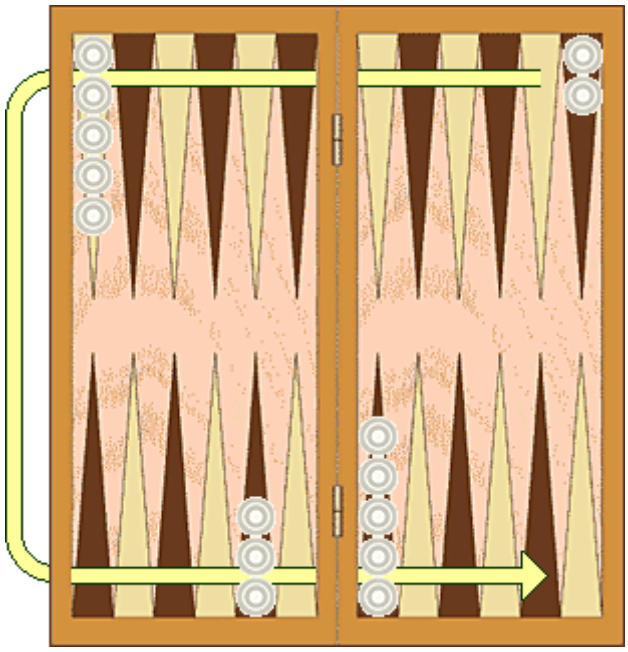

Рисунок 1.2 – Направление движения белых шашек. Черные шашки движутся в противоположном направлении.

При этом применяются следующие правила:

- шашка может двигаться только на открытый пункт, то есть на такой, который не занят двумя или более шашками противоположного цвета.

- числа на обеих костях составляют отдельные ходы.

К примеру, если у игрока выпало 5 и 3, то:

- он может пойти одной шашкой на три шага, а другой - на пять,

- либо он может пойти одной шашкой сразу на восемь (пять плюс три) шагов, но последнее лишь в том случае, если промежуточный пункт (на расстоянии три или пять шагов от начального пункта) также открыт (рисунок 1.3).

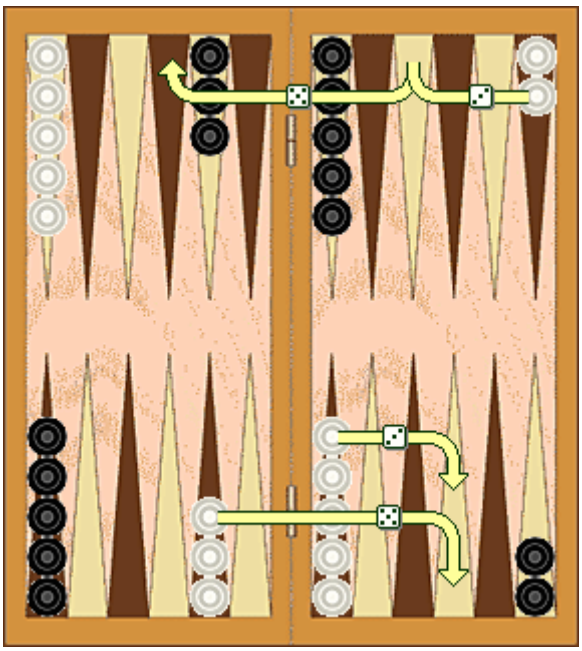

Рисунок 1.3 – Способы, которыми может сыграть белый игрок

Игрок, у которого выпал дубль, играет каждое из чисел на каждой из костей дважды. Например, если выпало 6-6, то игрок должен сделать четыре хода по шесть очков, и он может передвинуть шашки в любой комбинации, как сочтет нужным.

Игрок должен использовать оба числа, которые ему выпали, если они допускаются правилами (либо все четыре числа, если у него выпал дубль). Когда можно сыграть только одно число, игрок обязан сыграть это число.

Если каждое из чисел по отдельности можно сыграть (но не оба вместе), игрок должен играть большее число.

Если игрок не может сделать хода, то он пропускает ход. В случае, если выпал дубль, если игрок не может использовать все четыре числа, он должен сыграть столько ходов, сколько возможно.

Когда игрок привел все свои пятнадцать шашек в свой дом, он может начать выбрасывать их с доски. Игрок выбрасывает шашку следующим образом: бросается пара костей, и шашки, которые стоят на пунктах, соответствующих выпавшим значениям, снимаются с доски. Например, если выпало 6 очков, можно снять шашку с шестого пункта (рисунок 1.3).

Если на пункте, соответствующем выпавшей кости, нет ни одной шашки, игроку разрешается переместить шашку с пунктов, больших, чем выпавшее число. Если игрок может сделать какие-либо ходы, он не обязан выбрасывать шашку с доски.

