СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ИИ – искусственный интеллект

BT – Behavior Tree

FSM – Finite State Machine

GA – Generic Algorithm

GOAP – Goal-oriented Action Planner

HTN – Hierarchical Task Network

MARL – Multi-Agent Reinforcement Learning

MCTS – Monte-Carlo Tree Search

NPC – Non-Player Character

RL – Reinforcement Learning

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Агент – персонаж, управляемый искусственным интеллектом

Восприятие – способность агента получать информацию об окружающих его среде и объектах.

Коммуникация – способность агента передавать и получать информацию от других агентов, а также взаимодействовать с ними

NPC – персонаж игрового мира, не управляемый игроком.

ВВЕДЕНИЕ

Существует малое количество игр, в котором уделено должное внимание разуму животных, в том числе коллективному. Например, животные в Metal Gear Solid 5: The Phantom Pain (Kojima Productions, 2015) играют большую роль в основной игровом цикле и поэтому обладают уникальным поведением – они не будут нападать на главного героя на прямую, а будут маневрировать вокруг, уворачиваясь от выстрелов, пока не поймают удачный момент для атаки.

Другая игра, в которой существует нетипичное для других игр поведение животных – это Rise Of the Tomb Raider (Crystal Dynamics, 2015). В ней животное обладает уникальным поведением в зависимости от его размера, например, белки начнут убегать, если главная героиня выстрелит рядом с ними, а медведь, наоборот, нападет на нее.

Последним примером может являться игра Avatar: Frontiers of Pandora (Massive Entertainment, 2023), в которой есть открытый мир, а в нем сотни животных, которые бродят по этому миру. В этой игре можно увидеть, как одни группы животных вместе охотятся в лесу, когда другие, большие животные, образовывают маленькие семьи, чтобы вместе передвигаться по миру.

Во многих играх животные очень сильно влияют на погружение игрока в атмосферу, поэтому изучение и построение более продвинутого искусственного интеллекта животных является актуальной проблемой в игровой индустрии. Однако построение

такого искусственного интеллекта животных требует больших усилий и времени.

Данная работа направлена на моделирование коллективного поведения животных в разных игровых ситуациях. Для проверки построенной модели используется симуляция прайда львов.

Объектом исследования были выбраны методы реализации искусственного интеллекта в играх, а предметом исследования – коллективный разум животных.

Цель работы – смоделировать коллективный разум животных в играх.

Для достижения поставленной цели были сформулированы следующие задачи:

1) провести анализ существующих подходов создания ИИ,
2) разработать теоретическую модель,
3) создать прототип разработанной модели,
4) провести тестирование модели и проанализировать результаты.

Структурно выпускная квалификационная работа состоит из введения, трех основных глав, заключения, списка использованных источников.

В первой главе выпускной квалификационной работы рассмотрены существующие методы реализации ИИ, описаны их преимущества и недостатки. Также предложен метод, который можно взять за основу модели.

Во второй главе выпускной квалификационной работы описаны необходимые системы для построения ИИ, их взаимодействие друг с

другом. Построена модель коллективного разума животных. Проведено исследование социальной структуры и поведения львов в естественной среде.

В третьей главе выпускной квалификационной работы описан разработанный прототип построенной модели, приведены сценарии тестирования. Проведены испытания работы и производительности алгоритма. Также был проведен анализ полученных результатов.

1 Обзор существующих методов

В 1970 гг. примитивный ИИ стал применяться в видеоиграх, таких как Speed Race (Taito. 1974), Space Invaders (Atari. 1978) и Galaxian (Namco, 1979). Позднее, с разработкой более мощных устройств, стали появляться более продвинутые реализации ИИ для применения в видеоиграх, такие как конечный автомат и дерево поведений [1]. С тех пор идет активное развитие ИИ в области видеоигр. Были созданы различные подходы, например, планировщики действий, а также множество алгоритмов, основанных на машинном обучении [2]. Все они могут использоваться для основы коллективного ИИ животных. Далее более подробно описаны несколько часто используемых алгоритмов.

1.1 Конечный автомат

Конечный автомат (FSM или Finite State Machine) — это простой и быстрый способ реализовать ИИ. Этот подход применяется в таких играх как Pac-Man (Nintendo. 1980) и Half-Life (Valve. 1998).

FSM, представленный на Рисунке 1, подразумевает описание некоторого количества состояний и переходов между ними, а затем программирование действий для каждого из этих состояний и правил для перехода в другое состояние [3].

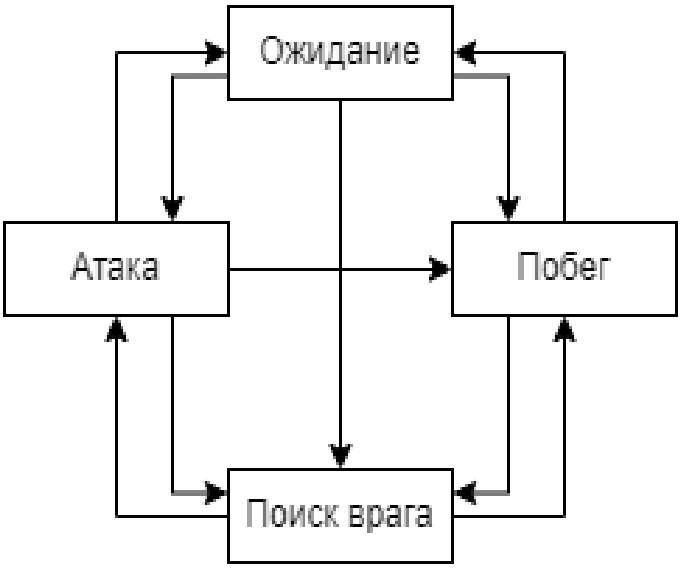

Рисунок 1 – Конечный автомат

При увеличении количества состояний, растет и количество переходов между ними, что приводит к излишней сложности алгоритма и трудности долгосрочной поддержки. Однако эта проблема может быть облегчена с помощью использования иерархического конечного автомата (HFSM или Hierarchical Finite State Machine) [4]. Пример HFSM представлен на Рисунке 2. В этом методе создается несколько конечных автоматов, выстроенных в иерархию. Это дает больше гибкости, но это не решает основную проблему конечных автоматов — предсказуемость действий ИИ.

1.2 Дерево поведений

Дерево поведений (BT или Behavior Tree) является другим популярным подходом. Он успешно применяется во множестве игр, например, Spore (Electronic Arts. 2008), Bioshock (2K Games. 2007) и Halo (Xbox Games Studios. 2001).

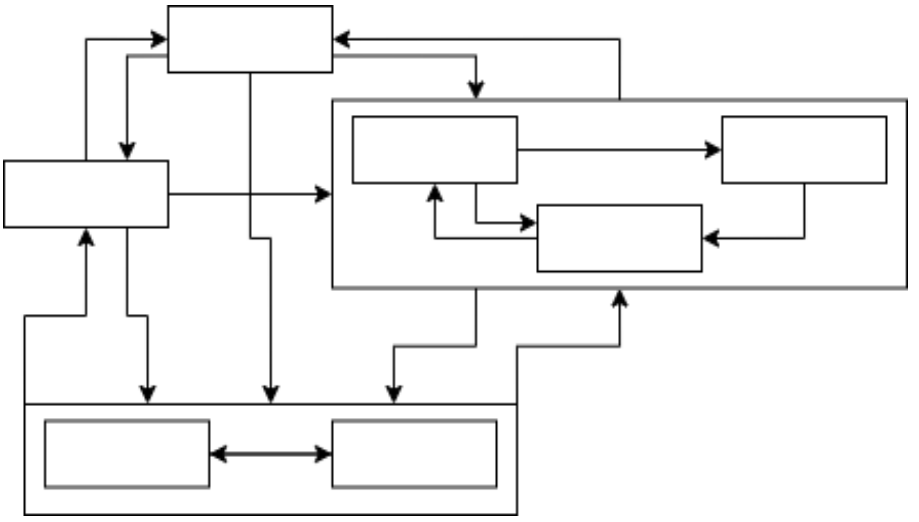

Рисунок 2 – Иерархический конечный автомат

Дерево поведений — это однонаправленный ациклический граф, где узлы или задачи (поведения) представляют из себя действия, совершаемые агентом ИИ [5].

У каждого узла может быть несколько состояний:

– «успех» — задача успешно завершилась,

– «неудача» — не были выполнены условия, для начала работы этой задачи, либо задача завершилась с ошибкой,

– «в процессе» — задача выполняется в данный момент.

Дерево выполняет поиск в глубину, чтобы найти действие для исполнения. При наличии у узла подзадач, они выполняются слева направо. При завершении задачи ее результат передается родительскому узлу.

Дерево поведений может состоять из задач нескольких типов:

– селектор (selector) — возвращает успех, когда один из дочерних узлов вернул успех,

– последовательность (sequence) — возвращает успех, когда все дети вернули успех,

– параллельный узел (parallel) — выполняет несколько задач параллельно, пока некоторое количество этих задач не вернет свой статус,

– инвертер (inverter) — возвращает неудачу, если дочерняя задача выполнилась с успехом и наоборот,

– действие (action) — это основная логика. Агент ИИ выполняет заданные действия.

Для каждой задачи есть возможность указать предусловия, при выполнении которых можно начать выполнение этой задачи. При возвращении результата в корень дерева, проход начинается заново. Пример BT представлен на Рисунке 3.

