СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Актор – объект, который может находиться на уровне в игре [1].

Блюпринты (Blueprint) – программный класс, реализованный в правилах языка визуального программирования Blueprints Unreal Engine 5.

Бот – система исскусственного интеллекта в игре.

Варгейм – видеоигра в жанре стратегия, которая особенно подчёркивает стратегические и/или тактические боевые действия, а также их историческую точность и реалистичность.

Игровой движок – специализированный комплекс программных средств, предназначенный для разработки компьютерных игр.

Компьютерная игра (в тексте также просто "игра") – компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

Нода – небольшая панель, содержащая данные или программные операции.

Юнит – боевая единица в компьютерной игре.

ВВЕДЕНИЕ

Жанр стратегий является перспективным для разработки новых игр и их монетизации, так как имеет свою аудиторию. Однако из-за небольшой доли рынка и большой рыночной власти потребителей, выпускаемый продукт может столкнуться с заметной конкуренцией. Поэтому разработчикам следует применять все доступные способы повышения конкурентоспособности продукта.

Повышение уровня сложности в большинстве компьютерных игр достигается не за счёт качественного улучшения поведения юнитов, а за счёт примитивного изменения их числовых параметров. Например, за счёт увеличения наносимого урона у противников или его уменьшения у игрока. Игроки воспринимают такой способ повышения уровня сложности как «искусственный» и не всегда честный [2]. Противник при этом не начинает играть лучше и способен периодически совершать нерациональные действия, приводящие к быстрым потерям своих войск. Излишняя лёгкость портит игрокам впечатления от игры, тем самым снижая качество (и успешность) всего продукта. Одним из факторов эффективности в бою является используемое построение, но юниты действуют в рассыпном строю, так как даже не знают о существовании друг друга и не могут полноценно взаимодействовать между собой [3]. Искусственный интеллект расставляет войска определённым образом только в исторических стратегиях (таких как серия Total War). Причём даже там он не способен обучаться и подстраивать строй под изменяющуюся обстановку. Особенно важна реалистичность поведения юнитов в «варгеймах» – симуляторах, ставящей своей целью максимально точное воссоздание реального боя. Улучшить поведение юнитов с стратегиях можно научив их действовать в строю, а также дав искусственному интеллекту возможность каким-либо образом анализировать поведение игрока в прошедших боях. Особенно актуально это становится на фоне того, что в данный момент в разработке находятся сразу несколько

варгеймов от различных компаний, в том числе и отечественных (такие как «WARNO» от «Eugen Systems», «Regiments» от «Bird's Eye Games», «Broken Arrow» от «Steel Balalaika», «Передний край» от компании «Играющие кошки» и другие). Использование новых систем и механик может помочь выделить продукт на фоне конкурентов. Для решения этой задачи была предложена модель генетического алгоритма, оптимизирующих боевой строй юнитов в стратегиях в зависимости от действий игрока на различных ландшафтах. Тип местности было решено учитывать, так как в играх он сильно влияет на характеристики юнитов, накладывая на них бонусы и штрафы.

Цель работы разработать генетический алгоритм для компьютерных игр, оптимизирующий боевой порядок юнитов.

Задачи работы:

1. Изучить работу генетических алгоритмов и возможность их применения в компьютерных играх.
2. Разработать модель работы генетического алгоритма для компьютерной игры в жанре стратегия.
3. Разработать прототип компьютерной игры в жанре стратегия с интегрированным в него генетическим алгоритмом.
4. Апробировать созданный алгоритм, собрать статистику его работы.

Результатом данной работы будет разработанная модель генетического алгоритма, оптимизирующая боевой порядок юнитов в стратегиях.

Ранее генетические алгоритмы для подобным образом не применялись, чем обусловлена актуальность данной работы.

1. ОБЗОР ОБЛАСТИ

1.1 Рынок игровой индустрии

Игровая индустрия оказалась одной из немногих отраслей, которая не просто не оказалась в стагнации от наступления COVID–19, а ускорила свой рост. На рисунке 1.1 видно, что объём рынка видеоигр продолжил увеличиваться и с наступлением пандемии в 2019 году, что вызвало рост инвестиций в него. На рисунке 1.2 представлен объём российского рынка видеоигр.

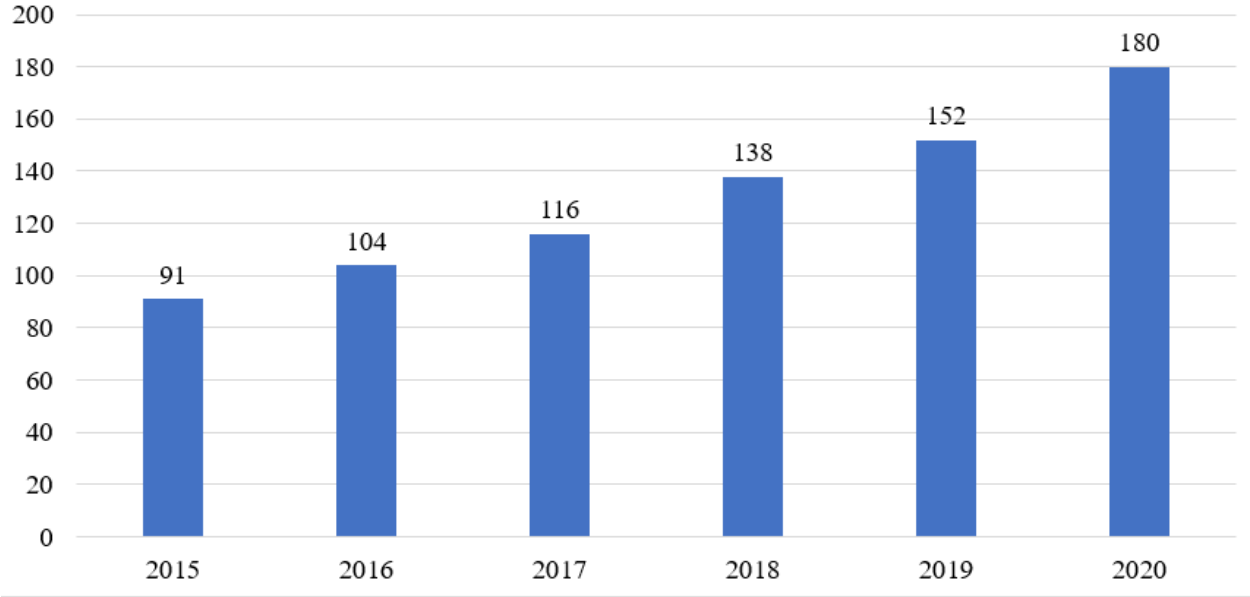


Рисунок 1.1 – Динамика мирового рынка компьютерных игр, млрд. долл. [4]

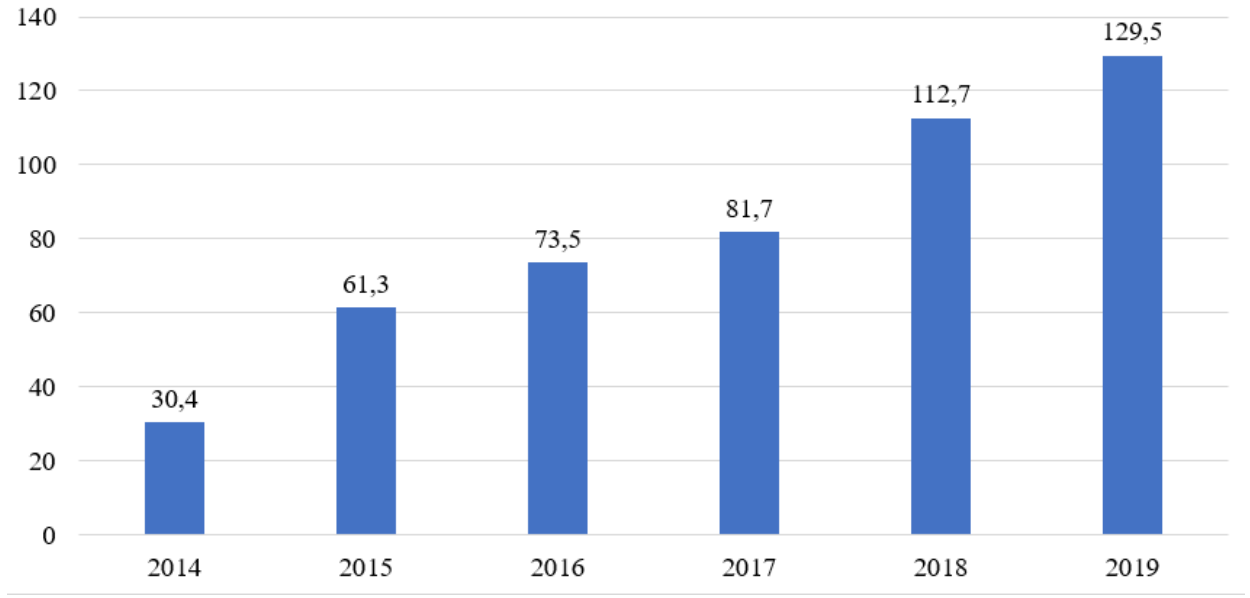


Рисунок 1.2 – Динамика российского рынка компьютерных игр, млрд. руб. [4]

Одним из жанров видеоигр являются стратегии. Хоть они и не преобладают на рынке, однако занимает заметную его долю и являются перспективными для разработки новых игр. Так в 2018 году доля стратегий по продажам на рынке относительно других жанров составляла 3,7 процентов. Доля игр по жанрам представлена на рисунке 1.3.

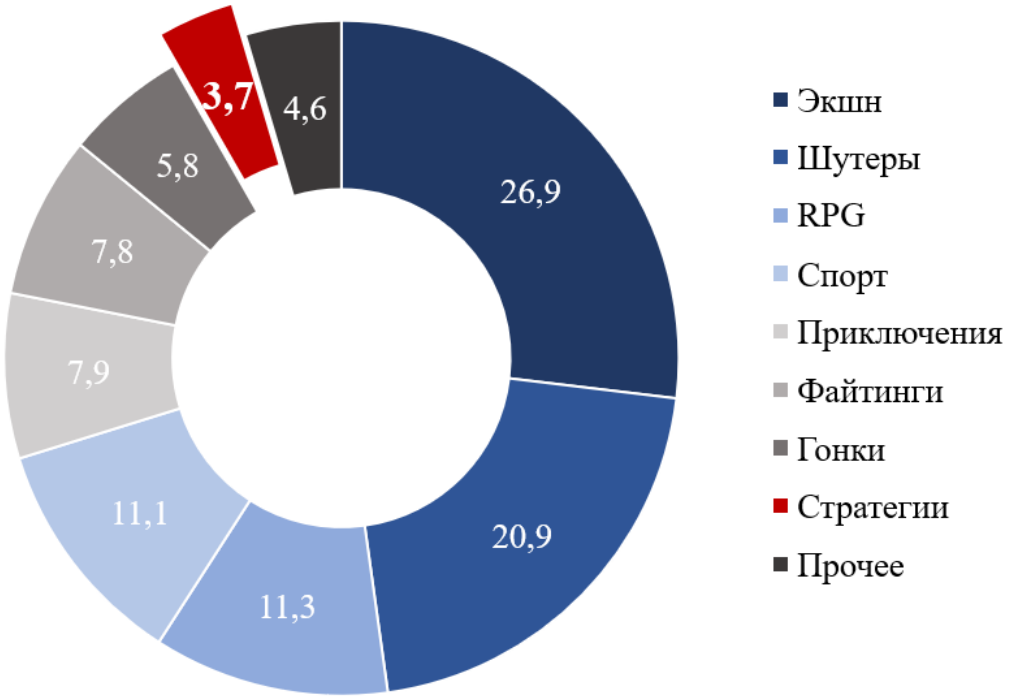


Рисунок 1.3 – Доля рынка различных жанров видеоигр по продажам в 2018 [4]

В 2020 году, согласно исследованию Wargaming и Superdata, доля стратегий по доходу на самой прибыльной платформе – мобильных устройствах – в России составляла 16% [5] [6]. В 2021 году в топ 10 игр по прибыльности на мобильных устройствах также входили стратегии [7].

В начале десятых годов выходило мало крупных стратегий, и жанр находился в упадке. Однако сейчас интерес к стратегиям у аудитории вновь усилился и число игр в жанре, выпускаемых ежегодно, возросло.

Большинство популярных стратегий создаются несколькими студиями: The Creative Assembly, Paradox Interactive, Firaxis Games. Примечательно то, что выход игр от них сопровождается значительным увеличением интереса ко всем играм жанра в целом [8]. Так на рисунке 1.4 показана выручка 4X

стратегий по годам, видно, что пики совпадают с выходом игр из серии Civilization. Со стратегиями крупных студий сложно конкурировать, однако они выходят относительно редко, поэтому малобюджетные проекты независимых разработчиков могут стать успешными. Из-за особенностей отрасли – возможно получение хорошей прибыли с минимальными вложениями – игровая индустрия является перспективным местом для стартапов и создания собственных студий [9]. Существует много примеров, когда малобюджетная игра (даже созданная в одиночку) становилась хитом.