Длинные нарды – также игра на двоих игроков. Количество шашек такое же, как и в коротких: по 15 на каждого игрока (рисунок 1.4).

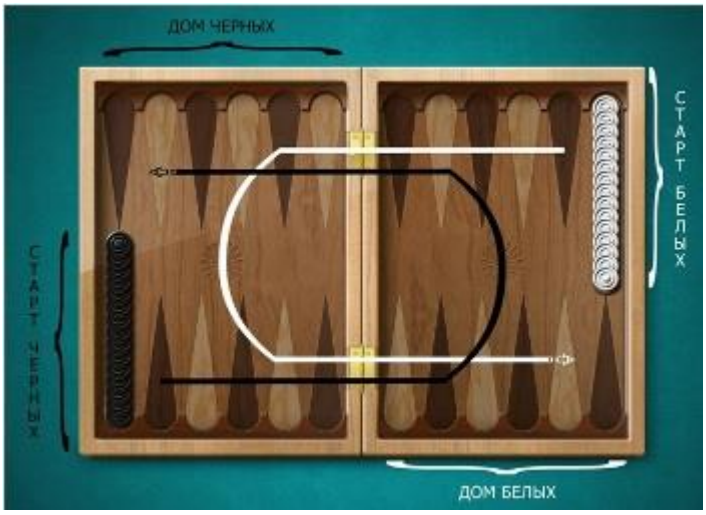

Рисунок 1.4 – Начальное расположение шашек на доске и направление хода для обоих игроков

Место начального расположения шашек, каждого из игроков называется голова, а ход из начального положения называется "из головы" или "взять с головы". За один ход с головы можно брать только одну шашку.

Игрок кидает одновременно два кубика. Сделав бросок, игрок должен передвинуть любую шашку на число клеток, равное выпавшему числу одного из кубиков, а затем одну любую шашку - на число клеток, равное выпавшему числу другого кубика. Т.е. если на кубиках выпало, например, шесть-пять, игрок должен одну шашку передвинуть на шесть клеток, а затем любую (можно ту же, можно другую) на пять клеток. При этом с головы всегда можно брать только одну шашку.

Передвинуть две шашки на число клеток, показанное одним кубиком нельзя. Т.е. если на кубиках выпало - шесть-пять, игрок не может пойти одной шашкой, например, на три и другой на три клетки, чтобы вместе получилось шесть, а затем сходить "пятёрку".

Если выпал дупль, т.е. одинаковые очки на двух кубиках, например, пять-пять, игрок делает четыре хода (на соответствующее кубиков число клеток).

Нельзя поставить свою шашку не клетку, занятую шашкой противника. Если шашка попадает на занятую клетку, то про неё говорят что она "не идёт". Если шашки противника занимают шесть клеток перед какой-нибудь шашкой, то такая шашка оказывается запертой.

Нельзя запереть все пятнадцать шашек противника. Т.е., выстроить заграждение из шести шашек подряд можно только в том случае, если хотя бы одна шашка противника находится впереди этого заграждения.

Если игрок не может сделать ни одного хода на то количество очков, которое выпало на каждом кубике, т.е. если шашки не идут, то очки пропадают, а шашки не двигаются.

Если игрок может сделать ход на то количество очков, которое выпало на одном из кубиков, и не может сделать хода на то количество очков, которое выпало на втором кубике, он делает только тот ход, который возможен, а остальные очки пропадают.

Если у игрока есть возможность сделать полный ход, он обязан его сделать даже в ущерб своим интересам. Если выпал такой камень, который позволяет игроку сделать только один ход, причем любой из двух то игрок должен выбрать больше. Меньшие очки пропадают. Смысл игры заключается в том, чтобы, пройдя всеми шашками полный круг, прийти ими в дом и выбросить все шашки раньше, чем это сделает противник.

Домом для каждого игрока является последняя четверть игрового поля, начиная с клетки, отстоящей от головы на 18 клеток. Выбрасывать шашки - это значит делать ими такие ходы, чтобы шашки оказывались за пределами доски. Игрок может начать выбрасывать шашки только тогда, когда все его шашки пришли в дом (рисунок 1.5).

Рисунок 1.5 – Процесс «выбрасывания» шашек белым игроком

Ничьей не существует. Если игрок, начинавший первым, выбросил все свои шашки, а второй игрок может сделать тоже следующим броском, второй считается проигравшим, так как следующего броска не будет: партия заканчивается, как только один из игроков выбросил все свои шашки.

1.2 Постановка задачи

Мною для реализации были выбраны именно длинные нарды в силу того, что сам я именно в этот подвид играю чаще всего, а также по той причине, что в интернет-магазинах приложений по запросу «нарды» результатами поиска чаще всего являются либо приложения, реализующие различные виды игры, либо только «длинную» их вариацию.