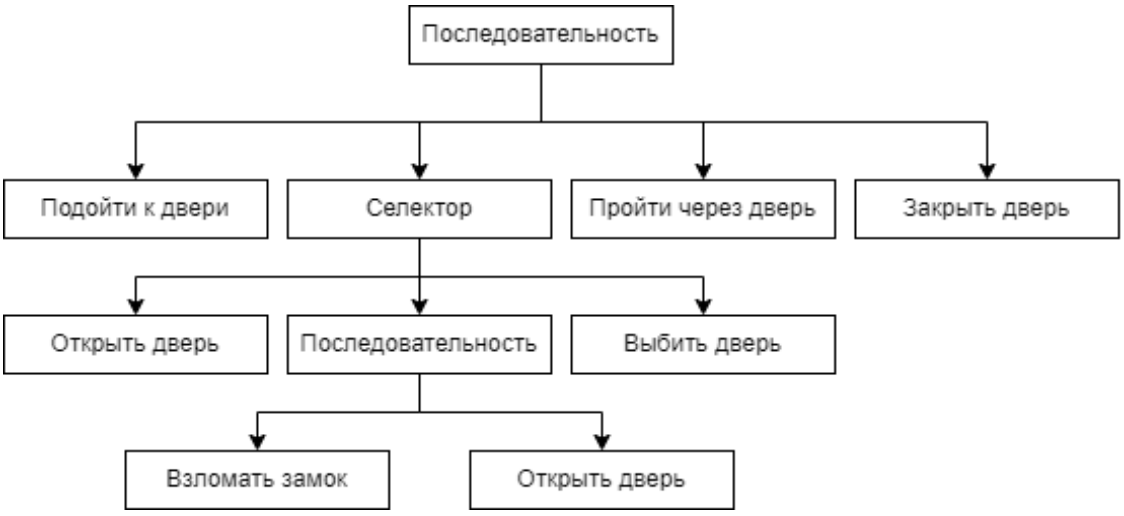

Рисунок 3 — Дерево поведений

Дерево поведений — это удобный инструмент для создания правдоподобного ИИ, однако само дерево программируется заранее, что дает предсказуемость действий и исключает возможность

адаптивности. Эту проблему попробовали решить в [6] с помощью генетического алгоритма, однако этот метод требует больших временных затрат для тренировки нейросети и генерации дерева.

1.3 Целеориентированное планирование действий

Одной из популярных систем ИИ является целеориентированное планирование действий (GOAP или Goal-Oriented Action Planning). Этот подход впервые реализован в игре F.E.A.R. (Monolith Productions. 2005), в которой ИИ до сих пор считается одним из лучших в шутерах от первого лица. Также GOAP применяется в Tomb Raider (Square Enix. 2001), Halo 2 (Microsoft Game Studios. 2004) и Just Cause (Square Enix. 2006).

Планировщик позволяет построить план действий, который удовлетворяет определенной цели [7]. У каждого агента ИИ есть свой определенный набор целей. У каждой цели присутствует коэффициент необходимости. Чем больше этот коэффициент, тем больше шанс, что агент выберет именно эту цель. Также у каждого агента есть набор действий, которые направлены на уменьшение коэффициента необходимости (далее — коэффициент) определенной цели. Например, когда у NPC полное количество очков здоровья, коэффициент цели «Лечение» имеет низкое значение, а когда он получает урон и очки здоровья уменьшаются, коэффициент возрастает. Затем NPC выбирает действия, которые способны уменьшить коэффициент цели «Лечение», а затем предпринимает самое выгодное действие, например, воспользоваться аптечкой и восстановить очки здоровья.

Для создания плана, GOAP строит граф всевозможных вариантов исполнения цели [7]. Узлы графа являются состояниями мира, а соединения узлов – действиями, которые агент ИИ может предпринять. Состояние мира хранит в себе переменные для определения условий выполнения действий, например, текущее количество очков здоровья. Для нахождения наилучшего решения применяется алгоритм нахождения пути A* [8]. Пример построенного графа изображен на Рисунке 4.

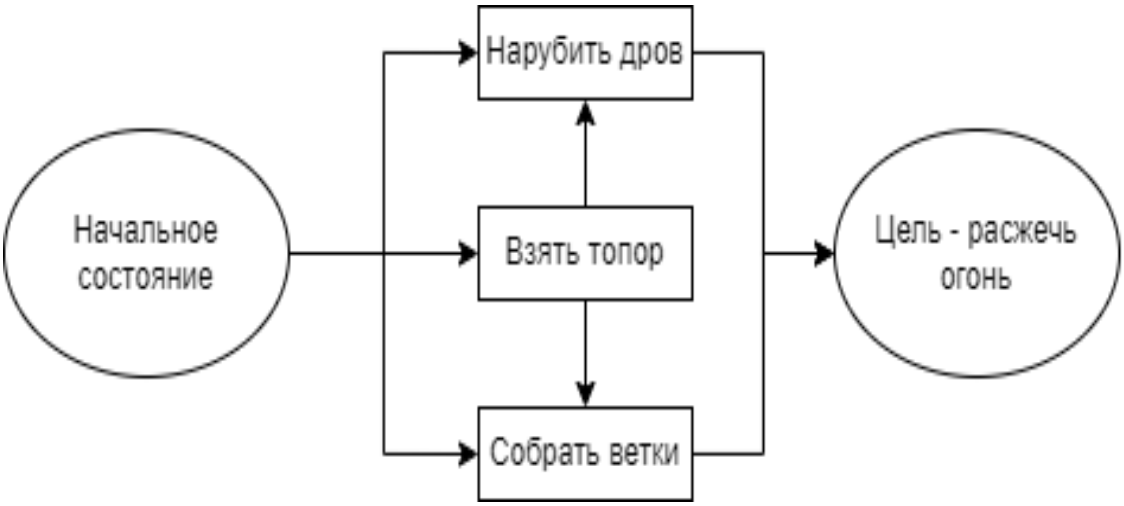

Рисунок 4 — Граф построенный GOAP

GOAP так же, как и другие методы, нацелен на поведение одного агента. Для решения этой проблемы, авторами статьи [9] был предложен мульти-агентный метод реализации GOAP. Однако этот метод все еще не подразумевает адаптивного поведения агентов.

1.4 Иерархический планировщик сети задач

Иерархический планировщик сети задач (HTN или Hierarchical Task Network) — это смесь подходов BT и GOAP. Данный подход применяется в Transformers: Fall of Cybertron (High Moon Studios.

2012), а также в Horizon Zero Dawn (Sony Interactive Entertainment. 2017).

Этот подход предполагает построение дерева с множеством поддеревьев единичной высотой и нахождение наилучшего плана для текущего состояния мира [10].

У искусственного интеллекта, построенного на базе планировщика HTN планирования, имеется две фазы работы [11], рассмотрим их подробнее.

Первая фаза — планирование. Планировщик состоит из нескольких частей: состояние мира, простые и составные задачи, методы и операторы. Состояние мира, так же, как и в GOAP, хранит переменные состояния агента ИИ.

Простая задача состоит из оператора — действия, которое может совершить агент ИИ некоторым количеством эффектов и предусловий. Эффекты влияют на состояние мира. Чтобы задача могла быть выполнена, должны быть соблюдены ее предусловия. Примером простых задач являются задачи “Преследовать игрока” и “Спрятаться в укрытии”, эти две задачи содержат один оператор “Идти”, но разные эффекты и предусловия.

Составные задачи состоят из одного или нескольких методов, которые представляют собой варианты решения этой задачи. При разбиении составной задачи, выбирается метод, который удовлетворяет заданным предусловиям. Пример составной задачи представлен на Рисунке 5.

Так же, как и BT, планировщик проходится по узлам слева направо.

Результатом работы планировщика является последовательность действий, которые передаются в следующую фазу — выполнение плана.

Задачи, поставленные на исполнителя плана, влияют на глобальное состояние мира, а также состояние самого агента ИИ. Переход обратно в первую фазу происходит в нескольких случаях:

– план полностью выполнен,

– невозможно выполнить запланированные действия.

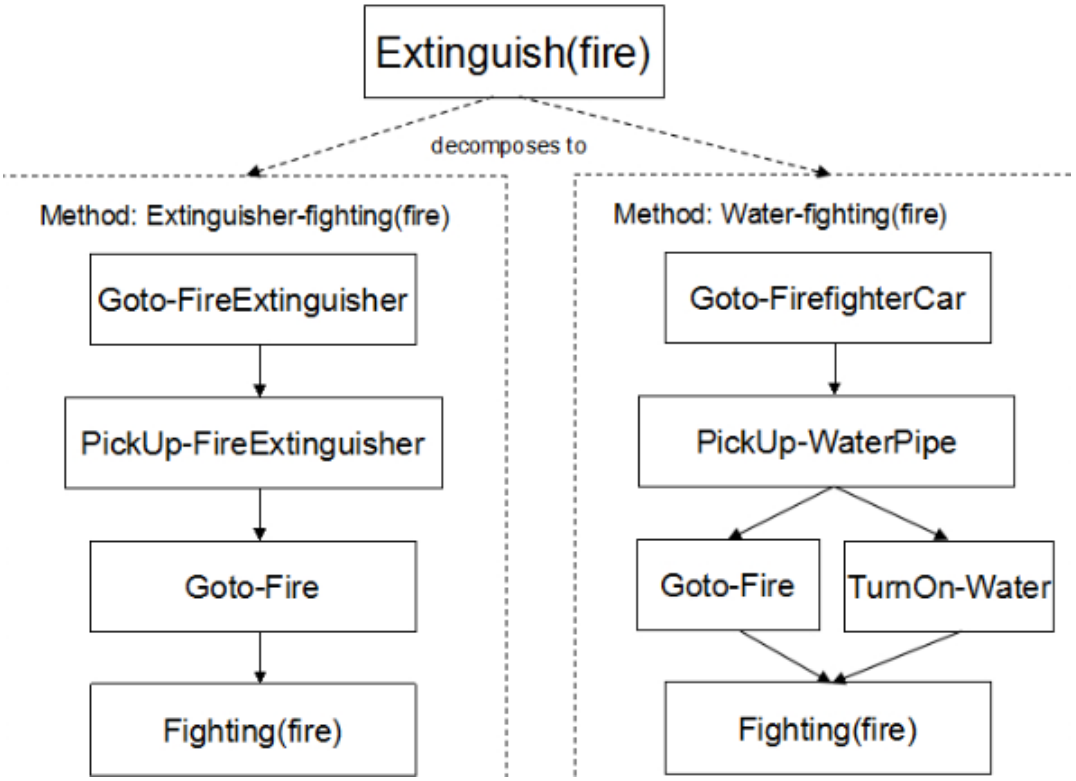

Рисунок 5 — Пример составной задачи [10]

Классическая реализация алгоритма HTN имеет большую стоимость планирования, поэтому в [12] авторы предложили идею о

повторном использовании уже построенного плана, что улучшает производительность алгоритма.