Рисунок 1.4 – Выручка 4X стратегий по годам

В 2022 году Россия входила в топ 5 по числу загрузок игр пользователями и потреблению контента [10] [11], однако доля отечественного рынка от мирового по прибыльности составляет всего 2 процента. По этой причине игры (в том числе и стратегии), ориентированные на российских игроков, являются исключением и выходят довольно редко. Например, таковой является игра “Сирия. Русская буря”. Однако российский рынок и рынок Восточной Европы всё равно являются привлекательными для разработчиков, в том числе и стратегий. Судить об этом можно, например, по

тому, что русский язык является одним из самых популярных для локализации игр, что видно на рисунке 1.5 [12].

Рисунок 1.5 – Количество стратегий, вышедших за первые 5 месяцев 2022 года, имеющих различные локализации [12]

В России есть перспективные для игровой индустрии кадры. На рисунке 1.6 видно, что в 2021 году отечественные разработчики находились на втором месте по числу выпущенных стратегий. Хотя это преимущественно небольшие инди–студии. Студий, делающих AAA–проекты (так принято называть крупнобюджетные блокбастеры), почти нет. Россия в настоящий момент подвержена “утечке мозгов” в игровой индустрии, так как многие специалисты стремятся устроиться на работу в иностранные компании.

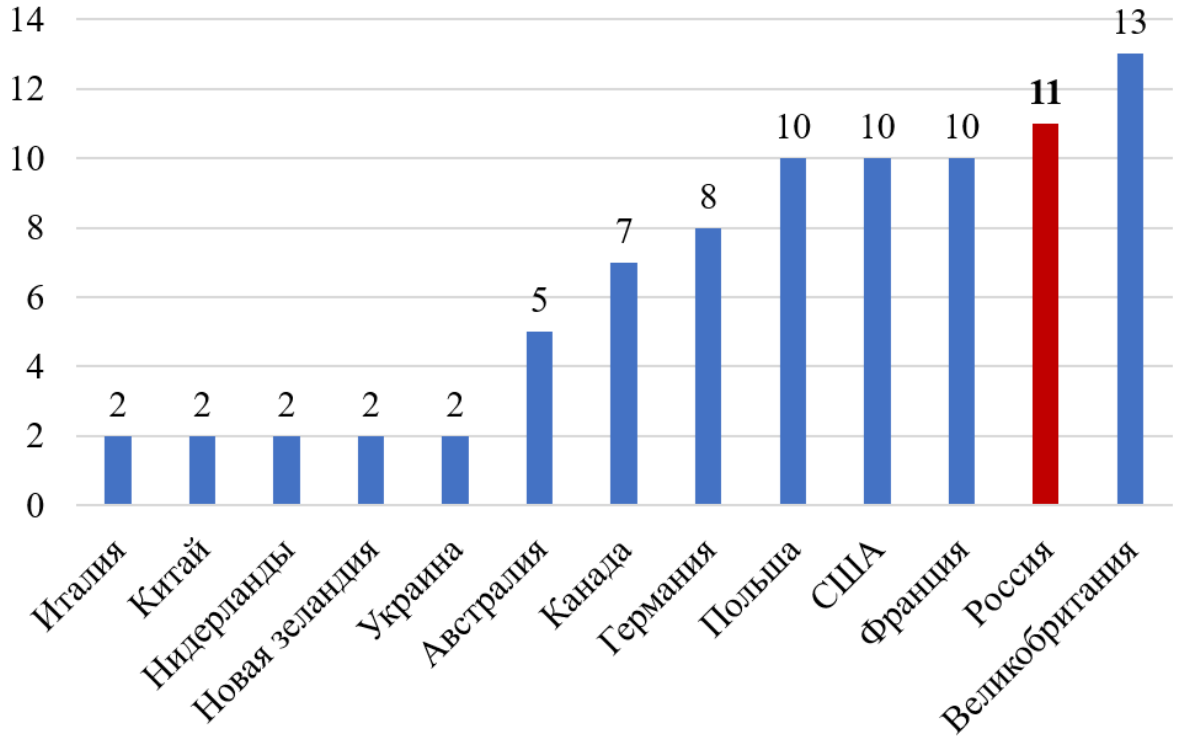


Рисунок 1.6 – Число выпущенных стратегий по странам в 2021 году [12]

Целевую аудиторию стратегий можно разделить на следующие сегменты:

1. «Онлайн игроки» – игроки, предпочитающие крупные MMO– и MOBA–игры. Преимущественно мужчины до 40 лет. Выделяются большим общением в онлайне. Пример игры, интересной для сегмента – Dota 2.
2. «Олды» – условное название для сегмента взрослой аудитории, предпочитающих старые стратегии, выходившие до 10–х годов. Преимущественно мужчины от 30 лет, имеющие семью, как правило имеют слабые компьютеры, так как во флагманских системах не нуждаются. Они играли в стратегии в детстве и сейчас отдают предпочтение тем же играм. Пример игры, интересной для сегмента – Heroes of Might and Magic III.
3. Хардкорные игроки – сегмент геймеров, выделяющийся вниманием к современным стратегиям со сложной системой управления и симуляторам, имеющих высокий порог вхождения.

Преимущественно мужчины до 35 лет. Пример игры, интересной для сегмента – Victoria 3.

4. Казуальные игроки – игроки, предпочитающие мобильные и браузерные гипер–казуальные, казуальные, реже мидкорные стратегии. Большая их доля – женщины. Сегмент включает в себя различившие возрастные категории. Пример игры, интересной для сегмента – Clash of Clans.
5. «Пираты» – сегмент, характерный для российского рынка. Игроки, которые являются только потребителями, но не покупателями. Состоит из детей и взрослых людей с низким доходом, которые не могут ни покупать игры, ни покупать внутриигровые предметы [13].

Сегменты могут пересекаться друг с другом, то есть один и тот же человек может относиться сразу к разным целевым группам.

1.2 Особенности и классификация стратегий

Рассмотрим основные особенности этого жанра.

Стратегия – жанр видеоигр, характеризующийся тем, что игрок управляет одновременно несколькими группами юнитов или крупными системами (экономическими, политическими, социальными или другими, а чаще их комбинацией), путём отдачи приказов, которые эти юниты уже самостоятельно исполняют. Типичной чертой стратегий является расположение камеры сверху, взгляд на происходящее с высоты [14].

Классификация стратегий [15]:

1. по времени:
   - 1.1. стратегии в реальном времени (также часто употребляются английские названия без переводов: Real Time Strategy или сокращённо RTS);
   - 1.2. пошаговые стратегии (англ. Turn Based Strategy или TBS);
2. на основе геймплея:
   - 2.1. Классические стратегии (или просто стратегии) – поджанр стратегий, в котором присутствует экономический элемент и такие

механики как строительство базы, добыча ресурсов, производство юнитов, а геймплей сконцентрирован на военных действиях. Также включает категорию стратегий, которые невозможно отнести к более специфичным поджанрам. Пример игры этого поджанра – Heroes of Might and Magic.

2.2. Тактические стратегии – стратегии, в которых геймплей сфокусирован на тактических военных действиях и управлении группами юнитов. Как правило содержат в себе экономическую составляющую. Отличаются от классических стратегий тем, что вовсе не обязательно содержат такие механики, как строительство баз, добыча ресурсов и тому подобное. Пример игры – Сирия. Русская буря.

2.3. 4X стратегии (от англ. eXplore, eXpand, eXploit, eXterminate – изучай, расширяй, эксплуатируй, уничтожай) – стратегии, в которых кор-геймплей строится вокруг четырёх механик: изучение мира, расширение базы/сферы контроля, использовании имеющихся ресурсов для развития и военных действий. Примером может быть серия игр Civilization.

2.4. Глобальные стратегии – игры, в которых игрок получает контроль над такими большими системами, как государства или цивилизации, на протяжении длительного периода истории. Классическими примерами глобальных стратегий являются игры от студии Paradox Interactive: серии Europa Universalis, Hearts of Iron, Victoria, Stellaris.

2.5. Варгеймы – поджанр, делающий особый упор на военных действиях, их историчности и реалистичности. Пример игры – Steel Division.

2.6. MMO–стратегии (MMO – аббревиатура “массовая многопользовательская онлайн игра”) – стратегии с онлайн–взаимодействием большого числа игроков, как правило в одном игровом мире. Пример – World of Warcraft.

2.7. MOBA (с англ. Multiplayer Online Battle Arena – многопользовательская онлайн боевая арена) – поджанр онлайн стратегий, в котором игроки управляют лишь одним персонажем и в составе команды сражаются против других игроков. Пример известной игры – Dota 2.

2.8. Tower Defense (с англ. оборона башни) – поджанр стратегий, кор-геймплей которых строится вокруг обороны от наступающих волн врагов с помощью строительства различных защитных приспособлений. Пример игры – Plants vs. Zombies.

2.9. Симуляторы строительства и управления – поджанр стратегий, в которых игроку предстоит управлять различными системами без ведения военных действий, чем он принципиально отличается от других. Чаще всего это симуляторы градостроительства, экономические, политические или транспортные стратегии, симуляторы ферм. Примеры игр – SimCity.

Отдельно отметим, что игра может одновременно принадлежать к разным поджанрам. Так, например, игры серии Civilization является одновременно и 4X, и глобальными стратегиями [16]. На рисунке 1.7 представлен скриншот классической RTS Command & Conquer 3: Tiberium Wars, на котором видны юниты, база с постройками и интерфейс.

Рисунок 1.7 – Скриншот Command & Conquer 3: Tiberium

1.3 Проблемы искусственного интеллекта в стратегиях и способы его улучшения

Искусственный интеллект в играх часто не способен учитывать особенности локации. Как было сказано во введении не умеют взаимодействовать между собой полноценно и поддерживать друг друга сознательно. Периодически это приводит к тому, что искусственный интеллект нерационально распоряжается войсками. Например, бот в серии Wargame бот может без разведки наступать танками по лесистой местности, где они из засады быстро уничтожаются более дешёвой пехотой. И если игрок быстро поймёт, что это плохая тактика, попав в неё один или несколько раз, то искусственный интеллект не перестанет этого делать. В военных симуляторах это выглядит странно.

Другая проблема связана с тем, что стратегии содержат достаточно сложные комплексные механики, влияющие друг на друга, поэтому их сложнее балансировать, чем другие жанры игр. Игра не должна быть ни слишком лёгкой, ни слишком сложной. Для интересной игры стратегия должна быть сложной настолько, чтобы игрок был вынужден прилагать

усилия, но мог её пройти. Из-за сложностей настройки искусственного интеллекта, повышение уровня сложности в стратегиях достигается не за счёт совершенствования применяемой противником тактики, а за счёт изменения числовых параметров: увеличения урона, наносимого противником, увеличения его ресурсов, штрафов игрока и тому подобного. Такой способ изменения сложности воспринимается игроками как искусственный и плохой, ухудшающий игровой опыт. Игроки бы предпочли, чтобы сложность можно было бы увеличить, изменяя поведение искусственного интеллекта.

Важным фактором эффективности в бою является боевой строй юнитов. Научив юнитов подстраивать его под окружающую обстановку и действия игрока можно повысить эффективность игры искусственного интеллекта.

Сделать это можно следующими способами. Во-первых. Можно заранее прописать искусственному интеллекту возможные тактики. Обычно часть тактик действительно прописана вручную, но продумать их все невозможно, потому что в таком случае возникает комбинаторный взрыв – число комбинаций столь велико, что предусмотреть всё невозможно. В конечном итоге, чем сложнее будет алгоритм действий, тем он становится ненадёжнее, имеет больше шансов дать сбой.

Во-вторых, можно использовать нейросеть для управления искусственным интеллектом. В таком случае компьютер мог бы быстро реагировать на действия игрока и быстро подстраиваться под изменяющуюся обстановку. Минусом их использования является то, что для обучения нейросетей требуется большое число данных. Обучить их сразу на компьютере игрока становится невозможно, значит нужно обучать нейросеть заранее до выпуска игры. Поэтому пока что их нельзя эффективно использовать для данной задачи.

В-третьих, можно разработать для данной задачи модель на основе генетического алгоритма. Суть метода состоит в том, чтобы каждую итерацию изменять строй отрядов случайным образом и сохранить из них только те, которые покажут максимальную эффективность. Поэтому он и называется

генетическим – по аналогии с естественным отбором в природе, скрещиваниями и мутациями. Такая модель может относительно быстро если не найти самое оптимальное, то хотя бы исключить неэффективные построения. Такая модель будет работать с некоторым запозданием, так как выводы она будет делать уже по итогам прошедшей итерации, но в следующих боях полученную информацию будет можно применить [17]. Поэтому она хорошо годится для одиночных игр с продолжительной сюжетной кампанией. Именно данный способ выбран в данной работе из-за его простоты и надёжности.

1.4 Заключение по главе

В рамках первой главы был рассмотрен рынок видеоигр в России и мире, в частности жанра стратегий, были рассмотрены основные игровые механики, присущие играм этого жанра, дана классификация стратегий и выделены сегменты их целевой аудитории. Резюмируя всё вышесказанное, игры в жанре стратегий имеют свою аудиторию и становятся всё более актуальными. Поэтому этот жанр перспективен для разработки игр и их последующей монетизации. Но из-за конкуренции разработчикам следует использовать все доступные средства для оптимизации разработки, повышения качества игры, её конкурентоспособности.