Для организации разработки продукта было составлено следующее техническое задание:

Разработать программное обеспечение, предоставляющее функционал для игры в длинные нарды на базе операционной системы Windows. Целевой платформой программного продукта была назначена Windows. Основной язык, используемый в разработке – *C#*. Основной фреймворк, используемый в разработке – *.NET Framework*. Пользовательский интерфейс должен быть реализован с использованием графических компонентов на базе фрейморка *Windows Presentation Foundation*. Программный продукт должен представлять собой клиент пользовательского интерфейса, реализующего запрашиваемый функционал используя реализованные независимо от него библиотеки сервисов и сущностей.

Основная функция, реализуемая разрабатываемым программным обеспечением – интерактивная игровая доска, позволяющая проводить игру между двумя игроками на одном стационарном персональном компьютере в соответствии с вышеизложенными правилами игры в длинные нарды.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Разработка функциональности программного средства

Для разработки программного продукта был выбран язык программирования как *C#*, так как совместно с платформой .NET он предоставляет широкий набор инструментов и библиотек для создания сложных приложений, включая игры. В качестве фреймворка для реализации пользовательского интерфейса был выбран фреймворк *Windows Presentation Foundation*, так как он предоставляет широкий выбор элементов управления, стилей, анимаций и возможность создания красивого и интуитивно понятного пользовательского опыта. В качестве интегрированной среды разработки была выбрана *Microsoft Visual Studio 2022*, поскольку данная платформа имеет высокую степень интеграции с перечисленными выше инструментами разработки в силу принадлежности всех указанных продуктов *Microsoft*. Следствием этого фактора является мощность и удобство, предоставляемые данной платформой, а также большое количество качественной документации. Кроме того, для разработки визуальных компонент пользовательского интерфейса был использован *Adobe Photoshop 2024*, так как данный инструмент я нахожу наиболее подходящим для создания двумерных графических изображений.

2.2 Архитектура программного средства

Для реализации программного средства была выбрана слоистая архитектура. Она представляет из себя три уровня: уровень представления, уровень бизнес-логики и уровень данных.

Самый верхний уровень – уровень представления, он же уровень пользовательского интерфейса. В данном слое реализуются необходимые инструменты для отображения и получения контента от пользователя посредством взаимодействия с графическими элементами приложения. Основной задачей данного слоя является предварительная обработка данных, введенных пользователем, передача их на уровень бизнес-логики приложения, последующее получение данных от уровня бизнес-логики и изменение графических элементов в соответствии с полученными данными. Непосредственно в решении разрабатываемого программного продукта данный класс представлен проектом *UserInterface*.

Уровень приложения, или уровень бизнес-логики. Данный слой реализовывает непосредственно сам процесс игры. Он хранит в себе объекты моделей из уровня данных и, посредством вызова реализованных в нем

методов, изменяет хранимые в объектах моделей данные для последующей их передачи на уровень представления. Данный слой не знает о том, откуда пришли данные и куда пойдут после обработки. На этом уровне обеспечивается лишь общая логика работы приложения и преобразование данных в нем. Непосредственно в решении разрабатываемого программного продукта данный класс представлен проектом *Logic*.

Центральный уровень – уровень данных. Данный слой реализует набор моделей – абстракций, представляющих собой структуры данных, реализованные в соответствии с моделируемыми объектами и требованиями к функционалу приложения. Непосредственно в решении разрабатываемого программного продукта данный класс представлен проектом *Entities*.

В соответствии с заявленной выше архитектурой разработана *UML* диаграмма классов, демонстрирующая взаимодействие между сущностями и между слоями проекта (рисунок 2.2.1).

