МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра информационных технологий управления

Научно-исследовательская работа

“Искусственный интеллект в шашках”

Направление 09.03.02 Информационные системы и технологии

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_д.ф-м.н., проф. Махортов С.Д. \_\_.\_\_.2023

*Подпись, расшифровка, ученая степень, звание*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Родионов Р.И.\_\_\_\_\_\_\_

*Подпись, расшифровка подписи*

Научный руководитель\_\_\_\_\_\_\_\_\_\_ Соломатин Д.И.\_\_\_\_\_\_\_\_\_\_\_\_\_

Воронеж2023

Содержание

[Введение 3](#_Toc137642521)

[1. Теоретический обзор 4](#_Toc137642522)

[1.1 Обзор искусственного интеллекта 4](#_Toc137642523)

[2.2 История и развитие ИИ в настольных играх 5](#_Toc137642524)

[1.3 Основы игры в шашки и ее правила 6](#_Toc137642525)

[1.4 Обзор алгоритма минимакса и его применение в играх 7](#_Toc137642526)

[2. Применение алгоритма минимакса в шашках 9](#_Toc137642527)

[2.1 Описание алгоритма минимакса в контексте игры в шашки 9](#_Toc137642528)

[2.2 Процесс реализации алгоритма минимакса 10](#_Toc137642529)

[2.3 Оптимизация алгоритма минимакса с использованием альфа-бета отсечения 10](#_Toc137642530)

[3. Разработка искусственного интеллекта для игры в шашки 12](#_Toc137642531)

[3.1 Описание архитектуры программного решения 12](#_Toc137642532)

[3.2 Реализация алгоритма минимакса в программном решении 13](#_Toc137642533)

[Заключение 16](#_Toc137642534)

[Список использованной литературы 17](#_Toc137642535)

# **Введение**

Искусственный интеллект (ИИ) стал неотъемлемой частью современного мира, проникая в различные сферы жизни, включая игры. Одним из примеров такого применения является использование ИИ в классической игре шашки. Цель данной работы - разработать и реализовать искусственный интеллект для игры в шашки, используя метод минимакса.

Метод минимакса - это алгоритм, который используется в играх с нулевым суммарным исходом, таких как шашки, шахматы и т.д. Этот алгоритм просматривает все возможные варианты ходов и выбирает тот, который минимизирует возможный максимальный ущерб для игрока.

Задачи, которые ставятся в рамках данной работы, включают:

1. Изучение основ искусственного интеллекта и метода минимакса.

2. Разработка алгоритма для игры в шашки, используя метод минимакса.

3. Реализация алгоритма в виде программного кода.

4. Тестирование и анализ эффективности реализованного искусственного интеллекта.

Ожидается, что в результате выполнения этих задач будет создан искусственный интеллект, способный эффективно играть в шашки, а также будет получен опыт в области разработки алгоритмов искусственного интеллекта и их реализации.

# **Теоретический обзор**

## **1.1 Обзор искусственного интеллекта**

Искусственный интеллект (ИИ) — это область науки и технологии, которая занимается созданием и развитием машин и программного обеспечения, способного имитировать человеческий интеллект и выполнять задачи, которые обычно требуют человеческого вмешательства. Это включает в себя обучение, понимание естественного языка, распознавание речи и изображений, принятие решений и многое другое.

Искусственный интеллект можно разделить на две основные категории: слабый и сильный. Слабый ИИ, также известный как узкий ИИ, специализируется на выполнении конкретной задачи, такой как рекомендация песен в музыкальном приложении или предсказание погоды. С другой стороны, сильный ИИ, также известный как общий ИИ, обладает способностью понимать, учиться и применять знания в различных контекстах, подобно человеческому интеллекту.

Одним из ключевых элементов ИИ является машинное обучение, подраздел ИИ, который использует статистические техники для обучения компьютеров выполнению задач без явного программирования. Глубокое обучение, подкатегория машинного обучения, использует нейронные сети с большим количеством слоев (отсюда и название "глубокое") для обучения на больших наборах данных.