Стратегии имеют ряд проблем, связанных со сложностью их игровых механик и с ограничениями искусственного интеллекта: относительно сложно увеличить в игре уровень сложности, не ломая баланс, и улучшив поведение юнитов.

Для усовершенствования искусственного интеллекта было предложено использовать генетический алгоритм, в силу его простоты и надёжности.

2. ТЕОРЕТИЧЕСКАЯ МОДЕЛЬ АЛГОРИТМА

2.1 Требования к программному обеспечению, необходимому для исследования

Для проведения исследования необходимо интегрировать генетический алгоритм в какую-либо компьютерную игру в жанре стратегия. Использовать существующие проекты затруднительно либо вовсе невозможно по следующим причинам.

Было решено разработать свой собственный прототип по ряду причин:

1. Использовать чужие игры невозможно, так как большинство игра на рынке – коммерческие проекты, не имеют открытого исходного кода. Своя собственная разработка позволяет более точно настраивать проект под свои нужды. Кроме того, Unreal Engine 5 содержит систему визуального программирования Blueprint, позволяющую относительно просто и быстро разрабатывать игровые механики.
2. Во многих стратегиях на исход боя могут оказывать влияние какие-либо механики, являющиеся лишними для данной работы, усложняющие балансировку и как следствие мешающие разработке модели. Так во многих так называемых «4-х» стратегиях есть экономический элемент – механика зарабатывания денег, менеджмент ресурсов и тому подобное – который оказывает прямое влияние на ход боя, но в рамках настоящего исследования не нужен [14].

В связи с этим было решено разработать прототип компьютерной игры, в котором можно самостоятельно реализовать заявленный генетический алгоритм, а также собирать статистические данные для проверки его работоспособности. Для этих целей предлагается использовать Unreal Engine 5. Это последняя версия специализированного программного обеспечения для разработки компьютерных игр (так называемый “игровой движок”), разработанная компанией Epic Games. Unreal Engine работает на

операционных системах Windows и macOS и подходит как для разработки небольших инди-игр, так и для крупнобюджетных блокбастеров. На нём можно создавать игры для следующих платформ: Windows, Mac, PlayStation, Xbox, iOS, Android, HTML5 и Linux. В самой простой форме Unreal Engine 5 является коллекцией редакторов, используемых в различных стадиях производства игр или приложений. Для разработки в данном программном обеспечении используется язык C++, а также система визуального программирования Blueprint [18].

Как уже было сказано выше нам необходима компьютерная игра в жанре стратегия, включающая в себя основные игровые механики, присущие военным стратегиям в жанре варгейм. Она будет использоваться для проведения экспериментов и сбора статистики по работе с генетическими алгоритмами в искусственном интеллекте.

Для анализа того, что в итоге должно содержать в ней содержаться, рассмотрим стратегии компании Eugen Systems - французского разработчика игр, известного такими сериями продуктов, как Act of War, Wargame, Steel Division. Эти игры являются типовыми для жанра стратегий в реальном времени [12]. Нам нужно в упрощённом виде воссоздать их механики боя, отбросив те, которые не важны для эксперимента и не касаются эффективности работы искусственного интеллекта и взаимодействия юнитов [19].

Таким образом к разрабатываемому прототипу игры предъявляются следующие требования. Жанр – стратегия в реальном времени. Кор-геймплей следует строить вокруг военного столкновения двух сторон, именно его нужно моделировать. Условие победы – уничтожение войск противника или захват базы противника – особой точки на карте. Основные типы юнитов в варгеймах следующие, именно они будут представлены в исследовании:

1. Пехота – отделение из нескольких человек.
2. Автомобили.
3. Лёгкая бронетехника (БТР);

4. Танки.

Во многих военных стратегиях присутствует возможность занимать здания пехотой, что активно используется в боях, особенно на городской местности. Поэтому было решено добавить данную игровую механику в прототип. Постройки дают бонусы к защите юнитов, укрывшихся в них. В имеющихся играх наиболее часто она реализуется двумя разными способами:

1. солдаты могут войти, полноценно перемещаться в здании и вести огонь из окон. При штурме бой может завязаться прямо внутри здания и вестись за каждую комнату. Так работает эта механика в такой игре как «Сирия. Русская буря» [20].
2. Солдаты при приближении к зданию удаляются, в здании при этом появляются огневые точки как отдельные юниты. При выходе из здания огневые точки удаляются, рядом со входом появляется пехота. Так реализована механика в RTS «Command & Conquer 3: Tiberium Wars».

В работе будет использоваться второй способ для быстрого прототипирования, так как он гораздо более прост и надёжен.

В проекте важно, как именно юниты будут удерживать строй, как будут понимать, где находится позиция в строю каждого из них. Данную игровую механику можно реализовать следующим образом. Каждый юнит, как и во всех остальных играх не будет знать, где находятся его союзники. За связь между ними будет отвечать отдельный актор. Он будет содержать в себе несколько разных точек, расположение которых похоже на шахматную доску. При старте игры он получает на вход матрицу, в которой зашифровано расположение юнитов, и распределяет войска по ним. Каждому юниту в отделении соответствует одна такая точка, то есть своя определённая позиция. Войска всё время считывают их позиции, поэтому если этот актор перемещается, то и юниты следуют за ним, он выступает в роли посредника при передаче информации. Таким образом юниты не видят друг друга, но сами знают где должны находиться в сражении.

Причём для реализации некоторых других игровых механик передача информации может происходить между юнитами напрямую. Так во многих варгеймах присутствует “туман войны” – игровая механика, связанная с маскировкой и обнаружением противника, при которой стороны не знаю о местоположении противника до тех пор, пока не обнаружат его, только после этого его можно атаковать. При обнаружении врага юнит сможет передавать информацию о нём сразу всей команде без посредничества других акторов.

Реализовать весь алгоритм сразу в юнитах невозможно. Получилось бы что матрица с боевыми порядками хранится в нескольких акторах сразу, это затруднило бы процесс мутаций, так как он бы проходил в нескольких местах сразу с разными результатами. Поэтому все данные и вычисления должны храниться в одном отдельном объекте, который бы уже в свою очередь передавал результаты работы алгоритма отделениям [21]. Причём так как все вычисления, произведённые моделью, нужно сохранять при включении нового уровня, то реализовать её логику следует не в обычном акторе, а в объекте типа Game Instance – это особый класс в Unreal Engine, позволяющий сохранять данные при переходе между локациями [22].

В проекте следует представить разные типы местности, по-разному влияющие на характеристики юнитов:

1. Поля/равнины – никак не влияют на параметры юнитов.
2. Лес – значительно снижает дальность обзора, но значительно увеличивает маскировку.
3. Городская застройка – характеризуется высокой плотностью зданий, которые преграждают обзор.

Камера в стратегиях традиционно располагается сверху и направлена вниз на происходящее, она “летает в небе”. Таким образом игрок может видеть большую часть локации с происходящими на ней событиями [23].

Продукт должен быть способен поддерживать сражения как между игроком с одной стороны и ботом с другой, так и только между ботами. Для возможности быстрого моделирования большого числа боёв при тестах

следует предусмотреть возможность замедления и ускорения времени в стратегии.

Искусственный интеллект ботов обладает следующими умениями:

1. Определять точки интереса – позиции на карте, которые следует захватить или оборонять от противника.
2. Перемещение юнитов в точку интереса.
3. Атака определённого юнита противника своим.
4. Оборона точки интереса.
5. Распределение юнитов по боевым группам - отделениям.
6. Генерация боевого построения для отделений.

Следует предусмотреть возможность задать боевой строй для бота на все последующие сражения перед и отключить возможность его оптимизации одной из сторон - для сбора статистики о том, насколько эффективнее него играет ИИ с оптимизацией боевого строя.

Принципиальным отличием работы искусственного интеллекта в настоящей работе от используемого в других существующих компьютерных играх является умение не просто сохранять юнитами некоторый боевой строй, но и возможность оптимизировать его для разных типов местности с помощью эволюционного генетического алгоритма. Он может учиться на своих ошибках, запоминать их и подбирать более оптимальный строй для того или иного ландшафта [24]. Более подробно работа предлагаемого алгоритма описана в следующей главе настоящего исследования.

Перед запуском боя игрок выбирает состав армии. Его противник получает армию с таким же составом или иным. В бою у каждой стороны может быть несколько таких отделений – их число будет задаваться при запуске игры, у ботов они также сразу выступают в роли популяции, заданного размера. Предлагаемая модель должна работать как в симметричном балансе, так и при асимметричном. Однако для чистоты эксперимента в первую очередь будет использоваться симметричный баланс: все стороны будут начинать игру с одним и тем же набором юнитов [25].

Стороны размещают своих юнитов на начальных позициях. После чего начинается бой.

Стратегии характеризуются тем, что юниты самостоятельно выполняют приказы игрока, который управляет ими через команды [26]. Игроку доступны следующие возможности управления:

1. Выделение своего юнита кликом левой кнопкой мыши. Выделенный юнит подсвечивается.
2. Выделение группы юнитов мышью.
3. Приказ выделенным юнитам переместиться в выбранную точку нажатием по ней правой кнопкой мыши.
4. Приказ выделенным юнитам атаковать выбранную цель, нажатием на ней правой кнопкой мыши.
5. Камеру по локации можно перемещать, наводя курсор на край экрана или стрелками на клавиатуре – после чего камера сдвигается в указанную сторону.
6. Камера имеет возможность приближения к земле и отдаления с помощью прокрутки колёсика мыши.
7. При старте в левом верхнем углу отображается меню со списком доступных юнитов. Игрок может перетаскивать их на локацию для развёртывания. После этого игроку нужно нажать кнопку “старт”.

Причём юниты могут атаковать противника, попавшего в зону досягаемости их оружия, без приказа игрока. Данное управление также является классическим и встречается в подавляющем большинстве стратегий.

2.2 Модель работы генетического алгоритма

Особых требований к генерации особей первой популяции не предъявляется, так как нет никаких данных для анализа. Поэтому генерировать боевой порядок для них можно случайным образом.

Во-первых, модели нужно считать информацию о количестве имеющихся юнитов разных типов. Из них состоит особь.

Размер построения пять линий в пять эшелонов, но он может быть при необходимости изменён. Для записи этого значения следует использовать целочисленный тип данных [27].

Имеющихся юнитов нужно распределить по ним для формирования боевого строя. Его можно представить в виде матрицы пять на пять, содержащей следующую информацию о том, какой юнит располагается в том или ином месте. Тип юнита закодирован цифрой:

- Пехота – 4.
- Автомобиль – 3.
- Бронетранспортёры – 2.
- Танк – 1.

Юниты внутри эшелона распределяются относительно друг друга случайным образом. Расстояние между юнитами внутри эшелона и между эшелонами задаётся заранее и в процессе экспериментов не меняется. Первый эшелон расположим дальше остальных, чтобы он мог выступать в качестве разведки. Причём при генерации строя происходит проверка на то, нет ли такой же матрицы у другой особи. Если есть – то нужно сгенерировать другую заново.

Допустим, если нужно записать расположение четырёх групп пехоты, БТР и трёх танко, то матрица может выглядеть как на рисунке 2.1.

Эшелон \ Ряд	1	2	3	4	5
1			2		
2	4		4		4
3	1	4			1
4			1		
5					

Рисунок 2.1 – Пример матрицы с распределением юнитов по эшелонам

На данном рисунке в первом эшелоне разведки наступает БТР, на расстоянии от него во втором ряду следует три группы пехоты, за ними следуют танки и ещё одна группа пехоты.

Генерируется несколько матриц, отдельных для каждого типа местности в игре – для данного исследования их три. Именно значения этих матрицы будут перебираться для оптимизации целевых функций [28]. Целевых функций у группы также несколько – отдельная для каждого типа местности. Они оптимизируются отдельно друг от друга. Группа всегда удерживает одно боевое построение, но при входе на другой тип местности она меняет строй.

Отделение обладает полной информацией о типах местности на локации. Ему всегда известен ландшафт, на котором он находится. Данная игровая механика будет реализована через коллизии – то есть проверку того, не зашла ли группа на новую территорию. Коллизия – в Unreal Engine это компонент внутриигрового объекта, обладающий своей формой и геометрией, позволяющий считывать взаимодействия с другими внутриигровыми объектами [29]. На локации на каждом типе местности будут размещены уникальные акторы с коллизией в форме простых геометрических фигур (сферы, прямоугольники). При входе в коллизию (т.е. при переходе на новую местность) группа получает сигнал от системы о том, в какой тип местности она попала, и перестраивается.