Планировщик HTN позволяет агенту ИИ совершить наиболее подходящие действия. Однако классический алгоритм не подразумевает адаптацию под действия игрока.

1.5 Поиск по дереву Монте-Карло

Поиск по дереву Монте-Карло (MCTS или Monte Carlo Tree Search) является популярным выбором для создания адаптивных игровых агентов, поскольку он способен обрабатывать большие пространства поиска и эффективно сочетать исследование и эксплуатацию [13]. В условиях игры MCTS работает, создавая дерево поиска возможных ходов и используя моделирование и оценку, чтобы направлять поиск к наиболее многообещающим ходам. На каждом шаге MCTS последовательно выполняет несколько действий, представленных на Рисунке 6. Подробное описание представлено ниже:

Выбор: MCTS выбирает путь через дерево поиска, следуя наивысшей верхней доверительной границе (ВДГ) на каждом шаге. ВДГ учитывает, как количество посещений узла, так и качество его дочерних элементов, поощряя поиск новых, многообещающих узлов, при этом используя преимущества известных хороших ходов;

Расширение: MCTS расширяет дерево, добавляя к выбранному узлу одного или нескольких дочерних элементов. Это позволяет MCTS исследовать новые области пространства поиска;

Моделирование: MCTS имитирует воспроизведение с выбранного узла с помощью политики моделирования, которая может

быть либо фиксированной политикой (например, эвристика на основе правил), либо изученной политикой (например, нейронной сетью);

Обратное распространение: MCTS обновляет дерево путем обратного распространения результатов моделирования по дереву, обновляя счетчики и значения каждого узла на пути. Это позволяет MCTS уточнять свои оценки ценности каждого хода и направлять поиск в более перспективные области пространства поиска.

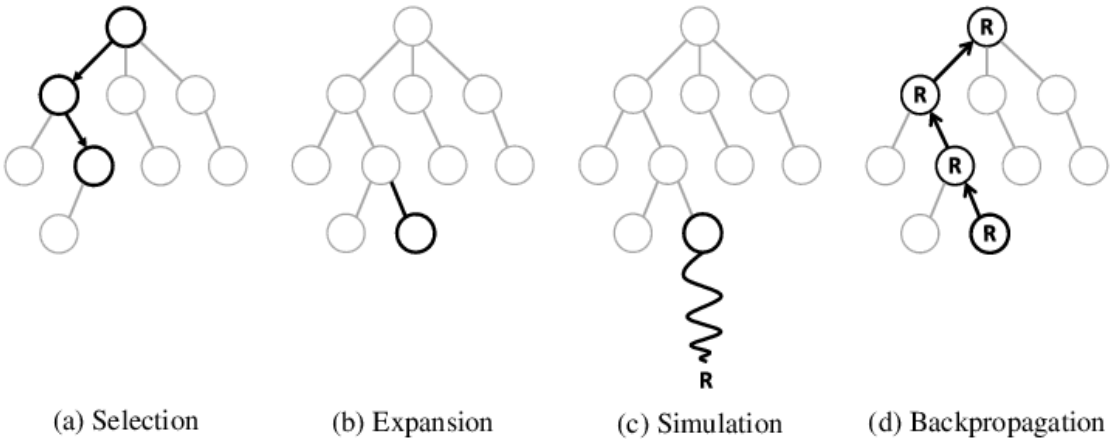

Рисунок 6 – Шаги выполнения MCTS. (a) Выбор; (b) Расширение; (c) Моделирование; (d) Обратное распространение [14]

В целом, применение MCTS в качестве адаптивного ИИ в играх позволяет агенту адаптироваться к изменяющемуся состоянию игры и со временем находить новые эффективные стратегии. Например, авторы [14] настроили MCTS для динамического изменения сложности игры.

1.6 ИИ на основе полезности

Относительно новым и эффективным подходом создания ИИ является система на основе полезности (Utility-based System или Utility-based AI). Этот способ реализован в таких играх как The Sims

(Electronic Arts. 2000), X-COM Enemy Unknown (2K Games. 2012) и League of Legends (Riot Games. 2009).

Теория полезности используется в сфере экономики и психологии с XIX-XX вв. [15]. Этот подход предполагает определение полезности действий и дальнейший выбор наиболее полезного их них. Похожим образом мыслит и человек, поэтому данный подход приобрел большую популярность среди разработчиков искусственного интеллекта.

Термин Utility-based AI используется для описания класса техник, в которых решения принимаются на основе эвристических функций, которые представляют собой относительную величину (или подходимость) каждой опции при оценке в рамках числа с плавающей запятой. ИИ на основе полезности обычно имеет 3 основных шага [16]:

1) построение списка опций, из которых будет происходить выбор,
2) вычисление одного или больше чисел с плавающей запятой, которые описывают насколько привлекательна эта опция в данной ситуации. Эти значения называют по-разному: полезность, приоритет, вес, ранг, важность и т.д.
3) выбор опции (или набора опций) для выполнения на основе посчитанных значений на шаге 2.

Ключевым фактором является то, что шаг 2 должен выполняться в режиме реального времени. Другими словами, значения полезности не выбираются при разработке сценария и не являются фиксированными. Они вычисляются в реальном времени на

основе данных в текущей ситуации в определенный момент времени. Следовательно, Utility-based AI постоянно пересчитывает значения и выбирается наилучшую опцию или опции в каждый момент времени.

Использование чисел с плавающей запятой вместо булевых значений позволяет иметь более детализированную картину. Вместо того, чтобы иметь несколько значений из которых можно выбирать (набор «да или нет» проверок), появляется возможно выразить вещи полностью непрерывными способами.

Данный метод эффективно показывает себя при большом количестве возможных действий, так как для них не требуется выстраивать правила перехода, что в результате позволяет проще расширять систему. Также к плюсу можно отнести тот факт, что агент на основе Utility-based AI может справляться с ситуациями, не предусмотренными разработчиком. Еще одно преимущество данного подхода заключается в использовании маленьких самодостаточных блоков – действий или оценок, что позволяет их переиспользовать в разных сценариях или даже других играх.

Несмотря на большое количество преимуществ, данный подход обладает недостатком. Настройка ИИ на основе полезности требует от разработчика выражать каждое действие в виде чисел, что кажется не естественным (особенно в начале). Это можно сказать о любой задаче программирования. Люди не думают так же, как компьютер, что в результате заставляет разработчика выражать себя так, чтобы компьютер мог легче понимать его намерения.

1.7 Комбинации разных подходов

Для решения проблем, возникающих при применении одного из подходов, используются комбинации из нескольких. Так, например, авторы статьи [17] предлагают подход, соединяющий методы BT и HTN. Этот метод подразумевает, что высокоуровневое принятие решений выполняется HTN, а низкоуровневое делегируется BT. Такой подход позволяет строить долгосрочные планы, при этом оставаясь достаточно реактивным для изменений в окружающем мире. Метод основан на том, что централизированный планировщик строит планы для нескольких агентов без знания того, как будут выполнены низкоуровневые задачи отдельных агентов.

В статье [18] совмещен подход GOAP и HTN, в котором GOAP отвечает за непредсказуемость действий агентов ИИ, а HTN за долгосрочное планирование.

Однако эти методы не решают поставленные проблемы в полной мере из-за особенностей этих подходов.

1.8 Подходы, основанные на машинном обучении

Существует множество реализаций искусственного интеллекта, использующих идеи машинного обучения. Так, например, в [19] применяют подход обучения с подкреплением (RL или Reinforcement Learning) для создания адаптивного ИИ. Авторы [20] описали применение мульти-агентного обучения с подкреплением (MARL или Multi-Agent Reinforcement Learning). В статье [21] приводится пример использования MARL для тренировки ИИ в игре Starcraft (Blizzard Entertainment. 1998). В [22] авторами применяется генетический

алгоритм (GA или Genetic Algorithm) для динамического изменения сложности игры.

Однако эти подходы требуют больших мощностей для обучения и большого количества итераций для получения приемлемого результата [23]. Из этого можно сделать вывод, что они не пригодны для применения в процессе разработки игр, так как при любом изменении логики игры, агентов придется обучать заново.

2 Разработка теоретической модели

В данной главе описаны системы, необходимые для создания достоверного искусственного интеллекта, такие как восприятие, память, механизм принятия решений и коммуникации. Также представлено описание взаимодействия этих систем. Общая схема взаимодействия всех систем искусственного интеллекта представлена на Рисунке 7.