Искусственный интеллект применяется во многих областях, включая здравоохранение, образование, финансы, транспорт и многое другое. Он используется для автоматизации рутинных задач, предсказания трендов, улучшения точности диагностики, персонализации обучения и многого другого.

Однако, несмотря на все его преимущества, ИИ также вызывает определенные проблемы и вызовы, включая вопросы этики, безопасности, приватности и влияния на рабочие места.

В целом, искусственный интеллект продолжает развиваться и оказывает все большее влияние на нашу повседневную жизнь и общество в целом.

## **2.2 История и развитие ИИ в настольных играх**

Искусственный интеллект (ИИ) играет важную роль в развитии настольных игр. Вот краткий обзор истории и развития ИИ в настольных играх:

Ранние годы (1950-е - 1960-е годы): Искусственный интеллект начал применяться в настольных играх в середине 20-го века. Одним из первых примеров была игра в шахматы, где были разработаны программы для имитации или моделирования человеческого противника. Это было большим прорывом, поскольку шахматы традиционно считались игрой, требующей высокого уровня интеллекта и стратегического мышления.

Прорывы в 1990-е годы: В 1997 году ИИ достиг важной вехи, когда IBM Deep Blue победил мирового чемпиона по шахматам Гарри Каспарова. Это было первым случаем, когда машина победила человека в стандартной игре в шахматы под контролем времени.

Развитие в 2000-е годы: В 2000-е годы ИИ продолжил развиваться и стал использоваться в различных настольных играх, включая Go. В 2016 году Google DeepMind's AlphaGo победил мирового чемпиона по Go, Ли Седоля, в серии матчей. Это было значительным достижением, поскольку Go считается одной из самых сложных настольных игр из-за огромного количества возможных ходов.

Современное развитие: Сегодня ИИ используется в широком спектре настольных игр, от классических, таких как шахматы и Go, до более современных, таких как Magic: The Gathering и Hearthstone. ИИ используется не только для создания более сложных и вызывающих противников, но и для обучения игроков, анализа стратегий и улучшения общего игрового опыта.

Искусственный интеллект продолжает развиваться и внедряться в настольные игры, и мы можем ожидать еще больших прорывов в ближайшем будущем [1].

Современные настольные игры также активно используют ИИ для улучшения игрового опыта. Вот несколько примеров:

Gloomhaven: Это популярная настольная игра, в которой ИИ контролирует врагов. Игроки должны стратегически планировать свои ходы, учитывая предсказуемые действия ИИ.

Magic: The Gathering Arena: В этой цифровой версии популярной карточной игры ИИ используется для создания разнообразных и вызывающих противников, а также для обучения новых игроков.

Hearthstone: Это еще одна популярная цифровая карточная игра, где ИИ играет важную роль. ИИ используется для создания ботов-противников, а также для обучения и тестирования стратегий игроков.

Spirit Island: В этой кооперативной настольной игре ИИ контролирует вражеские силы, и игрокам нужно работать вместе, чтобы противостоять им.

The 7th Continent: Это приключенческая настольная игра, где ИИ контролирует мир игры и события, которые происходят в нем.

Искусственный интеллект в настольных играх продолжает развиваться, и мы можем ожидать еще больших прорывов в ближайшем будущем [2].

## **1.3 Основы игры в шашки и ее правила**

Шашки - это настольная игра для двух игроков, которая играется на шахматной доске 8x8. Цель игры - захватить или "заблокировать" все шашки противника. Вот основные правила игры:

Начало игры: Игра начинается с 12 шашками для каждого игрока, расположенных на черных полях доски. Один игрок имеет светлые шашки, другой - темные.

Ходы: Игроки ходят по очереди, начинает игрок с темными шашками. Шашка может двигаться только вперед на одно поле по диагонали на свободное поле.

Захват: Если соперник находится на соседнем поле по диагонали, и следующее поле за ним свободно, шашка может "прыгнуть" через шашку соперника и захватить ее. Захваченная шашка убирается с доски. Если после захвата возможен еще один захват, он должен быть выполнен.

Превращение в дамку: Если шашка достигает противоположного края доски, она становится "дамкой". Дамка может двигаться вперед и назад на любое количество полей по диагонали и может захватывать шашки соперника с любого расстояния, если поля за ними свободны.