Целевая функция (или же функция приспособленности) показывает, насколько эффективен тот или иной боевой порядок на данном типе местности [30]. Как уже было сказано выше, популяция имеет несколько целевых функций – по одной для каждого типа местности. Каждый тип юнитов обладает определённой ценностью, которая будет использоваться для подсчёта эффективности особи с тем или иным строем. Некоторые юниты мощнее остальных и поэтому обладают большей ценностью. Нужно подсчитать сколько баллов заработало отделение за уничтожение противника в бою. Необходимо учитывать, что некоторые отделения могут вообще не

поучаствовать в игре, например, вступить в сражение под конец матча или просто не вступить в бой на каком-либо отдельном типе местности. Поэтому могут не нанести достаточно урона противнику и не заработать баллы для целевой функции, но при этом они могут являться потенциально эффективными. Это нужно отразить в целевой функции. Поэтому отдельно следует также подсчитать сумму очков за свои собственные потерянные юниты. Предлагаем для расчёта целевой функции использовать соотношение очков за уничтожение противников и очков за потерю группой собственных юнитов в бою на той или иной местности.

Ценность всех типов юнитов следующая (теоретически она может измениться в процессе экспериментов для балансировки):

1. Пехота – 1 единица.
2. Автомобили – 2 единицы.
3. Лёгкая бронетехника и артиллерия – 3 единицы.
4. Танки – 5 единиц.

Причём если отряд по каким-то причинам не понёс потери, то получилось бы, что знаменатель равен нулю. Чтобы избежать ситуаций, когда очки, набранные за уничтожение юнитов, делятся на ноль, добавим к знаменателю единицу. Таким образом он никогда не будет нулевым, и функция будет рассчитываться корректно [24]. Таким образом целевая функция приобретает вид, представленный на рисунке 2.2, где:

1. z – значение целевой функции.
2. v – ценность данного типа юнитов.
3. f - количество уничтоженных юнитов данного типа.
4. l - количество потерянных собственных юнитов данного типа.
5. n – количество различных типов юнитов в игре.

$$Z = \frac{\sum_{i=1}^{n} v_i \cdot f_i}{1 + \sum_{j=1}^{n} v_j \cdot l_j}$$

Рисунок 2.2 – Целевая функция для описанного генетического алгоритма

Чем больше баллов заработает особь за бой, тем лучше её целевая функция. Наилучший строй будет выявляться путём сравнения между собой информации о всех проверенных в боях.

Для получения новых генов у существующих особей используются мутации – преобразование случайно отобранных генов в другие, то есть случайное изменение какого-либо параметра [31]. Пример результата мутации представлен на рисунке 2.3. Красным подсвечен мутировавший ген, отвечающий за распределение бронемашин по эшелонам. После такой мутации пехота будет идти первой, а БТР сдвигается из авангарда дальше в третий ряд, что может как способствовать повышению её выживаемости и выживаемости всей группы, так и нет.

Для получения новых особей с новыми характеристиками используется скрещивание – создание новой дочерней особи («ребёнка»), путём объединения нескольких генов двух других родительских особей. Скрещивание – одна из ключевых особенностей генетического алгоритма. Причём для скрещивания рекомендуется использовать не только самых приспособленных особей из данной популяции, но и вообще всех. Так как это способствует повышению разнообразия и позволяет уменьшить вероятность появления особей, которые хороши сейчас по случайности, но которые являются тупиковыми ветками развития, приводящими функцию приспособленности к неэффективному локальному максимуму [32].

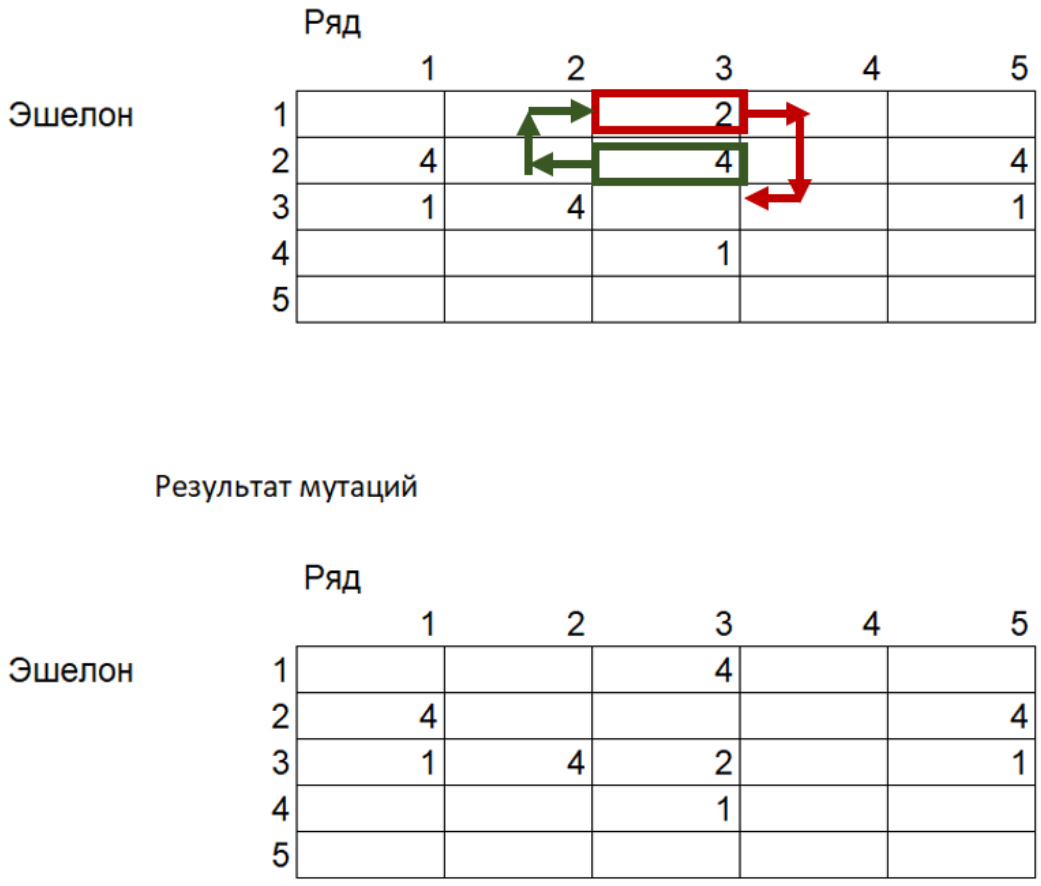


Рисунок 2.3 – Иллюстрация действия мутации

Для скрещивания необходимо каким-то образом отобрать две особи, которые будут обмениваться частями своих геномов. Существуют несколько подходов к выбору родителей:

1. Панмиксия – при использовании данного подхода из популяции выбирается несколько (обычно две) особи случайным образом. Они становятся родителями. Данный метод характеризуется тем, что все особи имеют одинаковый шанс случайно стать родителем.
2. Инбридинг – при использовании данного подхода только первый родитель выбирается случайным образом, а вторым становится та особь, которая оказалась ближе к первой. Особенность данного метода в том, что он концентрирует поиск в локальных узлах, что используется для поиска экстремума целевой функции в найденных местах.

3. Аутбридинг – при использовании данного подхода, как и в случае с инбридингом, первый родитель выбирается случайным образом. Вторым родителем становится особь, расположенная максимально далеко от первой. Данный метод напротив используется для того, чтобы не допустить застревания алгоритма в локальном максимуме, так как он подталкивает к исследованию новых неисследованных областей [33].

Для использования в работе выбраны методы панмиксии и аутбридинга. При каждом скрещивании будет случайным образом выбираться один из этих двух подходов. Получение новой особи с помощью скрещивания двух представлен на рисунке 2.4. Красным выделены те области, которые берутся у неё для передачи следующей популяции. В данном примере получилось боевое построение, развёрнутое почти в одну линию и действующее широким фронтом.

Рисунок 2.4 – Пример скрещивания

Существует несколько способов вычисления расстояния между двумя особями. Основными из них являются евклидово расстояние (или же евклидова метрика), манхэттенское расстояние (оно имеет множество других названий, таких как метрика L1, расстояние городских кварталов, расстояние такси, метрика прямоугольного города) и расстояние Чебышёва (оно же метрика шахматной доски) [19].

Евклидова метрика представляет собой расстояние между двумя точками x и y на плоскости (в данном исследовании это две особи), измеренное по кратчайшей прямой линии. При расчёте евклидова расстояния между двумя особями используется следующий алгоритм действий:

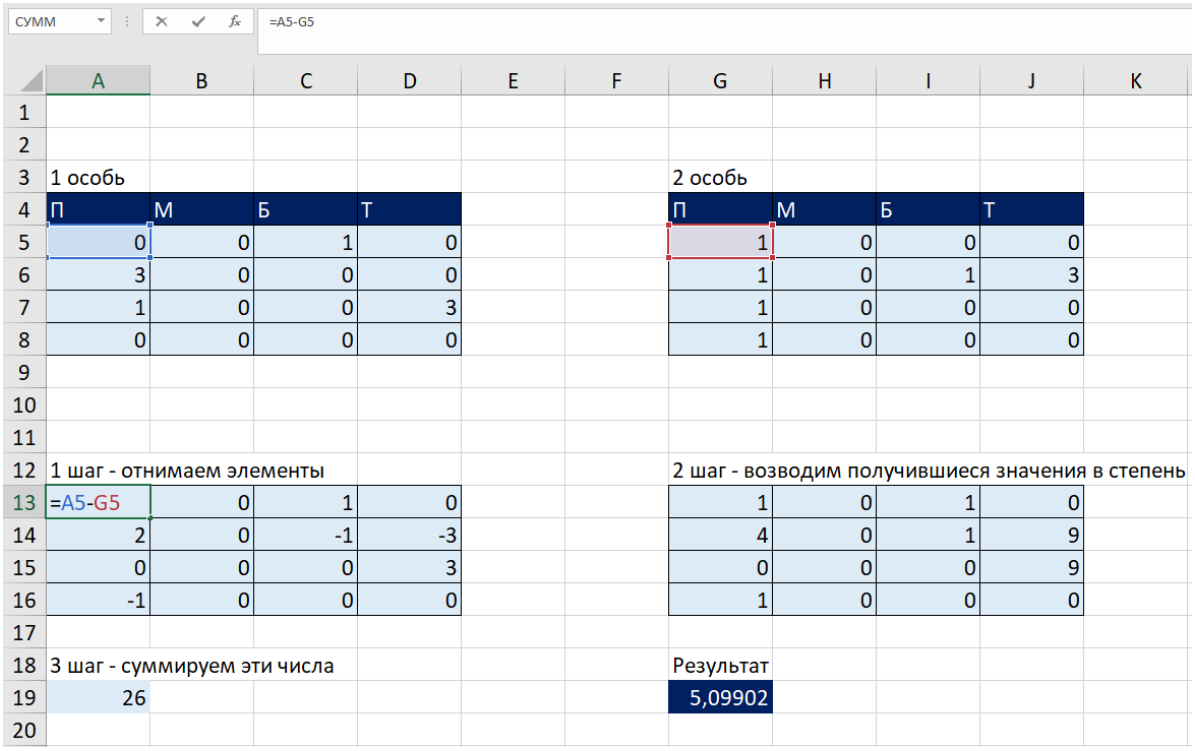
1. От каждого элемента первой матрицы отнимается соответствующий элемент второй. x_{ij}-y_{ij} – где x – элемент первой матрицы, а y – второй, i и j – номера строк и колонок, то есть позиция элемента в матрице.
2. Полученные значения возводятся во вторую степень.
3. Они суммируются.
4. Из найденной суммы извлекается корень. Полученное число и является итоговым значением.

Формула для расчёта евклидова расстояния представлена на рисунке 2.5.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}-y_{ij})^2}$$

Рисунок 2.5 – Формула для расчёта евклидова расстояния

Пример расчёта евклидова расстояния между двумя особями, реализованный средствами Microsoft Excel, представлен на рисунке 2.6 [20] [21].



	1 особь					2 особь			
	П	М	Б	Т		П	М	Б	Т
	0	0	1	0		1	0	0	0
	3	0	0	0		1	0	1	3
	1	0	0	3		1	0	0	0
	0	0	0	0		1	0	0	0
	1 шаг - отнимаем элементы					2 шаг - возводим получившиеся значения в степень			
	=A5-G5	0	1	0		1	0	1	0
	2	0	-1	-3		4	0	1	9
	0	0	0	3		0	0	0	9
	-1	0	0	0		1	0	0	0
	3 шаг - суммируем эти числа					Результат			
	26					5,09902			

Рисунок 2.6 – Расчёт евклидова расстояния между двумя особями в Microsoft Excel

Манхэттенское расстояние представляет собой расстояние между двумя точками, измеренное линиям, проведённым вдоль осей X и Y. Оно рассчитывается по следующему алгоритму:

1. Первый шаг такой же, как и при расчёте евклидова расстояния – отнять от элементов первой матрицы соответствующие им элементы второй.
2. Необходимо рассчитать модуль всех полученных чисел.
3. Суммируем имеющиеся значения. Полученное число и есть манхэттенское расстояние.

Формула для его расчёта представлена на рисунке 2.7.

$$d(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{m}|x_{ij} - y_{ij}|$$

Рисунок 2.7 – Формула для расчёта манхэттенского расстояния

Реализация расчёта манхэттенского расстояния с использованием Microsoft Excel представлена на рисунке 2.8.