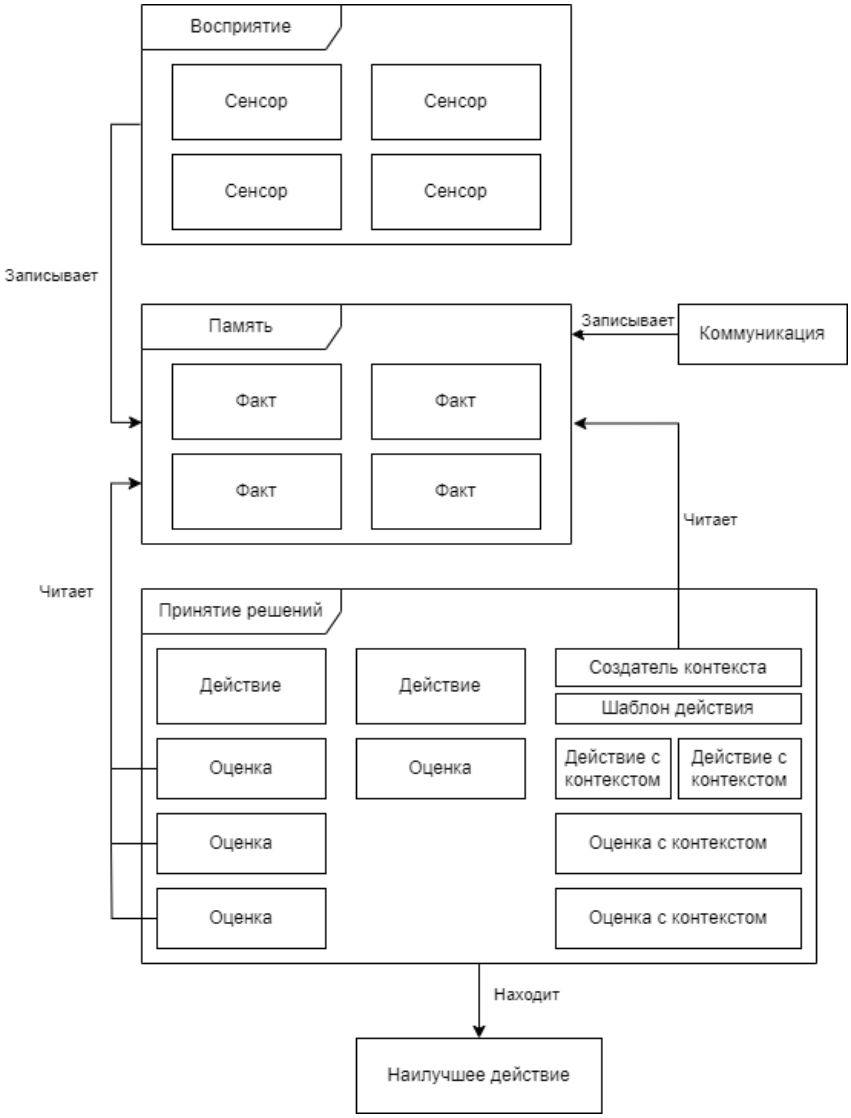

Рисунок 7 – Схема взаимодействия систем ИИ

2.1 Восприятие

Для искусственного интеллекта в играх система восприятия играет критическую роль, так как позволяет агентам воспринимать различные объекты игрового мира, а также других агентов или игроков. Эта система включает в себя несколько сенсоров, которые активно взаимодействуют с игровым пространством [24].

Самым значительным сенсором является зрительный сенсор, который имеет решающее значение для определения окружения. Он позволяет виртуальным агентам использовать технику бросания лучей для обнаружения других агентов в определенном поле обзора. Это позволяет персонажам реагировать на события в игре в соответствии с их движением и местоположением, избегать препятствий и осознавать свое окружение.

Следующим по значимости является сенсор слуха, который может улавливать звуки, издаваемые другими объектами. Он учитывает окружающий шум, производимый игроком или другими агентами, и изменяет осведомленность о событиях и вещах в непосредственной близости. Агент может реагировать и действовать в соответствии с информацией, например, слышать шаги другого агента.

Биологический механизм восприятия страха играет большую роль в принятии решений животными [25]. Например, высокий уровень страха может предостеречь животное приближаться к водоему или месту, где был замечен хищник в прошлый раз. Каждое животное обладает параметром опасности, который характеризует степень угрозы для других животных. Если рядом с животным

находится другое существо с более высоким уровнем опасности, степень страха у первого животного увеличивается в зависимости от разницы в уровнях опасности между ними.

Значение страха уменьшается постепенно в отсутствие угрозы. Однако скорость снижения страха может различаться для каждого индивидуума в зависимости от его биологических особенностей и предыдущего опыта. Этот механизм адаптации помогает животным эффективно реагировать на изменения в окружающей среде и минимизировать риск столкновения с потенциальными опасностями.

Параметры сенсоров могут меняться в зависимости от игровой ситуации, например, если животное пьет, у него уменьшается дальность взгляда и слуха, но увеличивается степень роста страха.

Также немаловажным механизмом является поиск пути. Он позволяет агенту найти оптимальный путь из одной точки в другую. Поиск пути может обеспечить реалистичное и эффективное передвижение агентов в игровом мире.

2.2 Память

В играх память агентов ИИ относится к способности неигровых персонажей (NPC) или сущностей в игре сохранять и вызывать информацию об себе, игровом мире и действиях других агентов или игроков. Моделируемая память имеет важное значение для создания более реалистичного и динамичного взаимодействия между игроком и игровой средой. Информация о мире хранится в памяти в виде ассоциативного массива, где каждый объект имеет свой уникальный идентификатор, выступающий в качестве ключа. Для каждого объекта сохраняются различные стимулы, содержащие в себе факт,

такой как местоположение объекта или его текущий статус. Важно отметить, что каждый из этих стимулов также обладает приоритетом, который определяет степень важности факта в системе.

Информация, поступающая из различных источников, может иметь разные уровни приоритета в зависимости от их достоверности или актуальности [26]. Например, данные от сенсоров зрения могут иметь более высокий приоритет по сравнению с данными от сенсоров слуха или коммуникационных каналов [27].

Система также обладает механизмом забывания информации, который реализуется через постепенное снижение приоритета фактов. Когда приоритет определенного факта ниже определенного порогового значения, факт удаляется из памяти. Этот механизм позволяет системе динамически адаптироваться к изменяющейся среде и поддерживать актуальность своего знания о мире.

2.3 Коммуникация

Система коммуникации позволяет объектам ИИ обмениваться информацией, координировать действия и выражать намерения [28].

Координация и сотрудничество становятся возможными, поскольку агенты ИИ кооперируются для решения таких задач, как групповое движение и тактическое планирование [29].

Агенты ИИ могут общаться, чтобы координировать атаки или корректировать стратегии в зависимости от меняющихся обстоятельств.

В целом эффективные системы связи способствуют реалистичности и сложности поведения ИИ в видеоиграх.

В рамках моделирования коллективного разума животных необходима возможность агентов формировать группы в целях достижения своих целей [30]. Групповая координация действий осуществляется путем передачи важной информации между агентами внутри группы.

Каждый агент имеет возможность либо создать собственную группу, либо вступить в уже существующую. Важно отметить, что группа перестает существовать, когда последний агент покидает её состав.

Цели и поведение группы задаются заранее разработчиком, однако каждый агент принимает решение о вступлении в определенную группу на основе своих собственных потребностей и учитывая предоставленную информацию о целях группы.

Такой подход позволяет моделировать взаимодействие между агентами с учетом их индивидуальных стратегий и целей, а также динамически адаптировать их поведение в зависимости от изменяющейся ситуации в игровой среде.

Таким образом, интеграция систем восприятия, функций памяти и механизмов коммуникации предоставляет все необходимое для дальнейшего принятия решений.

2.4 Принятие решений на основе Utility-based AI

Система принятия решений в данной работе основана на подходе Utility-based AI, так как он позволяет агентам быстро реагировать на изменяющиеся игровые условия. Главная цель принятия решений – выбрать наилучшее действие из всех возможных в текущей ситуации.

Каждый агент ИИ обладает набором поведений, которые обязательно включают в себя:

Действие – при выборе поведения, агент совершает действие, например, идет к определенной цели или взаимодействует с игровым миром;

Набор функций оценки – каждая функция считает ранг и полезность действия, определенного в этом поведении. Каждая из функций оценивает определенный аспект ситуации в изоляции и возвращает значение, которое может быть скомбинировано со значениями других функций оценок для определения финальный оценки действия.

Метод комбинирования этих оценок – определяет каким образом будут скомбинированы все посчитанные значения полезности.

В дополнение к этому, поведение, если оно контекстно-зависимое, может иметь «создателя контекста», который производит необходимые объекты для подсчета полезности и исполнения действия.

Используя такой подход, процесс настройки поведений сводится к указанию набора функций оценок и контролирующих параметров для каждой из этих функций. Из-за того, что существует относительно малое количество функций оценок, логика ИИ является более компактной, чем если бы она была реализована в каждом используемом месте. Благодаря этому, разработчики ИИ могут думать в рамках более крупных, высокоуровневых концептов, что больше подходит под их процесс принятия решений.

Также каждая функция оценки переиспользуется снова и снова. Вместо того, чтобы дублировать логику ИИ в каждом месте, она имплементирована один раз, протестирована и представляет из себя единое надежное решение. Это приводит к более гибкой и поддерживаемой системе.

Сам процесс принятия решений состоит из нескольких этапов:

Вначале все создатели контекста генерируют объекты для контекстно-зависимых действий. Далее генерируются опции, состоящие из действия и, опционально, контекста. Во время генерации проверяется валидность контекста и действия и, в случае провала проверки, эти действия отбрасываются. Также считается полезность и ранг действия. Для подсчета ранга используется следующая формула:

$$R = R_0 + max\left(rank\left(C_j\right)\right) + commit, \quad (1)$$

где R – ранг действия;

R_0 – базовый ранг действия;

$rank(Cj)$ – функция ранга каждого из функций оценок;

commit – функция, которая возвращает 1, если действие выполняется в данный момент и 0 в остальных случаях.

Для подсчета полезности используется следующая формула:

$$U = combine\left(utility\left(C_j\right)\right), \quad (2)$$

где U – полезность действия;

$combine$ – метод комбинирования всех значений полезности, посчитанными функциями оценок;

$utility$ – функция полезности каждой из функций оценок.

Функции оценок содержат в себе несколько методов: проверку на валидность, функцию подсчета полезности и входные данные для этой функции.

Функции могут быть абсолютно разные. Самыми часто используемыми являются линейная (3), степенная (4), синусоидная (5) и косинусоидная (6) [31]. Эти функции представлены на Рисунке 8.

$$\mathrm{f(x)} = \left(\frac{x}{a}\right) - b \tag{3}$$

$$f(x) = x^a + b \tag{4}$$

$$f(x) = \sin(x\pi a) + b \tag{5}$$

$$1 - \cos(x\pi a) + b \tag{6}$$

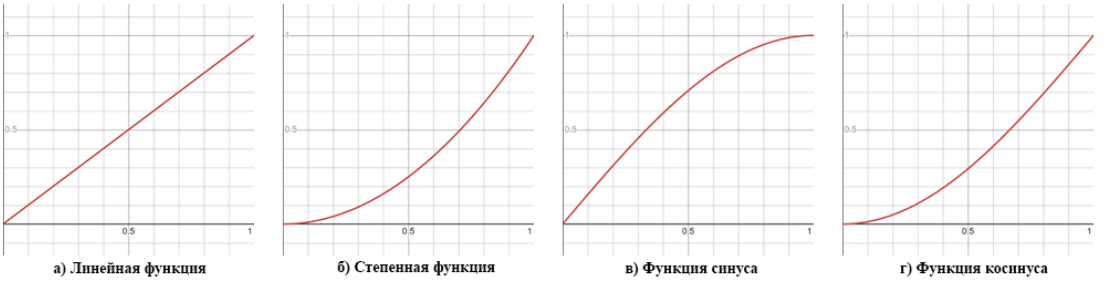

Рисунок 8 — Возможные функции для оценок. (а) Линейная функция. (б) Степенная функция. (в) Функция синуса. (г) Функция косинуса

Менее популярные, но все еще часто используемые: логистическая (7), логит (8), smoothstep (9), smootherstep (10). Они представлены на Рисунке 9.