Конец игры: Игра заканчивается, когда один из игроков захватывает все шашки противника или блокирует их так, что они не могут сделать ход. Если ни один из игроков не может захватить шашки или сделать ход, игра объявляется ничейной.

Это базовые правила игры в шашки, но существуют различные вариации, включая международные шашки, английские (или "американские") шашки, русские шашки и т.д., каждая из которых имеет свои уникальные правила и стратегии.

## **1.4 Обзор алгоритма минимакса и его применение в играх**

Алгоритм минимакса - это рекурсивный алгоритм, используемый для решения игр с нулевой суммой (таких как шахматы, шашки и крестики-нолики), где один игрок пытается максимизировать свой выигрыш, а другой - минимизировать его.

Вот основные аспекты алгоритма минимакса:

1. **Дерево игры**: Минимакс использует дерево игры для представления последовательности ходов, начиная с текущего состояния игры. Узлы в дереве представляют состояния игры, а ребра - ходы.

2. **Минимизация и максимизация**: В дереве игры игрок, который пытается максимизировать выигрыш (обычно называемый "Max"), и игрок, который пытается минимизировать его (обычно называемый "Min"), чередуются. Max начинает с корня дерева.

3. **Оценка**: Когда достигается конечное состояние игры (лист дерева), его оценивают с точки зрения Max: выигрыш Max оценивается как +1, проигрыш как -1, а ничья как 0.

4. **Обратное распространение**: Затем эти оценки распространяются обратно по дереву. В узле Max выбирается максимальная оценка из дочерних узлов, а в узле Min - минимальная.

5. **Выбор хода**: В конце концов, Max выбирает ход, который ведет к узлу с максимальной оценкой.

Алгоритм минимакса может быть очень эффективным для игр с небольшим числом возможных ходов, но для игр с большим числом ходов (таких как шахматы) он может стать непрактичным из-за "проклятия размерности". В таких случаях используются различные методы для упрощения задачи, такие как альфа-бета отсечение (которое уменьшает число узлов, которые нужно исследовать), и эвристические функции (которые дают "хорошие" оценки состояний игры без необходимости исследовать все возможные последовательности ходов).

1. **Применение алгоритма минимакса в шашках**

**2.1 Описание алгоритма минимакса в контексте игры в шашки**

Алгоритм минимакса в контексте игры в шашки работает по тому же принципу, что и в других настольных играх, но с учетом специфики шашек. Вот как это работает:

**Создание дерева игры:** Для каждого возможного хода из текущего состояния игры создается дерево игры. В этом дереве узлы представляют состояния игры после каждого возможного хода, а ребра представляют сами ходы.

**Оценка состояний игры:** Когда достигается конечное состояние игры (лист дерева), его оценивают с точки зрения игрока, который пытается максимизировать свой выигрыш. Это может быть, например, количество шашек игрока, количество дамок, количество возможных ходов и т.д.

**Минимизация и максимизация:** Затем эти оценки распространяются обратно по дереву. В узле игрока, который пытается максимизировать свой выигрыш, выбирается максимальная оценка из дочерних узлов, а в узле игрока, который пытается минимизировать его, выбирается минимальная оценка.

**Выбор хода:** В конце концов, игрок, который пытается максимизировать свой выигрыш, выбирает ход, который ведет к узлу с максимальной оценкой.

Важно отметить, что алгоритм минимакса предполагает, что оба игрока играют оптимально. Если это не так, то реальный ход, который выберет противник, может отличаться от предсказанного алгоритмом минимакса.

Также стоит отметить, что для игр с большим числом возможных ходов, таких как шашки, алгоритм минимакса может быть слишком медленным. В таких случаях можно использовать альфа-бета отсечение для ускорения алгоритма, ограничивая глубину поиска или используя эвристические функции для оценки состояний игры.

## **2.2 Процесс реализации алгоритма минимакса**

Реализация алгоритма минимакса включает в себя несколько ключевых шагов. Вот общий процесс:

1. **Определение функции оценки:** Функция оценки используется для оценки "ценности" конкретного состояния игры для игрока. В контексте шашек, она может учитывать такие факторы, как количество шашек у игрока, количество дамок, количество возможных ходов и т.д.