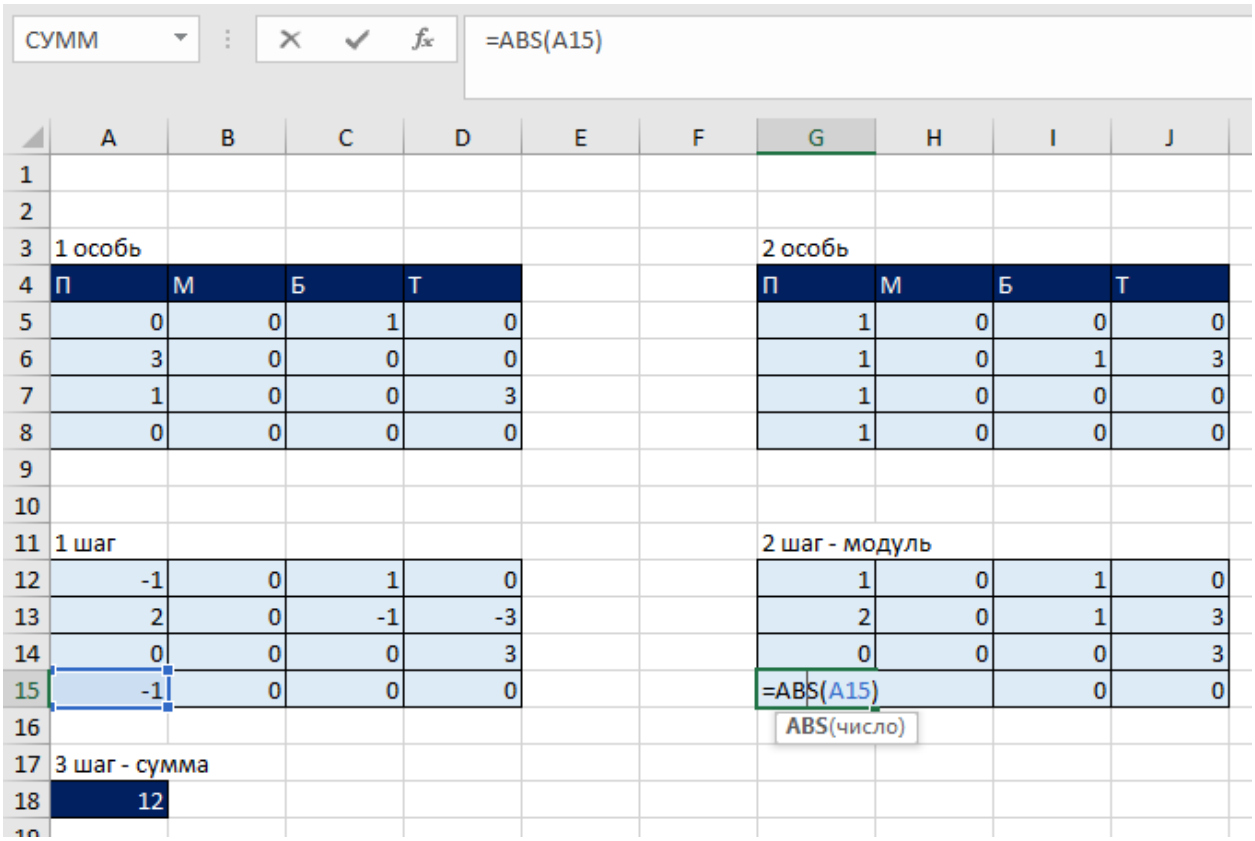


Рисунок 2.8 – Расчёт манхэттенского расстояния между двумя особями с помощью Microsoft Excel

Чтобы понять, что такое расстояние Чебышёва, можно представить шахматную доску и посчитать максимальное число ходов, которые можно сделать от каждой точки нужно сделать до другой [22]. Расстояние Чебышёва рассчитывается по следующему алгоритму:

1) Первый шаг совпадает с двумя описанными выше методами расчёта расстояния – от каждого элемента первой матрицы отнимается соответствующий элемент второй матрицы.
2) Второй шаг совпадает со вторым шагом при расчёте манхэттенского расстояния –у полученных на первом шаге значений рассчитываются модули.
3) На третьем шаге среди всех имеющихся значений выбирается максимальное. Оно является расстоянием Чебышёва.

Формула для вычисления расстояния Чебышёва представлена на рисунке 2.9.

$$d(x, y) = \max_{\substack{i = 1 \dots n, \\ j = 1 \dots m}} |x_{ij} - y_{ij}|$$

Рисунок 2.9 – Формула для вычисления расстояния Чебышёва

Пример вычисления расстояния Чебышёва с помощью Microsoft Excel представлен на рисунке 2.10.

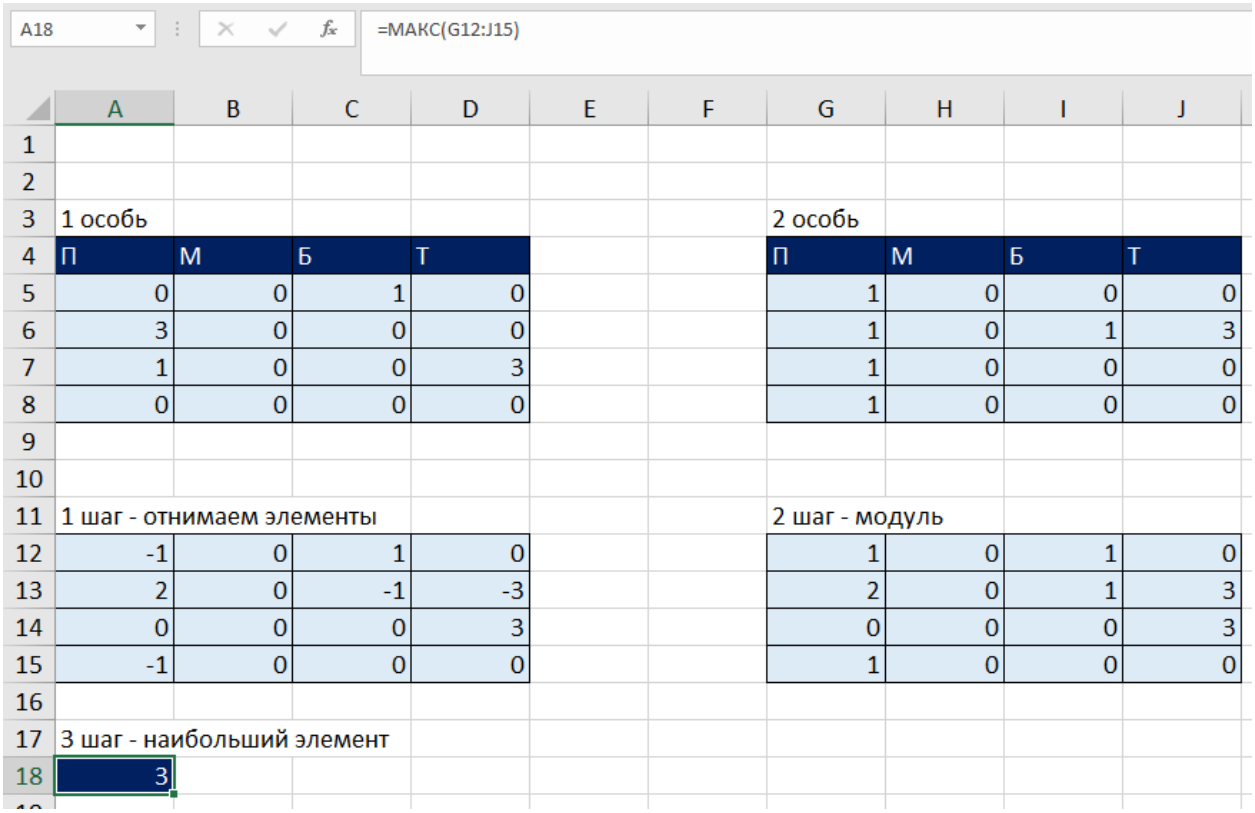


Рисунок 2.10 – Вычисление расстояния Чебышёва между двумя матрицами в Microsoft Excel

В данной работе предлагается использовать евклидово расстояние. На следующем этапе исследования алгоритм расчёта различий между особями, смоделированный в Microsoft Excel и представленный на рисунке 2.6, будет

реализован в Unreal Engine 5 через язык программирования C++ и систему Blueprint.

Селекция – отбор наиболее приспособленных особей для дальнейшего развития – происходит следующим образом. После боя у всех особей считывается информация об их боевом построении и значении соответствующей целевой функции для каждого типа местности. После первой итерации для каждого типа местности отбирается не более 40% наиболее эффективных построений. Из них будут формироваться новые популяции. В следующем бою не более 40% групп будут использовать эти боевые порядки, изменённые с помощью мутации случайным образом. На второй и последующих итерациях для отбора будут сравниваться между собой вообще все построения, ранее используемые в эксперименте, а не только те, которые использовались в последнем бою. Не более 40% новых особей будет получено с помощью скрещивания отобранных построений. Остальные будут сгенерированы заново. Причём такой отбор будет производиться независимо для каждого типа местности отдельно. То есть возможна ситуация, при которой у одной и той же группы будет отобрана матрица для ведения боевых действий в лесистой местности, но отвергнута отвечающая за боевой порядок на равнине, при следующей итерации она будет сгенерирована заново. Подобный подход при селекции с одной стороны позволяет выделять и сохранять наиболее приспособленных особей, а с другой получать большое количество генов с новыми параметрами при генерации популяций [23].

Некоторые группы юнитов могут не вступить в контакт с противником вообще, а значит эффективность её построения в данной итерации не удастся проверить. Для того, чтобы всё же проверить их эффективность в бою, лучше переносить их в следующую итерацию, не изменяя их строй. Таким образом если группа не вступила в бой, то в следующем одно отделение примет такое же построение, а мутации и скрещивания будут продолжены на оставшихся свободных отделениях.

Состав армии в разных боях может отличаться. Нельзя начинать оптимизацию каждый раз сначала, когда в отделении меняется хотя бы один юнит. Нужно иметь возможность применять алгоритм в играх с большим числом юнитов. Так в стратегиях компании Eugen Systems (разработчик таких серий игр как Wargame и Steel Division) у некоторых фракций может быть несколько десятков разновидностей пехоты и бронетехники [34]. Так как их различия незначительны, то ими можно пренебречь при выборе их позиции в строю. Причём чем юнит дороже в игре, тем выше его боевая мощь. Для эффективной работы алгоритма нужно учесть возможность его обучения на одном наборе юнитов с использованием полученной информации на другом. Это можно реализовать следующим образом. Если состав отделения изменится, то нужно сравнить старый состав с новым. Место юнитов в новом строю определяется следующим образом:

1. Если какой-то класс войск был в отделении и раньше, то его нужно разместить на те же места. Например, если в прошлом в первом строю стояла пехота, то и в этот раз её расположат там же.
2. Затем, если в строю остаются свободные места, где раньше располагались юниты, то они заполняются наиболее похожим на них типом войск. Например, если раньше в отделении присутствовали БТР, а в новом отделении их нет, но зато там появились танки, то танки нужно поставить на те же места, где ранее располагались БТР. Каждый юнит имеет определённую стоимость: число баллов, которые зарабатывает противник при его уничтожении. Для пехоты это 1 балл, для машин – 2 балла, для БТР – 3 балла и для танков – 5. С помощью этих баллов в прототипе определяется насколько типы юнитов схожи.
3. Если уже заняты все позиции, где ранее располагались войска, но ещё остались новые нераспределённые юниты (такое может быть, если юнитов в новом бою в отделении больше), то они распределяются произвольным образом на свободные места.

Для иллюстрации рассмотрим следующую ситуацию. Допустим в старых боях улучшался строй взвода, состоящего из трёх отрядов пехоты и двух БТР. В новом бою состав отделения изменился и теперь оно состоит из трёх отрядов пехоты, танка и двух машин. Пример работы такого алгоритма представлен на рисунке 2.11. При необходимости строй этого отделения можно оптимизировать дальше в следующих итерациях.

Старый боевой порядок

	Ряд				
Эшелон	1	2	3	4	5
1	Пехота		Пехота		Пехота
2		БТР		БТР	
3					

Образовавшийся боевой порядок

	Ряд				
Эшелон	1	2	3	4	5
1	Пехота		Пехота		Пехота
2		Танк		Авто	
3	Авто				

Рисунок 2.11 – Иллюстрация использования старого боевого порядка на новом отделении

Функция приспособленности одного и того же отделения на разных итерациях может иметь разное значение, даже если входные данные не изменяются. Это объясняется тем, что в играх некоторые события могут происходить случайным образом. Например, в одной и той же ситуации юнит может и уничтожить противника с первого раза, и промахнуться мимо него. Нами было установлено, что целевая функция периодически может случайно

принимать высокие значения, приводящие к выходу из алгоритма. Наличие выбросов при сборе статистики – это нормально и естественно, но нам нужно минимизировать их влияние на работу модели. Как понять, что высокое значение функции приспособленности обусловлено работой алгоритма, а не случайным стечением обстоятельств на поле боя? Для выхода из алгоритма нужно создать такое условие, на которое не будут оказывать влияние единичные выбросы:

1. На первой итерации проверяется какую эффективность имеет начальное боевое построение. После этого начинается процесс мутаций.
2. Если в каком-то из боёв целевая функция превышает ту, которая была на первой итерации, на заданное число – можно взять в полтора раза – то процесс мутаций останавливается. Но работа всей модели на этом не прекращается.
3. На протяжении трёх следующих боёв проверяется значение целевой функции, из них высчитывается медианное значение. Если оно также в полтора раза больше, чем было на старте оптимизации, то значит построение действительно можно считать оптимальным и остановить алгоритм. Если нет – то можно считать, что ранее произошёл выброс в большую сторону и продолжить перебор возможных вариантов построений.