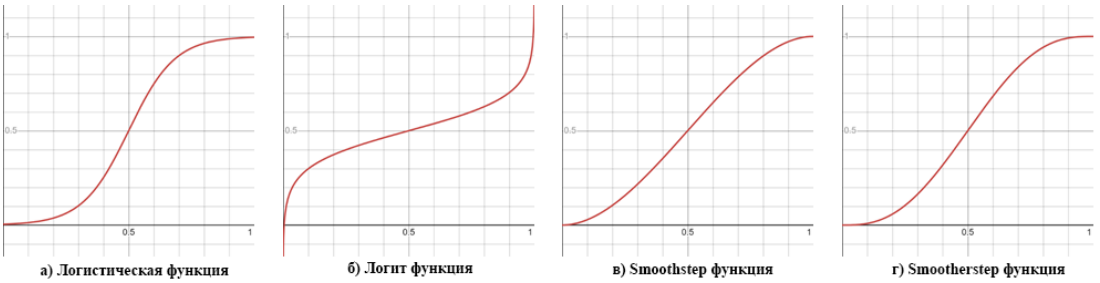

Рисунок 9 — Возможные функции для оценок. (а) Логистическая. (б) Логит. (в) Smoothstep. (г) Smootherstep

$$f(x) = \frac{1}{1 + e^{-k \times (4e \times (x - x_0) - 2e)}} \tag{7}$$

$$f(x) = \frac{log_e\left(\frac{x}{(1-x)}\right) + 2e}{4e} \tag{8}$$

$$f(x) = x^2 \times (3 - 2x) \tag{9}$$

$$f(x) = x^3 \times (x \times (6x - 15) + 10) \tag{10}$$

Далее происходит непосредственно выбор действия. Какой алгоритм использовать, зависит от конкретной цели. Обычно используется один из трех методов:

Абсолютный: выбирается действие с максимальным коэффициентом полезности.

Относительный: коэффициент полезности рассматривается как вес, а действие выбирается путем взвешенной случайности.

Вероятность для выбранной опции определяется делением коэффициента полезности этой опции на суммарную полезность всех опции:

$$P_o = \frac{U_o}{\sum_{i=1}^{n} U_i},$$

где P_o – вероятность опции;

U_o – полезность опции;

n – количество опций.

Двойной выбор: объединение двух методом. Сначала выбираются все действия с максимальным рангом (абсолютная полезность). Затем среди оставшихся ищется максимальный коэффициент полезности или вес (относительная полезность). После этого отсекаются все действия, которые сильно отличаются от максимума. В конце на оставшихся действиях используется относительный алгоритм.

Концептуально, ранг разделяет опции на категории, что позволяет выбирать опции из наилучшей категории. Вес используется для оценивания опций в рамках контекста их категории. Получается, вес опции имеет смысл только относительно других весов в данной категории и только веса опций в наилучшей категории имеют значение.

Еще одной модификацией является *Тег*. Каждому действию можно присвоить *Тег*, который добавляет дополнительный ранг, если сейчас выполняется действие с точно таким же тегом.

Таким образом можно сгруппировать действия, необходимые для определенной группы агентов. Это помогает исключить действия, которые, хотя и могут быть полезными в общем контексте, но не соответствуют текущим целям группы.

Ранг необходим, чтобы действия оценивались не только исходя из их собственной полезности, но и из того, насколько они соответствуют текущему контексту и целям группы.

Само принятие решений происходит раз в определенный период, определяемый разработчиком (по умолчанию 10 раз в секунду).

Также действие может быть прервано, что приведет к выполнению этапа принятия решений вне периода обновления.

2.5 Исследование поведения львов

Для проверки работоспособности алгоритма, было принято решение создать симулятор львиного прайда благодаря тому, что поведение львов является достаточно сложным из-за их социальной структуры. В области разработки игр моделирование реалистичного поведения животных является важным компонентом создания захватывающей виртуальной среды. Симуляция коллективного разума животных требует глубокого знания их поведения в естественной среде. Далее рассматривается специфика поведения львов, их социальная структура и стратегии охоты.

2.5.1 Социальная структура и кооперация львов

Львы — очень социальные животные, известные тем, что образуют прайды, состоящие из родственных самок, их потомков и группы самцов. Социальная структура прайда определяется сотрудничеством и скоординированными усилиями по охоте и защите территории [30]. В игровом контексте воспроизведение этой социальной организации имеет решающее значение для точного представления поведения льва.

При разработке прототипа можно черпать вдохновение из сложного общения и сотрудничества, наблюдаемого в львиных прайдах. Усовершенствованные алгоритмы искусственного

интеллекта могут помочь создать внутриигровых персонажей-львов с совместным поведением, таким как коллективная стратегия охоты и скоординированная защита от внешних угроз.

Львы — территориальные животные, которые коллективно создают и защищают свои территории. Выбор подходящей территории — это стратегическое решение, на которое влияют такие факторы, как наличие добычи, водоснабжение и укрытие [32]. Включение территориальной динамики в прототип может добавить глубины виртуальной экосистеме.

Разработчики могут использовать сигналы окружающей среды и поведенческие алгоритмы для имитации территориального поведения львов в играх. Сюда могут входить динамические изменения виртуального мира в результате действий виртуального львиного прайда, которые будут влиять на общий игровой процесс.

2.5.2 Стратегии охоты

Охотничьи способности — жизненно важная часть поведения львов. Львы — высшие хищники, которые используют различные методы охоты, в том числе скоординированную групповую охоту [33]. Одной из возможных стратегий охоты может быть охота с засадой. Когда один или несколько львов являются загонщиками, а остальные сидят в засаде и ждут, когда жертву загонят ближе к ней. Понимание этой тактики необходимо для разработки увлекательного и реалистичного игрового процесса.

Внутриигровой ИИ львов может быть запрограммирован так, чтобы имитировать динамические процессы принятия решений,

связанные с охотой. Это включает в себя такие сущности, как выбор цели, скрытые подходы и скоординированные атаки.

Черпая вдохновение из сложной социальной структуры львов, территориальной динамики и методов охоты, разработчики могут создавать игры-симуляторы, которые захватывают игроков реалистичной картиной животных в их естественной среде обитания. Коллективное сознание животных можно воплотить в жизнь в цифровой сфере улучшив игровой опыт для игроков по всему миру.

3 Реализация прототипа на основе модели

3.1 Описание прототипа

Прототип создан с помощью игрового движка Unity из-за его большой популярности, а также возможности для построения эффективных систем искусственного интеллекта.

Симуляция содержит в себе некоторое количество агентов-животных разных типов, которые полностью управляются искусственным интеллектом.

Каждый из агентов обладает набором характеристик:

– скорость усиления голода и жажды,

– трата энергии, в том числе во время ходьбы и бега,

– скорость утоления жажды, голода и сна,

– скорость передвижения во время ходьбы и бега.

Также у каждого животного есть 3 важных показателя – голод, жажда и усталость. Если один из показателей будет держаться на низком уровне в течении 10 секунд, то животное погибнет. Животное не может совершать действия (кроме отдыха), если у него максимальная усталость.

В прототипе присутствует 2 типа животных: антилопы и львы. Антилопы бродят по местности и иногда подходят к водоему, чтобы утолить жажду. А львы могут спать в высокой траве, утолять жажду, питаться мясом, добытым с помощью охоты на антилоп.

Кроме этого, проект включает в себя открытое пространство, разбитое на разные регионы, в которых находятся такие объекты как:

– Водоемы, состоящие из нескольких источников, из которых в свою очередь могут пить животные и утолять свою жажду. Со временем источники истощаются и водоем пересыхает;

– Высокая трава, в которой спят львы и восполняют энергию. Лев может спать в любой незанятой траве;

– Источник пищи, с помощью которого львы утоляют голод.

На Рисунке 10 продемонстрирован прототип, где показаны лев (белая фигура), антилопа (красная фигура), источники воды и пищи, а также высокая трава.

Рисунок 10 – Элементы прототипа

Цель симуляции – как можно дольше выживать львам, удовлетворяя потребности в пище, воде и сне. Для добычи пищи львы охотятся на антилоп, если у них нет другой доступной пищи. После смерти антилопы, она превращается в источник пищи.

Поведение львов состоит из набора действий, указанных в Таблице 1. Также у каждого агента есть набор создателей контекста, необходимых для контекстно-зависимых действий.

Таблица 1 – Поведение львов

Название	Контекст	Функции оценки	Группа
1	2	3	4
отдых в любом месте	-	оценка усталости	-
отдых в высокой траве	высокая трава	оценка усталости, оценка расстояния	основное
движение к высокой траве	высокая трава	оценка усталости, оценка расстояния	основное
питье из источника воды	источник воды	оценка жажды, оценка расстояния	основное
движение к источнику воды	источник воды	оценка жажды, оценка расстояния	основное
поедание источника пищи	источник пищи	оценка голода, оценка расстояния	основное
движение к источнику пищи	источник пищи	оценка голода, оценка расстояния	основное
вступление в группу охоты	-	оценка голода, оценка отсутствия источника пищи	основное
движение к антилопе	антилопа	оценка голода, оценка расстояния, оценка отсутствия источника пищи	охота
рывок к антилопе	антилопа	оценка голода, оценка расстояния, оценка отсутствия источника пищи	охота
атака антилопы	антилопа	оценка голода, оценка расстояния, оценка отсутствия источника пищи	охота
ожидание готовности засады	-	оценка готовности засады, оценка расстояния до цели	охота
движение к засаде	-	оценка расстояния до засады	охота

Продолжение таблицы 1

1	2	3	4
ожидание в засаде	-	оценка расстояния до цели	охота
вступление в группу миграции	регионы	оценка истощения водоема в регионе, оценка дальности	основное
движение в новый регион	регионы	оценка нахождения в группе миграции, оценка дальности	миграция

Для оценки расстояния используется степенная функция (4), в которую подается расстояние до цели.