2. **Создание дерева игры:** Дерево игры представляет все возможные ходы из текущего состояния игры. В каждом узле дерева хранится состояние игры, а ребра представляют ходы.

3. **Реализация алгоритма минимакса:** Алгоритм минимакса реализуется как рекурсивная функция, которая обходит дерево игры, вычисляя оценку для каждого узла. В узлах Max выбирается максимальная оценка из дочерних узлов, а в узлах Min - минимальная.

4. **Выбор хода:** После того, как все оценки были вычислены, игрок выбирает ход, который ведет к узлу с максимальной оценкой.

5. **Оптимизация:** Для игр с большим числом возможных ходов, таких как шашки, алгоритм минимакса может быть слишком медленным. В таких случаях можно использовать альфа-бета отсечение для ускорения алгоритма, ограничивая глубину поиска или используя эвристические функции для оценки состояний игры.

Важно отметить, что алгоритм минимакса предполагает, что оба игрока играют оптимально. Если это не так, то реальный ход, который выберет противник, может отличаться от предсказанного алгоритмом минимакса.

**2.3 Оптимизация алгоритма минимакса с использованием альфа-бета отсечения**

Альфа-бета отсечение - это метод оптимизации для алгоритма минимакса, который значительно уменьшает количество узлов, которые необходимо исследовать в дереве игры. Это достигается путем раннего отбрасывания тех ветвей, которые, очевидно, не приведут к лучшему решению.

Вот как это работает:

1. **Альфа:** Это максимальная оценка, которую мы можем гарантировать для игрока Max на данном уровне или ниже.

2. **Бета:** Это минимальная оценка, которую мы можем гарантировать для игрока Min на данном уровне или ниже.

3. **Отсечение:** Если в какой-то момент становится ясно, что есть путь, который приведет к оценке, хуже, чем текущее значение альфа (для Max) или бета (для Min), мы можем "отсечь" оставшуюся часть этой ветви и не исследовать ее, так как мы уже знаем, что она не приведет к лучшему решению.

Альфа-бета отсечение позволяет значительно ускорить алгоритм минимакса, особенно в играх с большим количеством возможных ходов, таких как шашки. Однако, стоит отметить, что эффективность альфа-бета отсечения во многом зависит от порядка, в котором исследуются ходы. Идеально, если мы сначала исследуем наиболее обещающие ходы, так как это увеличит вероятность раннего отсечения.

# **3. Разработка искусственного интеллекта для игры в шашки**

## **3.1 Описание архитектуры программного решения**

Архитектура программного решения включает в себя следующие классы:

**Класс Move:** Этот класс представляет собой ход в игре. Он содержит координаты начальной и конечной точки хода. Класс также предоставляет методы для сравнения ходов и представления хода в виде строки.

**Класс Game:** Этот класс представляет собой основную логику игры. Он содержит методы для анимации ходов, обработки ходов игрока и противника, проверки условий окончания игры и предсказания оптимальных ходов. Этот класс также взаимодействует с классом Canvas для отображения игры.