В игре может возникнуть ситуация, когда попытка улучшения построения приводит к снижению боевой эффективности отделения. Такое возможно, когда изначальный боевой порядок сразу являлся оптимальным для действий в данных условиях [33]. Нужно ограничить работу алгоритма в таких ситуациях. Он должен прекратить свою работу после заданного числа боёв, если функция приспособленности на их протяжении будет значительно ниже, чем на первой итерации. Для разрабатываемого алгоритма зададим 20% или более. Нельзя прерывать алгоритм сразу же, когда функция приспособленности станет низкой, так как к этому могут привести случайные

стечения обстоятельств в бою. Чтобы убедиться, что это не выброс, прерывать алгоритм можно только если целевая функция низкая на протяжении нескольких итераций подряд. То есть как минимум двух. При этом модель должна иметь шанс протестировать иные варианты построений, чтобы убедиться, что улучшить ситуацию не удаётся. Перебирать варианты в поисках лучшего бесконечно долго тоже нельзя, так как в такой ситуации эффективность игры искусственного интеллекта будет снижена, что противоречит задаче модели. Поэтому их число тоже надо ограничить. Для разработки алгоритма возьмём пять итераций. Таким образом, если на протяжении пяти боёв подряд с начала оптимизации значения функции приспособленности будут на ниже более чем на 20% того, что было в первом бою, то можно допустить, что самый первый стартовый строй был хорош изначально и в дальнейшем использовать его.

Эта ситуация маловероятна из-за особенностей работы алгоритма – так как на каждом этапе отбираются наиболее боеспособные особи, но возможна. Поэтому ожидаемый выигрыш от использования генетического алгоритма всё равно больше, чем без него.

Разные бои могут протекать по-разному. Игрок может поменять предпочитаемую тактику для противодействия противнику. В таком случае оптимизацию построения нужно запустить снова. Нужно условие повторного запуска алгоритма для приспособления к новым условиям среды. Если уже после выхода из алгоритма в нескольких итерациях значение целевой функции опускается ниже заданного значения, то оптимизацию следует запустить снова. Это произойдёт, если целевая функция в течении 2 боёв подряд будет опускаться ниже на 20% или сильнее, чем была во время прошлого выхода из алгоритма.

2.3 Отслеживание результатов работы модели

Для анализа эффективности модели нужно проверить увеличивается ли со временем функция приспособленности отделений. Чтобы снизить влияние выбросов следует не просто брать значение функций каждой группы по

отдельности, а высчитывать среднее значение целевых функций всех подразделений на поле боя, имеющих одно построение [35].

На каждой итерации нужно сохранять эти значения в отдельный файл, чтобы их можно было использовать при подготовке статистики. Для обработки данных отлично подходит программа Microsoft Excel, поэтому сохранение будет производиться в файл типа csv. Однако при экспорте в MS Excel данных нужно учитывать важный нюанс. В C++ целую часть числа и дробную отделяет точка, а в Microsoft Excel запятая. Точки там используются при записи дат. Причём сами даты перекодируются в целочисленные значения и в таком виде хранятся в памяти. Например, после ввода даты 01.01.2023 она запишется как число 44927, 02.01.2023 зашифровано значением 44928 и так далее. После для вывода это целочисленное значение расшифровывается и отображается уже как дата [36]. Нельзя все данные сразу распределить по колонкам. Это приводит к следующей ситуации. Если, например, после итерации значение целевой функции будет равно 1.2 и это число будет сохранено в csv файл, то из-за точки MS Excel воспримет это как 1 февраля текущего года. 01.02.2024 перекодируется в 45323. Из-за этого часть данных испортится и станет непригодной для анализа. Решить эту проблему можно двумя способами. Можно заменять точку на запятую прямо в C++. Либо можно все числа сохранять вместе в одну колонку, для разделения значений можно использовать, например запятые. MS Excel из-за нескольких точек и запятых в одной ячейке воспримет это как обычный текст и не станет заменять значения. Второй способ более просто использовать для данной работы. После все значения можно будет распределить по столбикам вручную уже средствами MS Excel, выбрав нужный формат для значений, чтобы они не переписывались в другие числа. Потом нужно вручную заменить точки в таблице на запятые, также используя инструменты MS Excel, чтобы он воспринимал их как числа, а не текст [37]. По получившимся значениям можно будет создать график и построить линию тренда для проверки эффективности модели.

2.4 Заключение по главе

Во второй главе данной работы были сформированы требования к прототипу игры и представлена модель генетического алгоритма, улучшающего боевой строй юнитов. Для реализации проекта выбрана программа Unreal Engine 5, так как она является одним из самых популярных игровых движков, имеющим мощный функционал и позволяющим быстро разрабатывать игровые механики. Был прописан порядок действий алгоритма в ряде ситуаций, которые могут случиться в игре. Сформированы условия повторного запуска алгоритма и выхода из него. Предусмотрено как минимизировать влияние выбросов, приводящих к досрочному выходу из алгоритма.

Отдельно рассмотрен ряд проблем с сохранением данных в программе Microsoft Excel, связанных с тем, что данное приложение имеет особенность перекодировать определённые значения в другие. Было предложены какие инструменты Microsoft Excel и как следует использовать, чтобы преодолеть эту проблему.

Таким образом представленный алгоритм можно представить в виде следующей блок-схемы, представленной на рисунке 2.12.

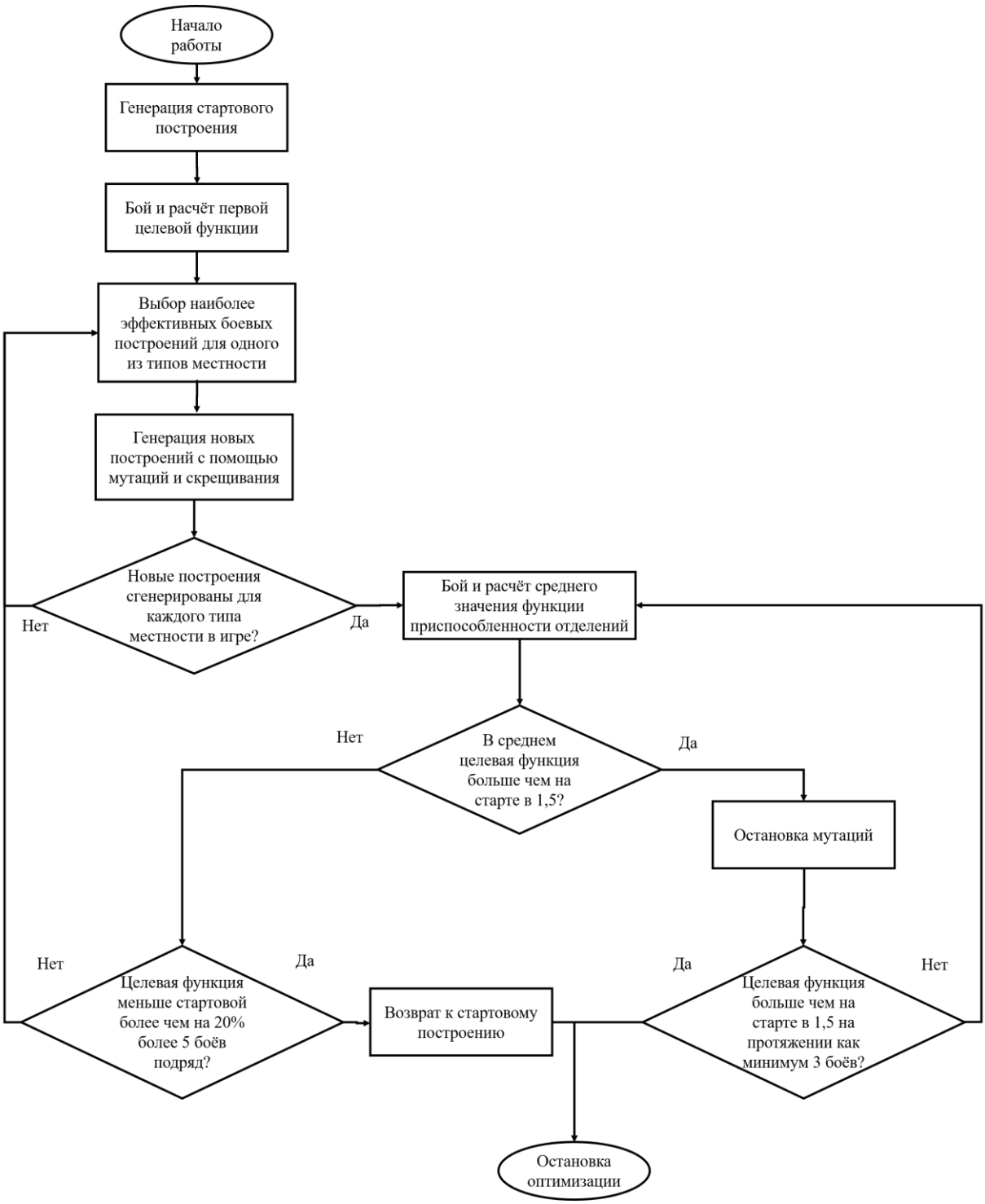


Рисунок 2.12 – Схема работы алгоритма

3. РЕАЛИЗАЦИЯ ПРОГРАММНОЙ ЧАСТИ

3.1 Структура прототипа игры

Для реализации проекта используется язык программирования C++ и система визуального скриптинга Blueprint.

Наиболее важные классы в проекте следующие:

1. BP_camera – камера игрока. Игрок в стратегиях не управляет конкретным персонажем, а отдаёт приказы юнитам. Соответственно камера нужна для наблюдения за полем боя. Также именно в этом акторе хранится некоторая логика, отвечающая за ряд игровых механик в бою. Например, именно в нём производится подсчёт уничтоженной техники для расчёта целевой функции.
2. BP_NPC_abstract – родительский класс для всех юнитов. Отвечает за большинство их игровых механик. Также от него унаследованы другие классы для реализации различных типов юнитов с разными характеристиками:
    2.1. BP_technic – класс, отвечающий за логику работы оружия и техники. От него унаследованы другие блюпринты для реализации разных типов юнитов:
        2.1.1. BP_Armor_Tank – танк, вооружён орудием с большим уроном, обладает высокой бронёй;
        2.1.2. BP_BTR – легкая бронетехника, вооружена скорострельной пушкой;
        2.1.3. BP_car – машина, вооружённая пулемётом.
        2.1.4. BP_soldier – пехота.
        2.1.5. BP_embrasure (от английского «амбразура», «бойница») – отдельный юнит-огневая точка, появляющаяся на стене здания и не способная перемещаться. Класс этого юнита унаследован от BP_soldier (пехоты), так как имеет с ним много общего.

Эта иерархия отражена на рисунке 3.1. Каждый тип юнитов обладает своими характеристиками и может быть по разному эффективен в той или иной ситуации и на различных типах местности. Юниты могут передавать друг другу информацию об обнаруженных противниках.

3. BP_squad (от английского «команда», «отделение») – актор, управляющий группой юнитов. Отвечает за то, чтобы они занимали удерживали определённую позицию в строю и двигались сообща. Причём юниты, как и во многих других игр, не видят другие союзные юниты напрямую. Однако через этот актор они знают свою позицию относительно остальных и тем самым могут поддерживать друг друга.
4. Game Instance:
    4.1. Game_Instance_Wargame – отвечает за работу генетического алгоритма. Реализован на C++. Изначально этот класс предназначен для того, чтобы сохранять информацию при переходе между локациями и перезагрузках [29]. Но в рамках исследования хранение данных о боевых построениях, а также действия над ними (мутации и т.п.) реализованы сразу в нём. Изначально планировалось использовать отдельный класс, отвечающий только за генетические алгоритмы, и отдельно использовать Game Instance. Но информацию о прошедших боях нужно постоянно сохранять и загружать, передавать её между этими двумя классами. Нужные данные в любом случае всегда бы хранились в Game Instance [22]. Поэтому было решено сразу использовать только его.
    4.2. BP_Game_Instance_Wargame – дочерний blueprint, предназначенный для возможности реализации части логики на блюпринтах.

5. BP_Lanshaft – игровой объект в виде куба, изменяющий характеристики попавшей в неё техники. Моделирует попадание в лесистую местность.
6. BP_City – игровой объект в виде куба, нужен для передачи информации о типе местности в котором находится группа. Характеристики напрямую не изменяет, но здания в городах будут значительно препятствовать обзору. Моделирует попадание в местность с плотной застройкой. Если говорить про типы местности, то в проекте также можно выделить равнины – ландшафт, не влияющий на характеристики юнитов. Они не имеют какого-то отдельного блюпринта.