Для оценки усталости используется линейная функция (3), в которую передается процент усталости животного.

Для оценки жажды и голода используется линейная функция (3), входным данными, которых является процент жажды и голода соответственно.

Оценка отсутствия источника пищи – это функция, которая возвращает 0, если лев знает об источнике пищи, иначе 1.

Оценка готовности засады возвращает 1, если хотя бы один лев уже достиг точки засады, иначе возвращает 0.

Оценка нахождения в группе миграции представляет из себя функцию косинуса (6), в которую подается процент львов, уже находящихся в этой группе.

Оценка истощения водоема в регионе – это функция синуса (5), в которую подается процент истощенных источников в этом регионе.

Антилопы бродят по миру стадом, изредка подходят к водоему чтобы утолить жажду и время от времени останавливаются, чтобы

утолить голод травой. Если хоть на одно животное из стада напал лев, у всех остальных животных поблизости повышается уровень страха, и они начинают убегать в безопасное место. Все антилопы принадлежат группе стада, поэтому их намерения в общем случае совпадают.

Поведение антилоп также состоит из набора разных действий, указанных в Таблице 2.

Каждое животные обладает тремя сенсорами: зрения, слуха и страха. Сенсор зрения, изображенный на Рисунке 11, настраивается с помощью четырех параметров:

1) полууглом,
2) дальностью,
3) частотой сканирования,
4) приоритетом.

Сенсор слуха, показанный на Рисунке 12, настраивается следующими параметрами:

– максимальным радиусом,
– коэффициентом затухания,
– приоритетом.

Сенсор страха, представленный на Рисунке 13, контролируется несколькими параметрами:

– радиусом восприятия,
– слоями, которые влияют на страх этого животного,
– максимальным значением страха,
– скоростью падения страха при отсутствии опасности.

Таблица 2 – Поведение антилоп

Название	Контекст	Функции оценки
отдых в любом месте	-	оценка усталости, оценка страха
питье из источника воды	источник воды	оценка жажды, оценка расстояния, оценка страха
движение к источнику воды	источник воды	оценка жажды, оценка расстояния, оценка страха
движение к источнику пищи	источник пищи	оценка голода, оценка расстояния, оценка страха
поедание источника пищи	источник пищи	оценка голода, оценка расстояния, оценка страха
выгул в окрестностях	-	оценка страха
бег от львов	-	оценка страха, оценка расстояния

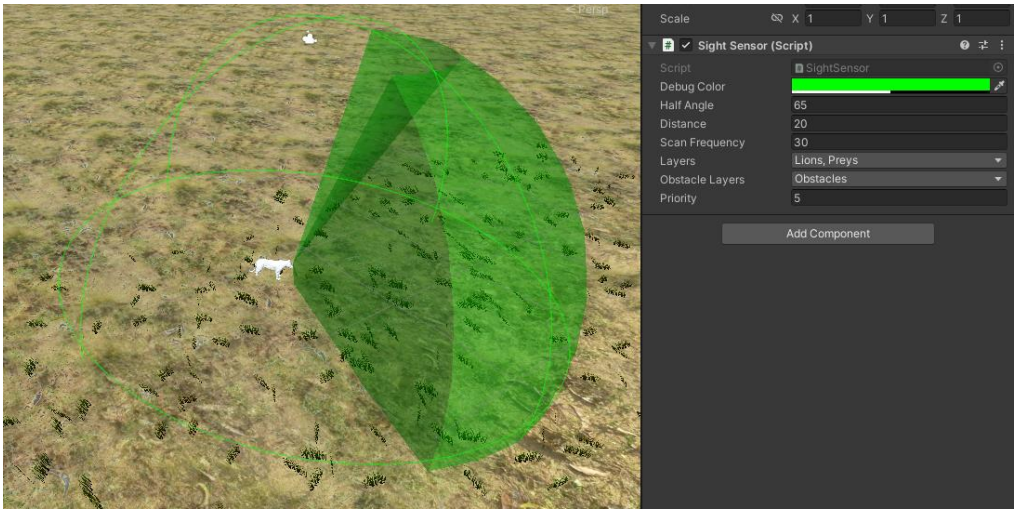

Рисунок 11 – Сенсор зрения в эдиторе

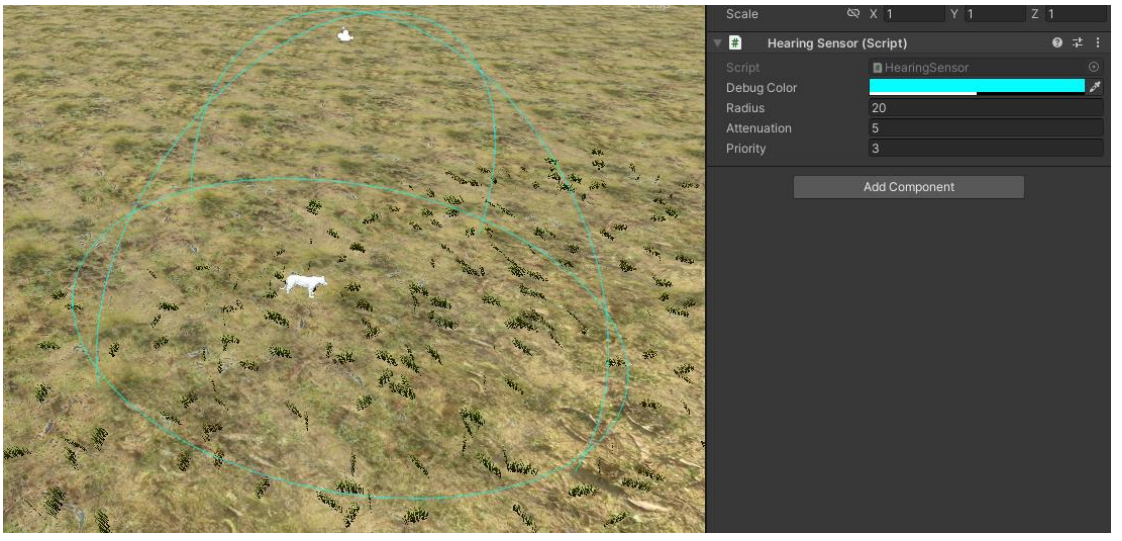

Рисунок 12 – Сенсор слуха в эдиторе

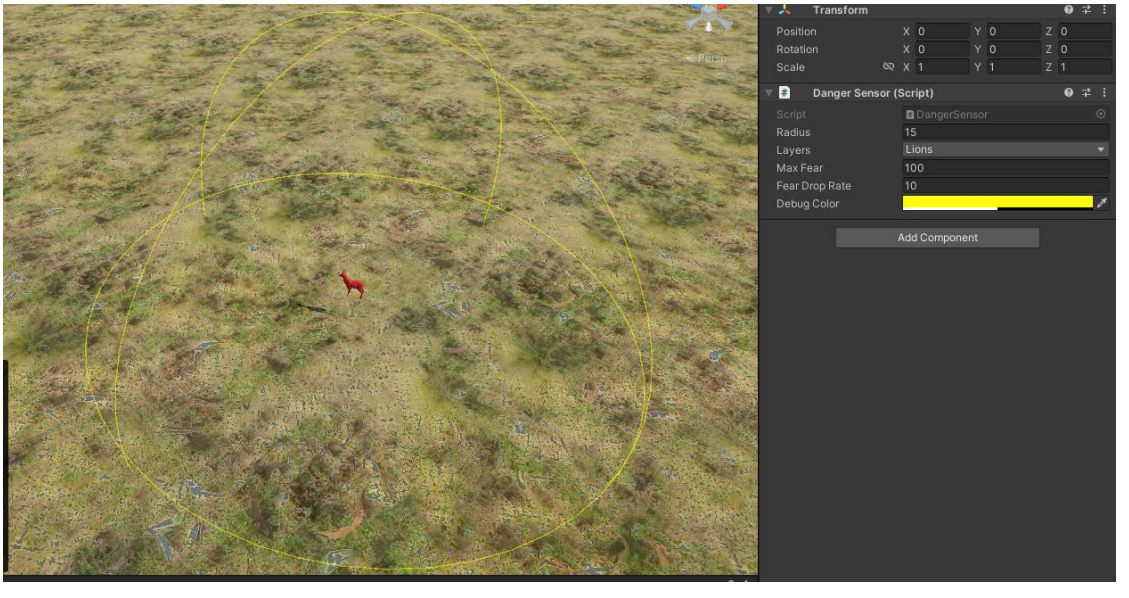

Рисунок 13 – Сенсор страха

Поиск пути реализован с помощью встроенных средств Unity – NavMesh. Как видно на Рисунке 14, вся земля покрыта навигационной сеткой, а область воды имеет штраф передвижения, чтобы животные проходили по воде только в случае полной необходимости. Каждое животные имеет компонент NavMeshAgent, в котором настроены

указываются скорость движения, поворота, ускорения и другие настройки. Дополнительно на каждом агенте присутствует компонент NavMeshModifier, который не дает животным влиять на построенную навигационную сетку.