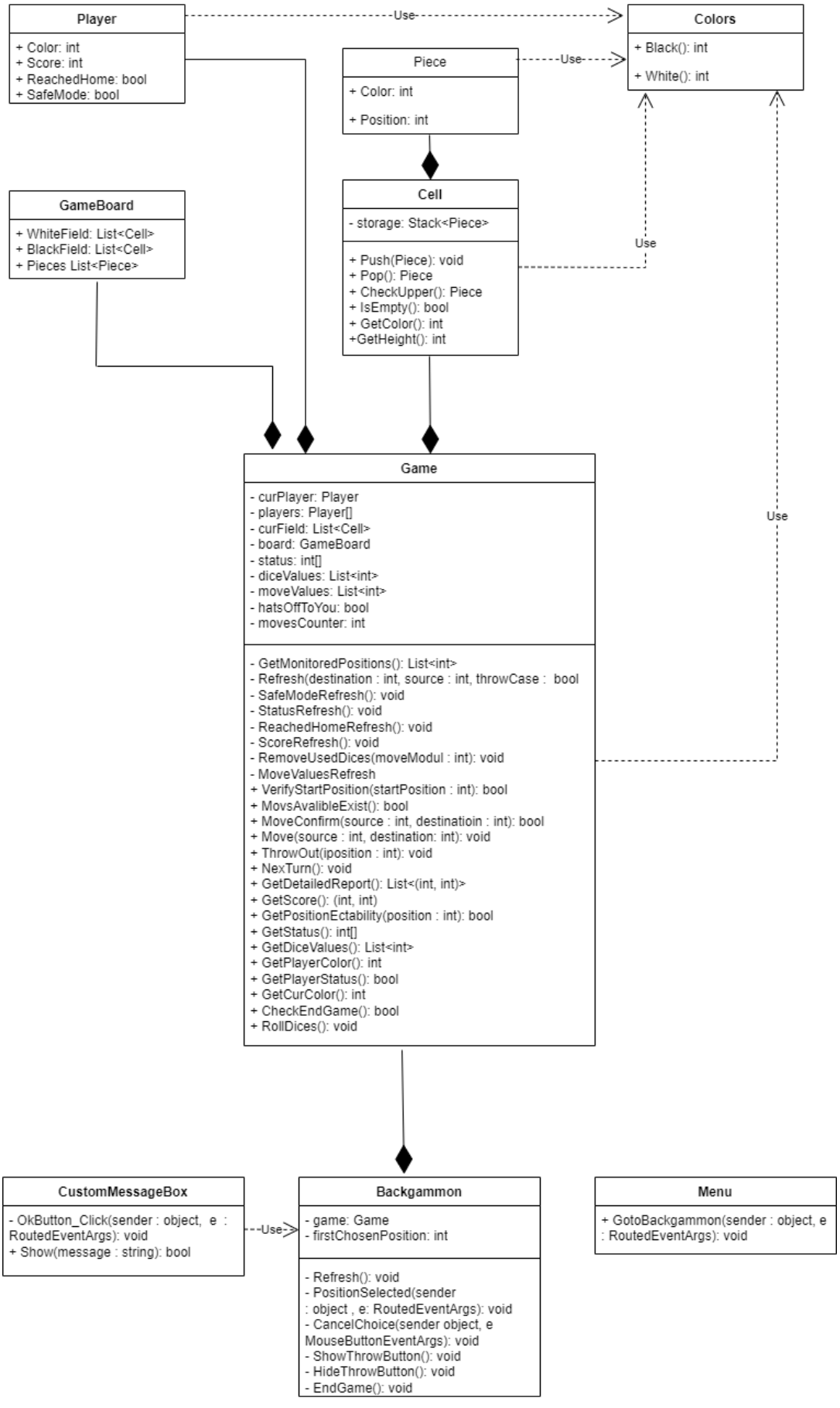

Рисунок 2.2.1 – Диаграмма классов проекта

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка уровня представления

Данный уровень реализован тремя классами с соответствующими каждому из них файлами *.xaml*, позволяющими отображать графические пользовательские элементы.

Backgammon – класс представления, отвечающий за саму страницу игры. Как можно заметить из диаграммы 2.2.1, не хранит никаких данных кроме индекса первой выбранной в текущем ходу позиции и объекта игры, реализующего логику игрового процесса, так же в своей работе использует объект класса *CustomMessageBox*. Реализует следующие функционал:

1. Обновление графических элементов
2. Обработка на нажатие игровой ячейки
3. Отмена выбора игровой ячейки
4. Изменение видимости кнопки сброса шашки
5. Обработка окончания игровой сессии

Menu – класс, реализующий одну функцию – навигацию со стартовой на другие страницы приложения. Так как реализован только один вид игры, то и навигироваться этот класс позволяет только на страницу, содержащую игровое поле.

CustomMessageBox – класс, реализовывающий представление диалогового окна, оповещающего об окончании игры. Его функция заключается в том, чтобы вывести информацию о том, какой игрок победил. Содержит одну фиктивную кнопку, просто закрывающую этот фрейм.

Ниже предоставлены скриншоты, демонстрирующие работу проекта Backgammon, являющегося фактической реализацией слоя представления:

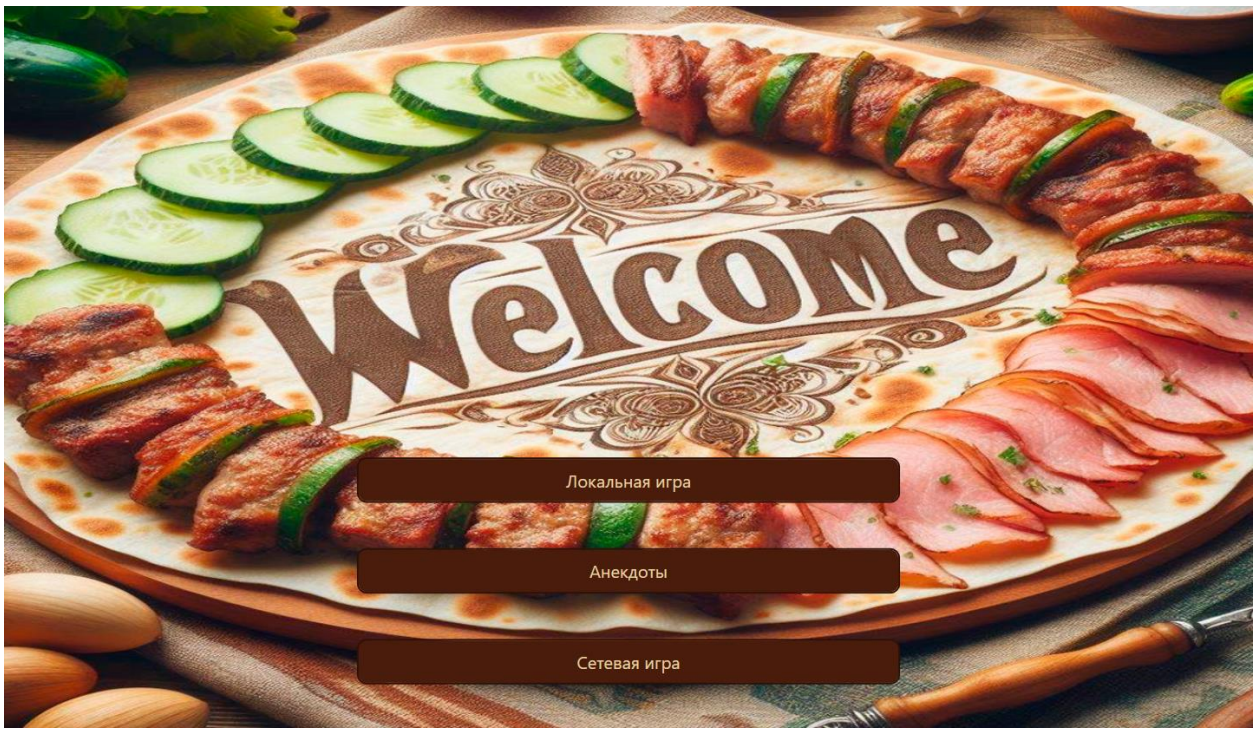

Рисунок 3.1.1 – Страница главного меню

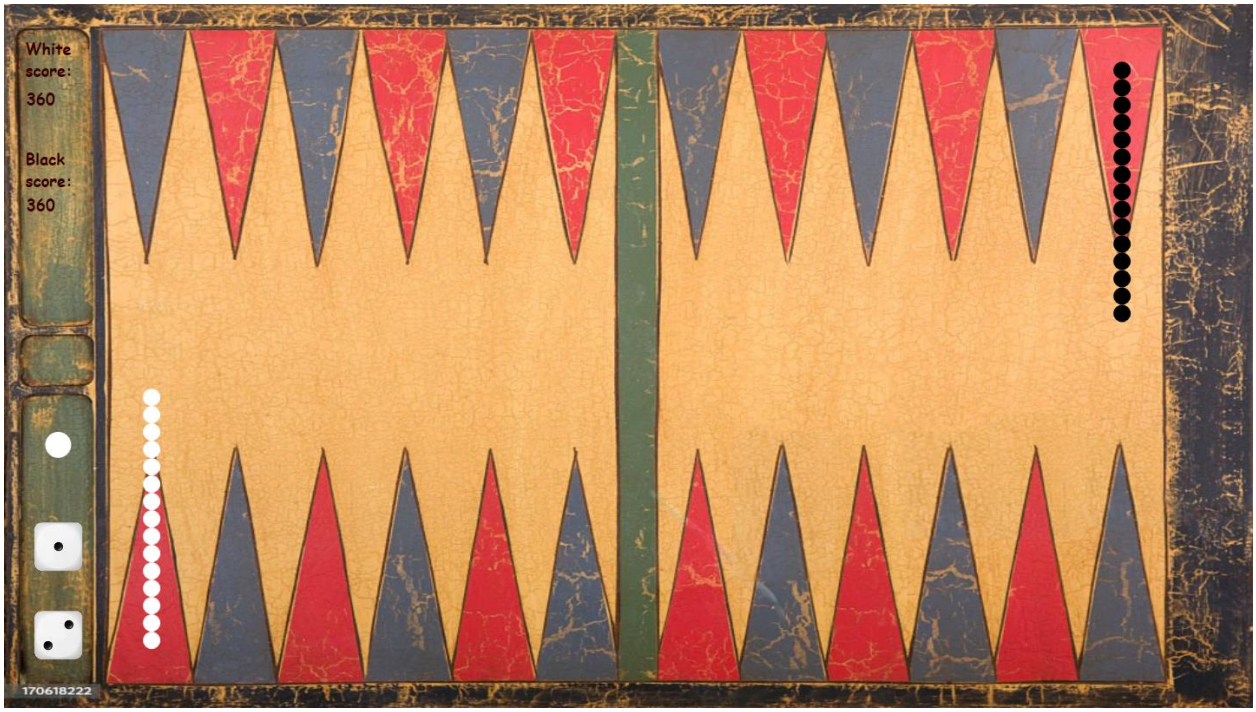

Рисунок 3.1.2 – Страница самой игры

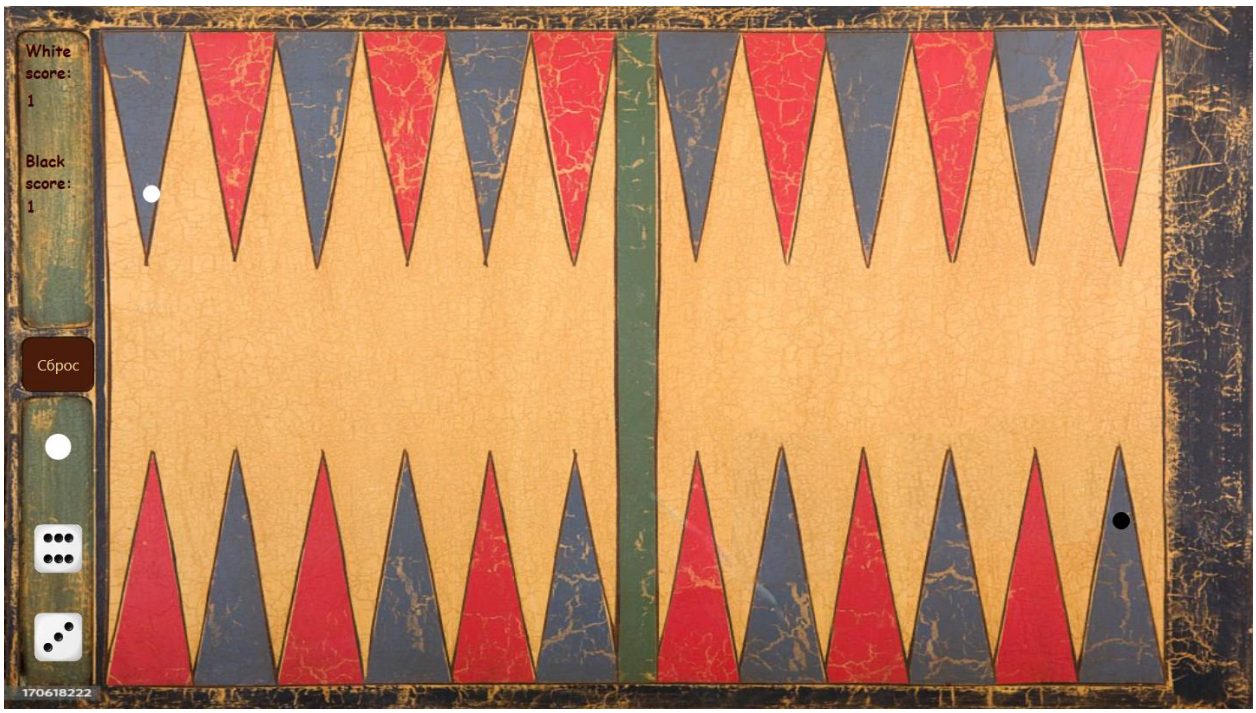

Рисунок 3.1.3 – Финальная фаза игры с активной кнопкой сброса

Рисунок 3.1.4 – Сообщение о победе одного из игроков

3.2 Разработка уровня бизнес-логики

Данный слой реализуется единственным классом – *Game*. Данный класс представляет собой сам игровой процесс. Он содержит в себе соответствующие представления всех моделей проекта и осуществляет их взаимодействие между собой в соответствии с правилами, изложенными во введении. Данный слой реализует исключительно логику и является промежуточным между слоем представления и слоем моделей, что обеспечивает слоеную архитектуру проекта.

Среди сущностей, содержащихся в объекте класса *Game,* можно выделить:

1. Массив игроков
2. Текущего игрока
3. Ссылку на текущее поле
4. Объект модели игровой доски
5. Значения брошенных кубиков
6. Значения возможных ходов
7. Флаг снятой с головы фишки
8. Счетчик совершенных ходов

Среди же методов можно выделить следующие:

1. Обновить внутриигровые данные в начале нового хода
2. Проверка запроса на назначение стартовой позиции
3. Проверка существования валидных ходов
4. Проверка валидности запрашиваемого хода
5. Осуществление хода
6. Осуществление сброса
7. Осуществление следующего хода
8. Получение кратких данных об игровом поле
9. Получение расширенных данных об игровом поле
10. Проверка окончания игры
11. Бросить кубики
12. Получения различного рода данных о состоянии игры

3.3 Разработка уровня данных

Данный слой представлен проектом *Entities.* В нем реализован набор сущностей, необходимых для осуществления бизнес-логики игры и представления игровых объектов в виде моделей, хранящих необходимые данные. Среди таких моделей можно выделить:

1. *Piece* – представляет собой модель шашки. Содержит цвет шашки и ее позицию на игровом поле.