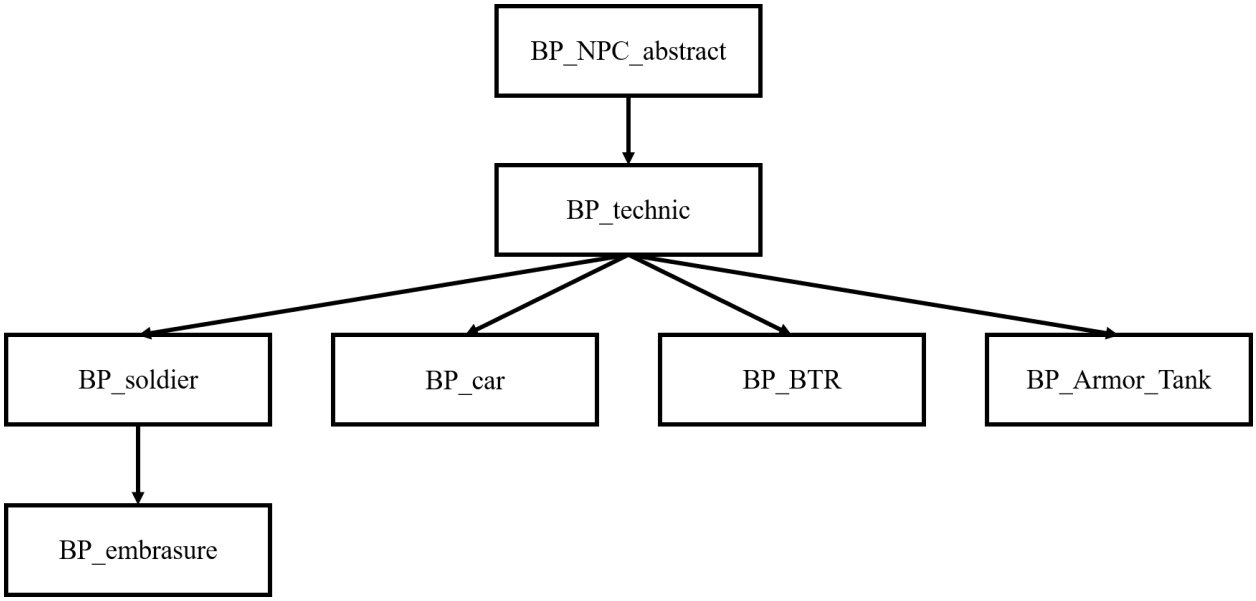


Рисунок 3.1 – Иерархия классов юнитов

Также для зданий были созданы два отдельных блюпринта: *BP_bunker* (от английского «бункер») – бункер, небольшая постройка, имеющая одну огневую точку, направленную в одну сторону, и унаследованный от него *BP_building* (от английского «здание») – более большое здание с четырьмя огневыми точками, направленными в разные стороны. Они нужны для работы механики с закреплением пехоты в постройке.

Требований к визуальному стилю прототипа не предъявлялось. Все модели реализованы в упрощённом виде с помощью примитивной графики:

кубов, сфер и т.п. Юниты разных команд окрашиваются в разные цвета для их опознавания: в синий и красный. Также для отслеживания информации в реальном времени в игре реализован интерфейс, показывающий сколько отделений в каждой команде находится на поле боя, сколько юнитов противника уничтожено ими, сколько потеряно собственных, какое соотношение между ними.

Изначально созданный прототип в программе Unreal Engine 5 не имел удобного управления. Хотя эти игровые механики не относятся к работе генетического алгоритма напрямую, но из-за этого модель было невозможно протестировать на реальных людях. Поэтому перед экспериментом в первую очередь было решено разработать ряд механик, повышающих удобство использования прототипа. При их создании следовало ориентироваться на многие известные варгеймы.

В первую очередь было доработано перемещение по сцене. Во всех имеющихся стратегиях управление камерой осуществляется с помощью компьютерной мыши – если навести курсор близко к краю экрана, то камера станет передвигается в его сторону. Пользователи привыкли к этому и ожидают от стратегий именно такого управления. Подобная механика была разработана и внедрена в проект.

В стратегии «Warcraft: Orcs & Humans» от компании «Blizzard Entertainment», вышедшей в 1994 году, впервые появилась возможность выделять несколько юнитов, обводя их рамкой [38]. Сейчас эта механика используется во всех стратегиях. Данная механика также была внедрена в прототип. Ранее присутствовавшая в проекте возможность выделять юниты обеих команд была ограничена, чтобы игрок мог отдавать распоряжения только своей команде. Для понимания того, кто выбран игроком в данный момент, была добавлена подсветка выделенных юнитов и зданий (про здания см. далее).

Курсор мыши меняет свой внешний вид в следующих ситуациях:

- При наведении на противника.

- При наведении на здание, если выделено хотя бы одно отделение пехоты.

Данное нововведение также способствует повышению удобства управления и иммерсивности (эффекту погружения в игру).

Для прототипа было разработано две локации. Одна карта представляет собой открытую равнину с лесистой местностью в некоторых местах, которую можно использовать для маскировки. Также она имеет несколько небольших линий укреплений на точках появления юнитов и посередине карты, дающих преимущества игроку, если он успеет занять её первым. В качестве референса для уровня была взята карта из демо-версии проекта «Передний край», локация была воссоздана в упрощённом виде. Вторая локация представляет собой город и характеризуется плотной застройкой. Иллюстрация внешнего вида прототипа и частей двух локаций, доступных игрокам, представлены на рисунках 3.2 и 3.3.

Рисунок 3.2 – Одна из карт с равнинным ландшафтом

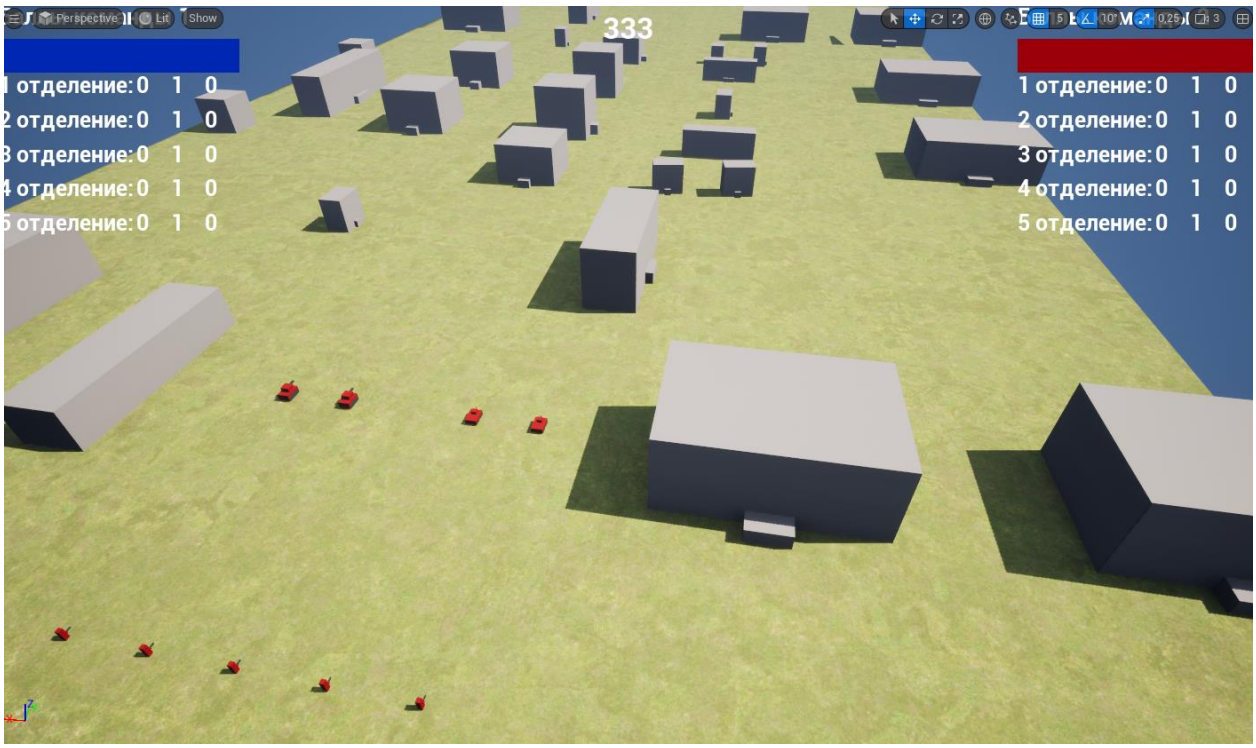


Рисунок 3.3 – Городская карта

3.2 Реализация генетического алгоритма

Работа генетического алгоритма заключается в следующем. Сначала для формирования отделения выбираются юниты и им задаётся некий боевой порядок. На поле боя может находится несколько таких отрядов. Каждое отделение имеет свою целевую функцию.

После сражения алгоритм определяет какие отделения лучше всего показали себя в бою. Их боевые построения используются для генерации новых в следующих боях с помощью мутаций и скрещиваний. Этот процесс может повторяться несколько боёв. В конечном итоге получается боевое построение, наиболее приспособленное к той обстановке, которая имелась на поле боя в прошедших сражениях.

Как уже было сказано выше, большая часть алгоритма находится в Game Instance, так как нужна возможность сохранять часть построений и переносить их в следующее сражение.

Используемые поля:

- width – ширина построения.

- depth – глубина построения отряда, то есть количество эшелонов.
- size_army – число отделений в команде для каждого из которых будет формироваться своё построение.
- squad[size_army][width][depth] – трехмерный массив, в котором записан какой тип техники находится в том или ином отделении в том или ином месте. Именно эта матрица будет оптимизироваться.
- squad2[size_army][width][depth] – трехмерный массив, содержащий информацию о построении второй команды. Этот массив остаётся неизменным на протяжении всех итераций. Используется, чтобы не расставлять юнитов второй команды вручную.
- squad_start[width][depth] – отвечает за расположение юнитов в строю в первом бою.
- number_battle – номер итерации, для работы самого алгоритма не требуется, но используется в дальнейшем для сбора статистики при сохранении данных в файл.
- top_number[2] – массив с номерами наиболее эффективных подразделений на данной итерации. Установив количество элементов массива, можно установить сколько отделений будут отобраны для следующей итерации цикла.
- save_squad[2][width][depth] – отдельный массив, записывающий наиболее эффективное расположение юнитов из прошлого боя для отобранных отрядов.
- i_tip – переменная, куда записывается тип перемещаемого юнита во время мутации.
- distance_unit[size_matrix] – расстояние между двумя соответствующими юнитами из двух отрядов. Используются как промежуточный результат во время расчёта евклидова расстояния.

- distance – евклидово расстояние между двумя построениями. То есть характеристика, показывающая насколько они отличаются друг от друга. Используется при скрещивании.

При начале боя запускается функция *f_start*. В ней вручную задаются классы юнитов для обоих команд и формируются их боевые построения для первого боя.

Отделения, находящиеся на игровой сцене, в начале раунда считывают данные из Game Instance с помощью функций *f_get_squad* для команды, строй которой оптимизируется, и *f_get_squad2* – для второй команды, он останется неизменным на протяжении всех итераций. Акторы получают к ним доступ, вызывая их в блюпринтах с помощью ноды – небольшой панели, содержащей данные и программные операции.

После боя наиболее эффективные боевые порядки записываются с помощью функции *f_set_top_squad*. Эта функция также вызывается из блюпринтов с помощью ноды.

Когда начинается новое сражение, то загружаются боевые порядки отделений, показавших себя ранее наиболее эффективно. Затем они меняются с помощью функции *f_mutation*.

Функция *f_distance* получает на входе номера лучших отделений и рассчитывает Евклидово расстояние между ними, то есть показатель, характеризующий насколько отделения похожи друг на друга. Используется при скрещивании [33]. Аргументация выбора именно Евклидова расстояния для этих целей была дана во второй главе.

Функция *f_Excel* вызывается из BP_camera в конце боя. В неё передаются результаты целевые функции всех отделений. Они сохраняются в два файла типа csv для последующего сбора статистики о работе алгоритма и создания графиков. В оба файла выводится одна и таже информация, но в разной структуре. В одном результаты команд записываются по порядку в один столбик. В другом – данные каждого отделения распределяются по своим колонкам. При неправильной записи данных и последующей работе с ними

может возникнуть ряд проблем. Особенности, которые необходимо учитывать при работе с данным файлом, были расписаны в предыдущей главе.

3.3 Заключение по главе

В третьей главе был реализован прототип стратегии со встроенным в него генетическим алгоритмом. Были разработаны несколько локаций для тестирования алгоритма. Реализованные механики:

- Генетический алгоритм в классе GameInstance.
- Юниты, их искусственный интеллект, актор, отвечающий за соблюдение боевого порядка юнитов.
- Блюпринты для передачи информации о типе местности.
- Здания.
- Управление мышью.
- Интерфейс.