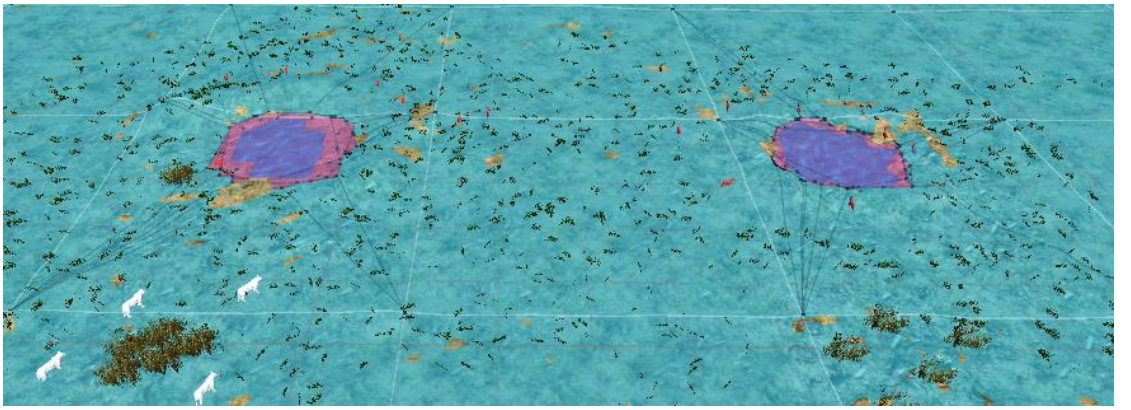

Рисунок 14 – Навмеш на прототипе

3.2 Тестирование алгоритма

В данной работе было проведено серия экспериментов в трех сценариях для проверки работоспособности алгоритма Utility-Based AI для моделирования коллективного разума животных. Эксперимент считается успешным, если все львы остаются в живых на протяжении 15 минут реального времени или 30 дней игрового. Такое число выбрано из-за того, что в среднем взрослый лев может обходиться без еды около 2 недель и большее время испытания не имеет смысла. Для получения более стабильных результатов было произведено 100 испытаний в каждом из сценариев.

В первом сценарии в мире были представлены: 1 лев, 5 антилоп, 1 водопой с 10 источниками воды и 2 штуки высокой травы. На старте симуляции лев имел 50% усталости, 30% голода, 10% жажды. Антилопы всегда начинают с 0% усталости, голода и жажды. Каждое

животное обладает набором своих характеристик, они показаны в Таблице 3.

Таблица 3 – Характеристики животных в сценарии 1

Название	Лев	Антилопы
скорость роста голода	0.2 %/с	0.8 %/с
скорость роста жажды	0.4 %/с	0.6 %/с
скорость роста усталости	0.6 %/с	0.3 %/с
скорость роста усталости при ходьбе	0.8 %/с	0.4 %/с
скорость роста усталости при беге	1.2 %/с	0.5 %/с
скорость восполнения жажды	6 %/с	10 %/с
скорость восполнения голода	3 %/с	5 %/с
скорость восполнения усталости	2 %/с	10 %/с
скорость ходьбы	5 м/с	7 м/с
скорость бега	20 м/с	22 м/с

В результате все испытания были провалены. Во всех испытаниях, лев успешно утолял жажду и восстанавливал энергию, однако погибал от голода примерно через 6 минут. Это можно объяснить его стратегией охоты, которая заключалась в банальном преследовании жертвы и оказалась неэффективной ввиду того, что у антилоп быстрая реакция и большая скорость бега, что позволяет им убегать от хищников.

Во втором сценарии в симуляции участвуют: 2 льва, 2 группы по 5 антилоп, 2 водопоя с 10 источниками воды каждый, 5 штук высокой травы. Второй лев в начале испытаний имел 20% усталости, 10% голода, 70% жажды. Характеристики животных указаны в Таблице 15.

В результате второго сценария опять же все испытания были провалены, но среднее время увеличилось до 11 минут. Это стало

возможным благодаря тому, что два льва кооперировались в охоте на антилоп.

Таблица 4 – Характеристики животных в сценарии 2

Название	Лев 1	Лев 2	Антилопы
скорость роста голода	0.2 %/с	0.15 %/с	0.8 %/с
скорость роста жажды	0.4 %/с	0.45 %/с	0.6 %/с
скорость роста усталости	0.6 %/с	0.55 %/с	0.3 %/с
скорость роста усталости при ходьбе	0.8 %/с	0.85 %/с	0.4 %/с
скорость роста усталости при беге	1.2 %/с	1.4 %/с	0.5 %/с
скорость восполнения жажды	6 %/с	8 %/с	10 %/с
скорость восполнения голода	3 %/с	1 %/с	5 %/с
скорость восполнения усталости	2 %/с	3 %/с	10 %/с
скорость ходьбы	5 м/с	6 м/с	7 м/с
скорость бега	20 м/с	19 м/с	22 м/с

Когда у одного из львов появляется необходимость в пище, он создает группу охоты и становится лидером и одновременно приобретает роль загонщика. Второй лев тоже вступает в группу, если у него нет более приоритетных дел, и становится исполнителем засады.

После создания группы, лидер группы выбирает цель и постоянно сообщает ее местоположение всем львам в группе. В это время загонщик подходит на безопасное расстояние к цели, чтобы не спугнуть ее и ждет пока второй лев будет в засаде. Засадой могут считаться ближайшие к жертве высокие кусты.

После того, как второй лев добирается до места засады, загонщик начинает двигаться на жертву, таким образом принуждая ее бежать в сторону засады. Как только жертва приблизилась достаточно близко, лев в засаде тоже начинает двигаться в сторону жертвы. Описанная выше тактика засады изображена на Рисунке 15.

Результатом такой кооперации становится достаточное количество пищи, позволяющая прожить львам нужное для прохождения испытания время.

Рисунок 15 – Стратегия засады

В третьем сценарии в мире присутствуют два региона, изображенные на Рисунке 16. В первом находятся 5 львов, 10 антилоп, 10 источников воды и 10 штук высокой травы. Также источники воды наполовину исчерпаны. Во втором регионе отсутствуют львы, а количество остальных сущностей не отличается. В первые минуты симуляции, в первом регионе заканчиваются все источники воды. В результате чего антилопы перебираются в другой регион. Львы тоже образуют группу миграции и вместе переходят в новый регион.

В этом сценарии в охоте участвуют все пять львов. Они пользуются такой же стратегией засады, только уже 3 льва являются загонщиками, а два других исполнителями засады. Из-за большего числа львов, более 80% испытаний оказываются успешно пройденными.

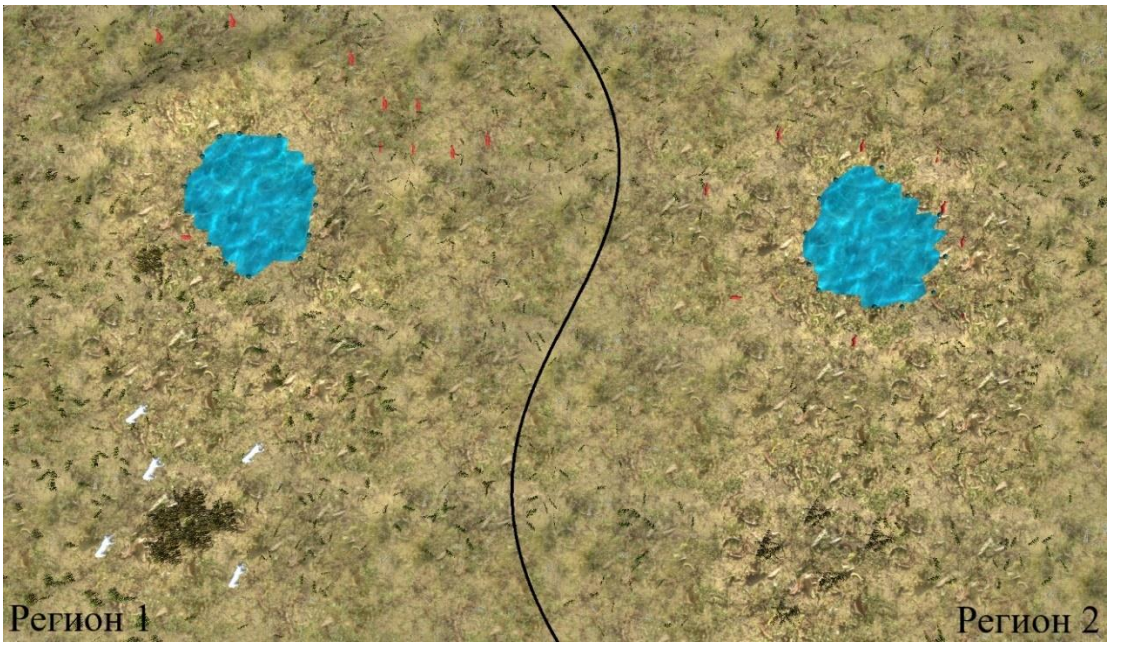

Рисунок 16 – Карта третьего сценария, слева – регион со львами, справа – без львов

На Рисунке 17 изображено время окончания испытания для каждого сценария. Как видно из результатов, алгоритм позволяет львам кооперироваться для достижения общих целей и общего выживания. Для более реалистичной симуляции и глубокого тестирования необходимо внедрение большего количества различных ситуаций. Например, опасных для льва животных, таких как буйволы или жирафы, что в таком случае потребует больше львов и уникальных стратегий для успешных охот.

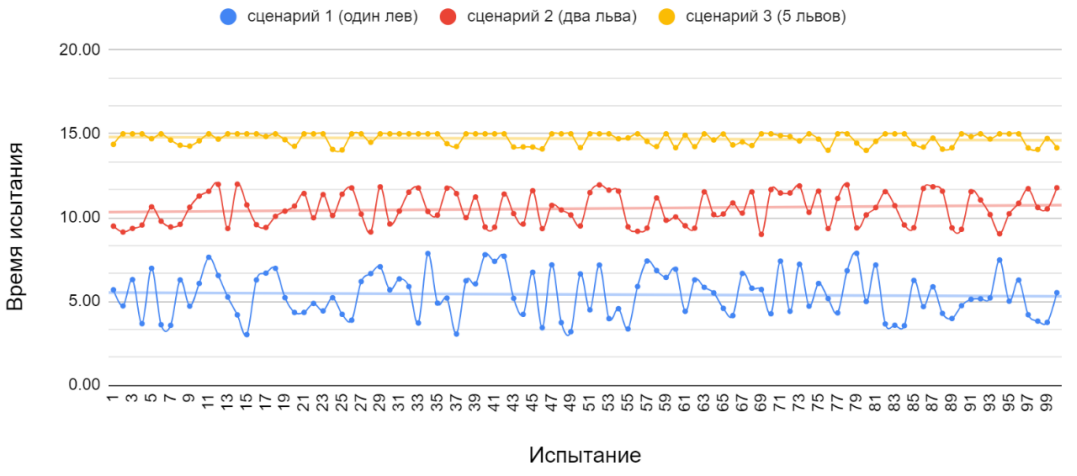

Рисунок 17 – Время окончания испытаний для каждого сценария

3.3 Тестирование производительности

Для замера производительности симуляция производилась с разным количеством львов и антилоп. Количество источников воды и высокой травы всегда равнялось 20 и 10 соответственно. В качестве инструмента замера использовался встроенный в Unity профилировщик. Тестирование проводилось на процессоре AMD Ryzen 5 3500 и видеокарте Radeon RX 590. Результаты тестирования показаны в Таблице 5.