2. *Player* – представляет собой модель игрока. Содержит такую информацию об игроке как цвет, очки, флаг достижения финальной фазы игры, выведения ходя бы одной шашки в последнюю четверть.

3. *Cell* – представляет модель игровой ячейки. Содержит информацию о содержащемся в ячейке наборе шашек. Содержит методы, позволяющие помещать представления шашек в представление ячейки и убирать их из него. Кроме того, позволяет получить информацию о количестве и цвете содержащихся в ячейке шашек.

4. *GameBoard* – модель игровой доски. Содержит два представления о расположении шашек на игровой доске: для черного игрока и для белого игрока. А также набор всех шашек, задействованных в игре.

4 ТЕСТИРОВАНИЕ ПРОЕКТА

Осуществлялось функциональное тестирование. Результаты проведенного тестирования приведены в таблице 4.1.

Таблица 4.1 – Тестирование программного средства

№	Тестируемая функция	Ожидаемый результат	Полученный результат
1	Переход между страницами	При переходе со страницы меню на страницу игры и со страницы игры на страницу меню вызываемые страницы, отображаются корректно	Соответствует ожидаемому
2	Отображение диалогового окна	При сработке условия в код-бихайнд страницы игры должна вызваться функция, генерирующая объект окна. В диалоговом окне должна корректно отображаться информация о победившем игроке.	Соответствует ожидаемому
3	Взаимодействие с диалоговым окном	После нажатия на кнопку «Отлично» Диалоговое окно должно сворачиваться, а затем должен выполниться переход на страницу главного меню.	Соответствует ожидаемому
4	Выбор первой позиции для хода	При выборе первой позиции для хода позиция должна быть успешно выбрана, если в выбранной позиции содержится шашка цвета текущего игрока, а первая позиция ещё не была выбрана, в противном случае информация о первой выбранной позиции не должна измениться.	Соответствует ожидаемому
5	Выбор второй позиции для хода	Вторая позиция должна быть выбрана, если она является пустой, либо заняты шашками цвета текущего игрока, а также если расстояние от первой выбранной позиции до второй выбранной позиции составляет значение одной из костей кубика	Соответствует ожидаемому
6	Проверка наличия	В начале каждого хода должна осуществляться проверка, если ли	Соответствует ожидаемому

	валидных ходов	доступные ходы для игрока текущего цвета хода. Если да, то пользовательский интерфейс должен ожидать действий от пользователя, если нет, совершается новый ход до тех пор, пока не появятся доступные фишки.	
7	Появление кнопки сброса	Кнопка сброса должна быть доступна и видима только в том случае, если игрок вывел все свои шашки в свой дом. В противном случае она не должна отображаться.	Соответствует ожидаемому
8	Корректная работа кнопки сброса	При нажатии на кнопку сброса, должны происходить изменения в соответствующих коллекциях данных, ранее хранивших сброшенную шашку, сама шашка должна исчезнуть. Кроме того, должен измениться игровой счет.	Соответствует ожидаемому
9	Обработка окончания игры	При достижении нуля счетом какого-либо из игроков должен сработать метод обработки окончания игры.	Соответствует ожидаемому
10	Проверка корректности хода	При проверке корректности хода должны быть корректно учтены значения на костях, начальная позиция хода, конечная позиция хода, информация об этих страницах. В результате должен быть получен корректный ответ о валидности данного хода.	Соответствует ожидаемому

ЗАКЛЮЧЕНИЕ

В результате работы над курсовым проектом было разработано программный продукт предоставляющий функционал для игры в длинные нарды на базе операционной системы *Windows*.

Реализация приложения включала в себя несколько ключевых этапов: анализ требований и проектирование, разработка и реализация, тестирование и отладка.

На первом этапе были определены основные функциональные требования к приложению, такие как перечень программных средств и инструментов для дальнейшей разработки, необходимых моделей, способ представления игровых данных в игре, алгоритм обмена данными между объектами и слоями приложения, *UI* и *UX*. Была определена архитектура будущего проекта в целом.

На втором этапе с использование фреймворка *WPF* и *.NET Framework* были разработаны две библиотеки классов: библиотека сущностей проекта и библиотека бизнес-логики проекта. В соответствии с продуманной в первом пункте архитектурой был разработан пользовательский интерфейс, соответствующий всем заявленным требованиям.