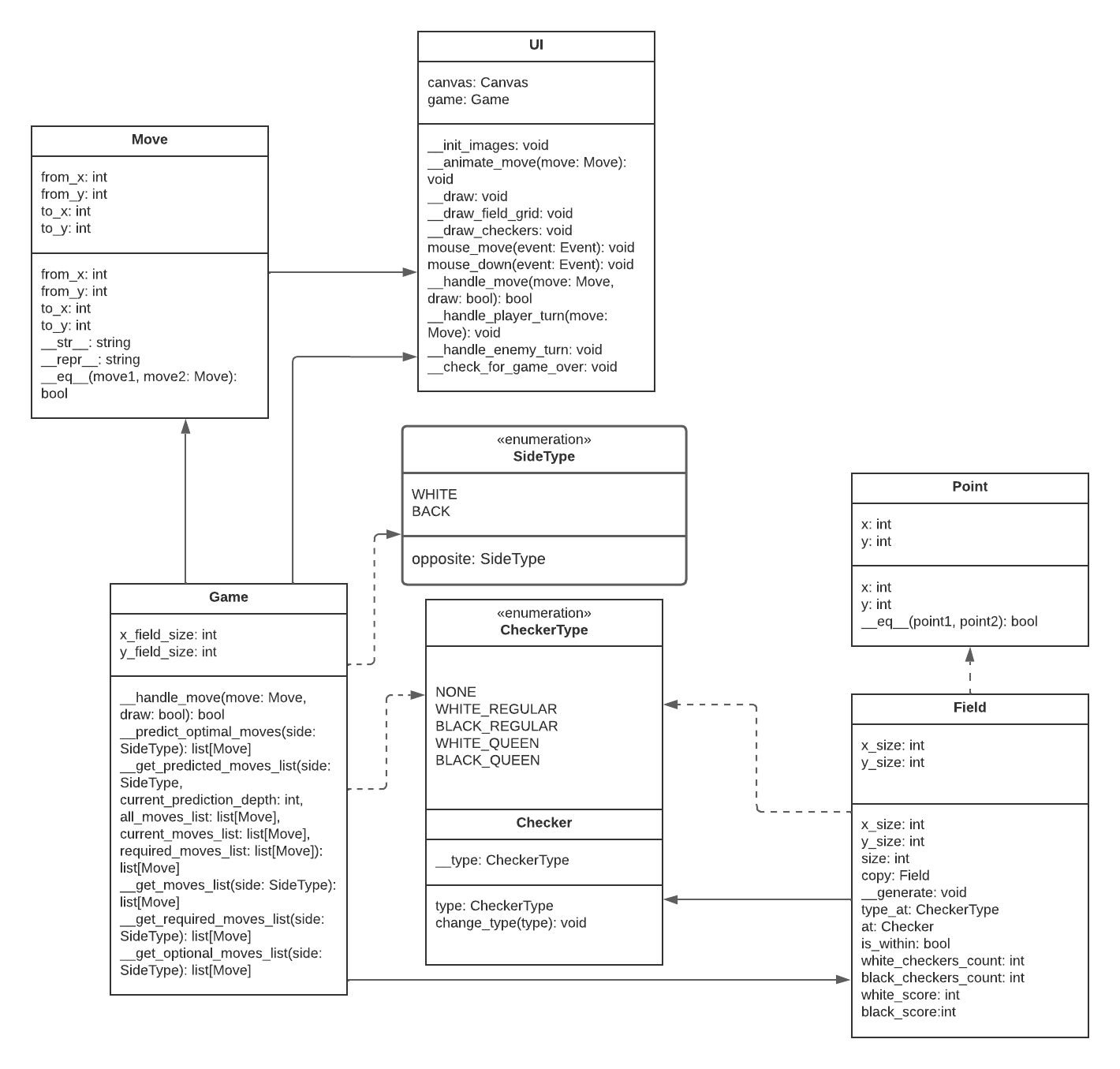
**Класс Checker:** Этот класс представляет шашку в игре. Он содержит тип шашки и метод для изменения этого типа (например, когда шашка становится дамкой).

**Класс Point:** Этот класс представляет точку на игровом поле. Он содержит координаты x и y и метод для сравнения точек.

**Класс Field:** Этот класс представляет игровое поле. Он содержит размер поля и методы для генерации поля, получения типа шашки в определенной точке, проверки, находится ли точка в пределах поля, и подсчета количества шашек каждого типа на поле.

В дополнение к этим классам, диаграмма также показывает два перечисления: SideType и CheckerType, которые представляют тип стороны (белый или черный) и тип шашки (обычная или дамка) соответственно.

Эта архитектура обеспечивает модульный подход к реализации игры в шашки, где каждый класс имеет свою специфическую роль.