По результатам тестирования производительности видно, что при увеличении количества агентов, время выполнения растет линейно. Учитывая, что для одного агента необходимо примерно 0.5 миллисекунд, можно сделать вывод, что алгоритм является достаточно эффективным.

Таблица 5 – Время работы алгоритма с разным количеством агентов

	1 лев	2 льва	5 львов	10 львов
5 антилоп	~ 0.286 мс	~ 0.658 мс	~ 0.894 мс	~ 1.949 мс
15 антилоп	~ 0.598 мс	~ 0.836 мс	~ 1.144 мс	~ 2.477 мс
30 антилоп	~ 1.292 мс	~ 1.399 мс	~ 1.780 мс	~ 3.987 мс
50 антилоп	~ 2.206 мс	~ 2.441 мс	~ 2.893 мс	~ 4.121 мс

ЗАКЛЮЧЕНИЕ

Коллективный разум животных является важной составляющей игр. Хороший искусственный интеллект может сильно повлиять на погружение игрока, предоставить ему достойный вызов и дать более качественный игровой опыт. Поскольку создатели игр стремятся к все более захватывающей и реалистичной виртуальной среде, понимание коллективного разума животных становится все более полезным.

В выпускной квалификационной работе было проведено и описано исследование на тему «Моделирование коллективного разума животных». В процессе исследования сделан вывод:

Во-первых: были раскрыты все главы данной работы, это анализ существующих методов реализации ИИ, построение модели коллективного разума, тестирование построенной модели.

Во-вторых: были полностью реализованы задачи:

1) провести анализ существующих подходов создания ИИ;
2) разработать теоретическую модель;
3) создать прототип разработанной модели;
4) провести тестирование модели и проанализировать результаты.

Исходя из решения поставленных задач и достижения цели, можно прийти к выводам, что:

Из рассмотренных методов реализации ИИ, таких как FSM, HFSM, BT, GOAP, Utility-based AI, HTN, RL, GA, больше всего подходит для реализации коллективного разума животных подход Utility-based AI.

Для создания искусственного интеллекта, обладающего свойствами адаптивности и коллективности, животных необходимы такие системы как восприятие, память, коммуникация и принятие решений. Также немаловажно их взаимодействие друг с другом.

В рамках проведенных экспериментов разработанная модель показала эффективность в решении задачи коллективного разума животных.

Будущие работы предполагают несколько доработок.

Для улучшения алгоритма возможно применение нейронных сетей вместо обычных функций для оценки действий. Такой подход может помочь достичь лучшего результата работы алгоритма, а также облегчить его настройку, благодаря возможности его обучения.

Другой возможностью улучшить алгоритм может быть создание подгрупп действия для разных ситуаций, например разные группы для открытых и замкнутых пространств.

Для более реалистичной ситуации необходимо добавить больше видов животных, на которых может охотиться львиный прайд. Также необходимо добавить львов мужского пола и маленьких львят – у них должно быть уникальное поведение. Вместе с этим можно добавить естественных врагов, таких как леопард, которые могут охотиться на львят, если они остались без защиты взрослых львов.

Также для более качественной симуляции можно ввести смену дня и ночи. Так как львы отлично видят в темноте, они могут приспособиться охотиться ночью на больших животных, у которых

ночное зрение ограничено, что позволит получать запасы пищи, которых может хватить на долгое время.

Не маловажным является улучшение производительности, чтобы алгоритм эффективно работал с большим количеством агентов даже на слабых устройствах.

Результаты ВКР и исходный текст программы опубликованы в репозитории на GitHub. Получить доступ к нему можно по ссылке [34].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Lu Y., Li W. Techniques and Paradigms in Modern Game AI Systems // Algorithms, Vol. 15, No. 8, 2022. P. 282.
2. Risi S., Preuss M. From Chess and Atari to StarCraft and Beyond: How Game AI is Driving the World of AI // KI - Künstliche Intelligenz, Vol. 34, No. 1, 2020. pp. 7-17.
3. Jagdale D. Finite State Machine in Game Development // International Journal of Advanced Research in Science, Communication and Technology, 2021. pp. 384-390.
4. Fauzi R., Hariadi M., Lubis M., Nugroho S.M.S. Defense behavior of real time strategy games: comparison between HFSM and FSM // Indonesian Journal of Electrical Engineering and Computer Science, Vol. 3, No. 2, 2019. P. 634.
5. Zijie W., Tongyu W., Hang G. A Survey: Development and Application of Behavior Trees // Lecture Notes in Electrical Engineering, 2021. pp. 1581–1589.
6. Colledanchise M., Parasuraman R., Ogren P. Learning of Behavior Trees for Autonomous Agents // IEEE Transactions on Games, Vol. 11, No. 2, 2019. pp. 183-189.
7. Shehabi A.A. Decision-making AI in digital games // Dissertation, 2022.
8. Conway C., Higley P., Jacopin E. Goal-Oriented Action Planning: Ten Years of AI Programming. GDC,
9. Boeda G. Multi-Agent Cooperation in Games with Goal Oriented

Action Planner: Use Case in WONDER Prototype Project // Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Vol. 17, No. 1, 2021. pp. 204-207.

10. Menif A., Jacopin É., Cazenave T. SHPE: HTN Planning for Video Games // Communications in Computer and Information Science. Springer International Publishing, 2014. pp. 119–132.
11. Kelly J.P., Botea A., Koenig S. Planning with Hierarchical Task Networks in Video Games, 2007.
12. Soemers D.J.N., Winands M.H.M. Hierarchical Task Network Plan Reuse for video games // IEEE Conference on Computational Intelligence and Games (CIG), Сентябрь 2016.
13. Świechowski M., Godlewski K., Sawicki B., Mańdziuk J. Monte Carlo Tree Search: a review of recent modifications and applications // Artificial Intelligence Review. Springer Science and Business Media LLC, 2022.
14. Ishihara M., Ito S., Ishii R., Harada T., Thawonmas R. Monte-Carlo Tree Search for Implementation of Dynamic Difficulty Adjustment Fighting Game AIs Having Believable Behaviors // IEEE Conference on Computational Intelligence and Games (CIG), 2018.
15. Lebedeva E., Brown J.A. Companion AI for Starbound Game Using Utility Theory // International Conference Nonlinearity, Information and Robotics, 2020.
16. Swiechowski M., Slezak D. Grail: A Framework for Adaptive and Believable AI in Video Games // IEEE/WIC/ACM International

Conference on Web Intelligence (WI), 2018.

17. Neufeld X., Mostaghim S., Brand S. A Hybrid Approach to Planning and Execution in Dynamic Environments Through Hierarchical Task Networks and Behavior Trees // Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Vol. 14, No. 1, 2018. pp. 201-207.
18. Addoum M.A., Rouffet M., Jacopin E. 3D Brawler Game Using a Hybrid Planning Approach // IEEE Conference on Games (CoG), 2021.
19. Jaderberg M., et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning // Science, Vol. 364, No. 6443, 2019. pp. 859–865.
20. Wang Y., Yang J. Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective // arXiv, 2020.
21. Vinyals O., et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning // Nature, Vol. 575, No. 7782, 2019. pp. 250-354.
22. Paduraru C., Paduraru M. Automatic Difficulty Management and Testing in Games using a Framework Based on Behavior Trees and Genetic Algorithms // 24th International Conference on Engineering of Complex Computer Systems (ICECCS), 2019.
23. Бравичев К.А. Искусственный интеллект на основе нейронных сетей в геймдеве // DevGAMM URL: https://www.youtube.com/watch?v=-sA63nBPYmY (дата обращения: 24.12.22).

24. Баринова Е.С., Арисова Д.А., Коняева О.С. Применение искусственного интеллекта в играх // Теория и практика современной науки, 2018.
25. Саульская Н.Б. Генерализация страха в моделях на животных: нейрофизиологические механизмы и возможные мишени коррекции // Успехи физиологических наук, 2018.
26. Eichenbaum H. Memory: Organization and Control // Annual review of psychology, Vol. 68, 2017.
27. Nieder A. In search for consciousness in animals: Using working memory and voluntary attention as behavioral indicators // Neuroscience & Biobehavioral Reviews, Vol. 142, 2022.
28. Petrović V.M. Artificial Intelligence and Virtual Worlds – Toward Human-Level AI Agents // IEEE Access, T. 6, 2018. C. 39976-39988.
29. Packer C., Scheel D., Pusey A.E. Why lions form groups: food is not enough // The American Naturalist, Vol. 136, No. 1, 1990.
30. Ward A., Webster M. Sociality: The Behaviour of Group-Living Animals, 2016.
31. Lewis M. Winding Road Ahead: Designing Utility AI with Curvature [Электронный ресурс] // GDC URL: https://www.youtube.com/watch?v=TCf1GdRrerw&ab_channel=GDC (дата обращения:10.01.24).
32. Stephanie B. Savannah: Designing a Location Based Game Simulating Lion Behaviour, 2004.
33. Scheel D., Packer C. Group hunting behaviour of lions: a search for

cooperation // Animal Behaviour, Vol. 41, No. 4, 1991. pp. 697-709.
34. https://github.com/mrLegenius/animals-collective-mind