Третий этап, заключающийся во всестороннем тестировании приложения, позволил отладить всю функциональность и подкорректировать все крупные и мелкие недочеты проекта. Финальные тесты, согласно таблице 4.1 доказывают полное соответствие финального программного продукта заявленным требованиям.

Данное приложение было разработано с учетом принципов программирования, таких как ООП и SOLID. В следствии чего продукт конечный продукт получился легко масштабируемым и открытым для дальнейшего совершенствования и использования. Что позволит в дальнейшем расширить его до полноценного коммерческого проекта.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода

Листинг 1. Метод проверки наличия валидных ходов для текущего игрока.

```
public bool MovsAvalibleExist()
{
    if (diceValues.Count > 0)
    {
        List<int> monitoredPositions = GetMonitoredPositions();

        if (hatsOffToYou && monitoredPositions.Contains(0))
            monitoredPositions.Remove(0);

        return monitoredPositions.Any(position
            => diceValues.Any(shift => MoveConfirm(position, position +
shift)));
    }
    return false;
}
```

Листинг 2. Метод проверки валидности запрашиваемого хода

```
public bool MoveConfirm(int source, int destinatioin)
{
    if (destinatioin > 23)
        return true;
    bool destExist = diceValues.Contains(destinatioin - source);
    bool moveForvard = source < destinatioin;
    bool isFree = status[destinatioin] == 0;
    bool capturedByFriendlyUnit = status[destinatioin] == curPlayer.Color;

    return (destExist && moveForvard && (isFree || capturedByFriendlyUnit));
}
```

Листинг 3. Метод непосредственного осуществления хода

```
public void Move(int source, int destination)
 {
     if (destination == 25)
     {
         destination = ThrowOut(source);
         Refresh(destination, source, true);
     }
     else
     {
         var movingPiece = curField[source].Pop();
         curField[destination].Push(movingPiece);

         if (!hatsOffToYou && source == 0)
```

```
            hatsOffToYou = true;
         Refresh(destination, source);
     }

 }
```

Листинг 4. Метод осуществления нового хода

```
public void NewTurn()
{
    do
    {
        hatsOffToYou = false;
        curPlayer = curPlayer.Color == 1 ? players[1] : players[0];
        curField = curField == board.BlackField ? board.WhiteField :
board.BlackField;
        StatusRefresh();
        RollDices();
        MoveValuesRefresh();
    } while (!MovsAvalibleExist());

}
```

Листинг 5. Метод обновления графических элементов представления

```
private void Refresh()
{
    List<(int, int)> positionsInfo = game.GetDetailedReport();
    for (int i = 0; i < 24; ++i)
    {
        StackPanel stackPanel = (StackPanel)FindName($"S{i}");
        stackPanel.Children.Clear();
        for (int j = 0; j < positionsInfo[i].Item2; ++j)
        {
            Ellipse piece = new Ellipse();
            piece.Width = 20;
            piece.Height = 20;

            if (positionsInfo[i].Item1 == 1)
                piece.Fill = Brushes.White;
            else
                piece.Fill = Brushes.Black;

            stackPanel.Children.Add(piece);
        }
    }

    try
    {
        var diceValues = game.GetDiceValues();

        Image dice1 = (Image)FindName("DiceOneImage");
        if (dice1 != null)
        {
            string firstDiceImagePath =
$"pack://application:,,,/Sprites/dices/dice{diceValues[0]}.png";
```

```
            Uri imageUri = new Uri(firstDiceImagePath, UriKind.Absolute);
            dice1.Source = new BitmapImage(imageUri);
        }

        Image dice2 = (Image)FindName("DiceTwoImage");
        if (dice2 != null && diceValues.Count > 1)
        {
            string secondDiceImagePath =
$"pack://application:,,,/Sprites/dices/dice{diceValues[1]}.png";
            Uri imageUri = new Uri(secondDiceImagePath, UriKind.Absolute);
            dice2.Source = new BitmapImage(imageUri);
        }
        else if (dice2 != null)
        {
            dice2.Source = null;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error setting image source: {ex.Message}");
    }

    Label whiteScoreLabel = (Label)FindName("WhiteScoreLabel");
    whiteScoreLabel.Content = game.GetScore().Item1.ToString();

    Label blackScoreLabel = (Label)FindName("BlackScoreLabel");
    blackScoreLabel.Content = game.GetScore().Item2.ToString();

    Ellipse currentMoveIndicator = (Ellipse)FindName("CurrentMoveIndicator");
    currentMoveIndicator.Fill = game.GetCurColor() == Entities.Colors.White()
? Brushes.White : Brushes.Black;

    if (game.CheckEndGame())
    {
        EndGame();
        return;
    }

    if (!game.MovsAvalibleExist())
    {
        game.NewTurn();
        Refresh();
    }
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Примеры диаграмм алгоритмов, используемых в программном средстве