## **3.2 Реализация алгоритма минимакса в программном решении**

Данное программное решение использует алгоритм предсказания всех возможных ходов и выбора оптимального хода на основе оценки текущего состояния игрового поля. Для этого используется функция \_\_predict\_optimal\_moves, которая предсказывает все возможные ходы и оценивает их на основе соотношения количества шашек на игровом поле. В данном программном решении, вероятно, используется простая эвристика, согласно которой, чем больше шашек на поле у текущего игрока, тем лучше его позиция. Соответственно, при выборе наилучшего хода для противника, учитывается количество шашек на поле у обоих игроков, и выбирается ход, который максимизирует количество шашек у противника и минимизирует количество шашек у текущего игрока. Однако, это только предположение, и конкретная реализация может отличаться.

def \_\_predict\_optimal\_moves(self, side: SideType) -> list[Move]:  
 *'''Предсказать оптимальный ход'''* best\_result = 0  
 optimal\_moves = []  
 predicted\_moves\_list = self.\_\_get\_predicted\_moves\_list(side)  
  
 if predicted\_moves\_list:  
 field\_copy = Field.copy(self.\_\_field)  
 for moves in predicted\_moves\_list:  
 for move in moves:  
 self.\_\_handle\_move(move, draw=False)  
  
 try:  
 if side == SideType.WHITE:  
 result = self.\_\_field.white\_score / self.\_\_field.black\_score  
 elif side == SideType.BLACK:  
 result = self.\_\_field.black\_score / self.\_\_field.white\_score  
 except ZeroDivisionError:  
 result = inf  
  
 if result > best\_result:  
 best\_result = result  
 optimal\_moves.clear()  
 optimal\_moves.append(moves)  
 elif result == best\_result:  
 optimal\_moves.append(moves)  
  
 self.\_\_field = Field.copy(field\_copy)  
  
 optimal\_move = []  
 if optimal\_moves:  
 # Фильтрация хода  
 for move in choice(optimal\_moves):  
 if side == SideType.WHITE and self.\_\_field.type\_at(move.from\_x, move.from\_y) in BLACK\_CHECKERS:  
 break  
 elif side == SideType.BLACK and self.\_\_field.type\_at(move.from\_x, move.from\_y) in WHITE\_CHECKERS:  
 break  
  
 optimal\_move.append(move)  
  
 return optimal\_move

Именно эта часть кода отвечает за предсказание ходов, и оценку поля, оценка считается как количество шашек на поле, при этом вес королевы в 3 раза выше, и именно на этой оценке и стоит вся суть предсказание, мы копируем наше поле, получаем все возможные ходы и прогоняет, после каждого хода мы считаем отношение счета белых и черных друг к другу в зависимости от стороны, и тот ход, в котором это отношение больше, мы и записываем в массив оптимальных ходов. Также мы регулируем количество просчитываемых ходов, от которого зависит как качество последующих ходов, так и время выполнения.

# **Заключение**

В ходе работы над проектом был реализован искусственный интеллект для игры в шашки. Основой для создания алгоритма стал метод минимакса, который позволил создать систему, способную анализировать возможные ходы и выбирать наиболее оптимальный из них.

Метод минимакса представляет собой алгоритм поиска оптимального хода в играх с нулевым суммарным исходом, таких как шашки, шахматы и т.д. Этот алгоритм просматривает все возможные варианты ходов и выбирает тот, который минимизирует возможный максимальный ущерб для игрока, отсюда и название - минимакс.

Реализация искусственного интеллекта для игры в шашки с использованием метода минимакса позволила создать достаточно сильного противника, способного анализировать ситуацию на игровом поле и принимать решения, максимально выгодные для себя.

Однако стоит отметить, что эффективность использования метода минимакса напрямую зависит от глубины просмотра ходов. Чем больше ходов вперед анализирует алгоритм, тем точнее и оптимальнее будет его решение. Однако увеличение глубины просмотра ведет к значительному увеличению вычислительной сложности, что может быть проблематично для реального времени игры.

В целом, реализация искусственного интеллекта для игры в шашки с использованием метода минимакса показала свою эффективность и может быть использована как основа для дальнейшего усовершенствования и развития системы.

# **Список использованной литературы**

1. <https://www.researchgate.net/publication/344417822_History_and_Development_of_Artificial>
2. <https://www.gamasutra.com/view/news/365995/DeepMind_turns_its_AI_to_playing_Magic_The_Gathering_and_Hearthstone.php>