Можно сказать, что модель состоит из двух подсистем:

1. Одна подсистема является непосредственно генетическим алгоритмом, который оптимизирует строй юнитов. Работает она один кадр при старте боя и в конце. Выбирая отряды, показавшие себя наиболее хорошо, и делая на их основе новые построения.
2. Вторая подсистема работает всё время и указывает юнитам в отделении, где они должны находиться на данный момент. Она реализована через создание отдельного актора для каждого отделения.

Также в главе представлен список переменных и функций, используемых в реализованном генетическом алгоритме, дано описание их функционала.

4. ОЦЕНКА ЭФФЕКТИВНОСТИ РАБОТЫ МОДЕЛИ

4.1 Формулировка требований к оценке модели

Для оценки работы реализованной модели необходимо проведение нескольких тестов с разными входными данными на разных типах ландшафта. Проверка работы модели разделена на два этапа. На первом этапе для предварительного тестирования была проведена серия автоматических боёв с участием искусственного интеллекта. Причём требуется проверить, что целевая функция не увеличивается сама по себе без использования модели. Для этого нужно провести серию боёв с отключенным генетическим алгоритмом и собрать статистику о том, как она меняется. Если она улучшается только с включенным генетическим алгоритмом, значит предложенная модель работает.

После доработки прототипа, в основном направленных на повышение удобства игры, был проведён второй этап – эксперимент проходил с использованием реальных игроков, сражающихся против искусственного интеллекта. В бою может быть несколько разных отрядов одновременно. Если разместить их на карте сразу, то игрок не сможет эффективно отдавать им приказы. Проведя аналогию со сферой менеджмента это можно назвать нормой управляемости. Поэтому было решено использовать механику подкреплений, активно применяемую в варгеймах (таких как «Передний край» и «Steel Division 2»). Её суть в том, что игрок сразу получает под контроль только часть армии и периодически может вызывать новых юнитов. В прототипе это реализовано таким образом, что отделения появляются не сразу, а только по истечению заданного таймера.

Перед экспериментом стоял вопрос: какое построение выбрать изначально для оптимизации? Так как исследование посвящено в первую очередь военным симуляторам, то за основу для стартового боевого порядка юнитов в игре было решено взять варианты используемых в реальности боевых построений, описанных в таких книгах как «Общая тактика» В.Н.

Зарицкого и Л.А. Харкевича [39], «Тактика – искусство боя» И.Н. Воробьёва [40], «Учебник сержанта танковых войск» [41] и некоторых других источниках Министерства Обороны РФ[42]. Далее было проверено можно ли улучшить их в рамках игры с учётом её специфики.

4.2 Автоматизированное тестирование модели

Описание первого эксперимента. Первая команда в наступлении – её строй оптимизируется. Вторая – в обороне. В начале боя в переднем эшелоне у обеих команд располагаются танки, следом за ними идёт лёгкая бронетехника и автомобили. При нескольких тестах с этими исходными данными работа алгоритма приводила к одному из двух вариантов. В одном случае машины и лёгкая бронетехника оказывались рядом с танками, как бы действуя с ними в двойках и обнаруживая цели раньше за счёт лучшего обзора. В другом – вся группа располагалась более плотно и за счёт этого и лучше обнаруживала противника, и быстрее под огнём достигала его позиций. Оба варианта приводили к улучшению целевой функции и повышению выживаемости группы. График, показывающий изменения целевой функции в одном из тестов, изображён на рисунке 4.1.

Рисунок 4.1 – Изменение средней целевой функции в первом эксперименте

В ряде данных есть пара выбросов. Они объясняются случайными событиями на поле боя: например, один и тот же юнит в одной итерации может попасть в противника с первого раза, а в другой – промахнуться. Причём даже во время выбросов вниз значение целевой функции выше, чем было на первой итерации. Уже на второй и третьих итерациях значение целевой функции значительно выше. При большом количестве итераций значение выходит на плато. Выбросы обусловлены особенностями применения генетических алгоритмов и тем, что во время первого эксперимента ещё не было реализовано условие выхода из алгоритма. Это приводило к тому, что, найдя некий оптимум, алгоритм пытался подобрать новый строй, что приводило к выходу из точки экстремума. В данном опыте на каждой итерации значение целевой функции увеличивалось в среднем на 16,46%.

Описание второго эксперимента. Моделируется засадная тактика. Первая команда в наступлении. Состоит из танков, лёгкой бронетехники и стрелков. Юниты развёрнуты в единую линию. Вторая команда в обороне в лесистой местности. Состоит из пехоты. Изначально при контакте с противником техника из первой группы достаточно быстро уничтожалась. В результате работы алгоритма через несколько итераций часть техники была перемещена во вторую линию и следовала не наравне с пехотой, а следом. Так она смогла атаковать противников, которых обнаружила пехота с безопасного расстояния. За счёт этого выживаемость группы увеличилась. Результат того, как изменялось среднее значение функции приспособленности у отделений на разных итерациях представлен на рисунке 4.2. Функция приспособленности в среднем на каждой итерации увеличивалась на 12,59%.

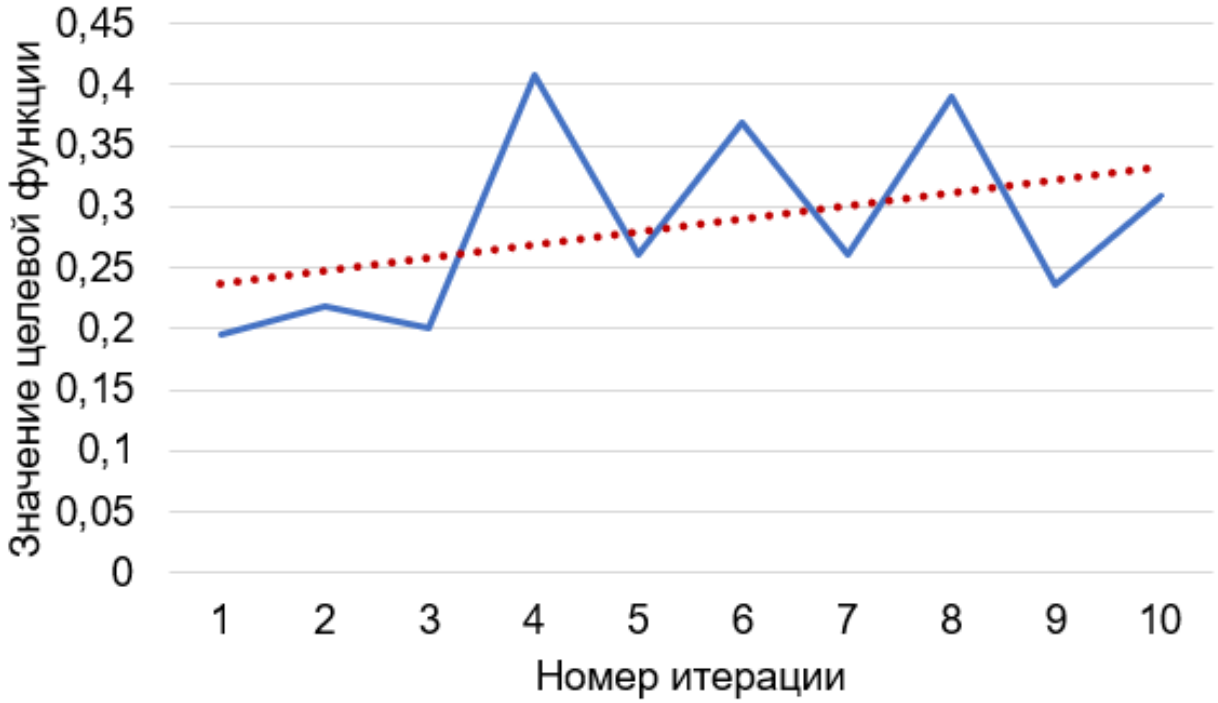


Рисунок 4.2 – Изменение средней целевой функции отделений в одном из экспериментов

Описание третьего эксперимента. Городская местность. Первая команда в наступлении. Состоит из танков и пехоты. Юниты были расположены так, чтобы идти в две линии по разным сторонам улицы. Вторая команда в обороне и состояла из разных типов юнитов, рассредоточенных по городу. В данном случае улучшить значение целевой функции не удалось. Её динамика представлена на рисунке 4.3. Это говорит не о том, что алгоритм не работает, а о том, что он применим в одних ситуациях, а не применим в других. В предыдущих опытах в иных ситуациях он показывал результат.

Рисунок 4.3 – Изменение функции приспособленности в третьем эксперименте

Чтобы убедиться, что на функцию приспособленности ранее влиял именно генетический алгоритм, отключим его и посмотрим будет ли меняться целевая функция без него. Для этого была проведена серия боёв в автоматическом режиме. Результат одного из тестов представлен на рисунке 4.4. В среднем изменение целевой функции стремится к нулю. Значит её изменение в прошедших экспериментах действительно можно объяснить работой генетического алгоритма. Однако было обнаружено, что даже при неизменяющихся входных данных периодически случаются выбросы. Как уже было сказано выше они объясняются тем, что в игре присутствует элемент случайности. Выбросы в данных – естественное явление [43].

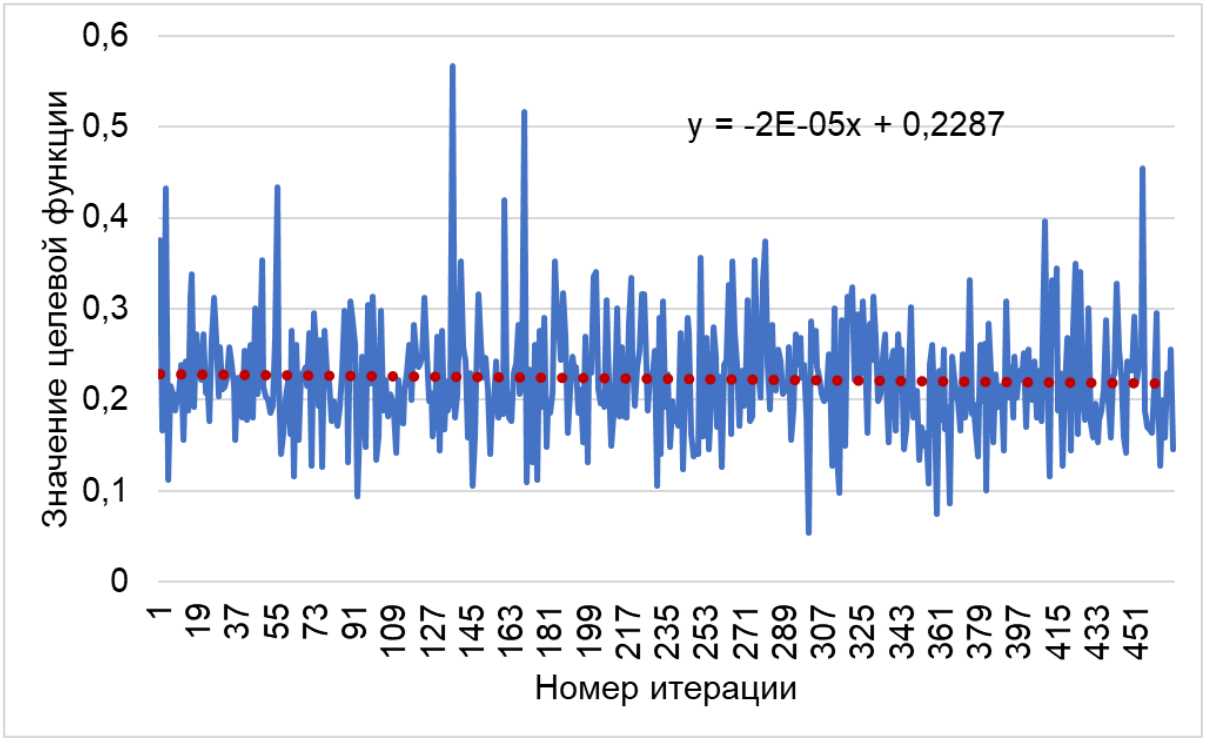


Рисунок 4.4 – Изменение целевой функции без использования разработанной модели

Примечательно то, что при одних и тех же вводных построения в конечном итоге принимали почти всегда один и тот же вид. Хотя казалось бы, что оптимизировать их можно по-разному. Что свидетельствует о том, что юниты при перераспределении занимают не случайные позиции, а более эффективные. В разных играх из-за разных игровых механик и характеристик юнитов модель может приводить к разному результату.

4.3 Оценка модели в игре с реальными игроками

Для проведения эксперимента была создана группа, состоявшая из двенадцати человек, регулярно играющих в компьютерные игры хотя бы раз в месяц, возрастом от 17 до 28 лет. В выборке есть люди, работающие в сфере разработки игр на позициях геймдизайнеров, программистов, дизайнеров и художников. За раз они играли от пяти боёв за раз и больше, чтобы оценить работу алгоритма. Отделения состояли из всех четырёх реализованных в прототипе основных типов юнитов: пехоты, автомобилей, БМП и танков. В первой линии отделения изначально была поставлена бронетехника, за ней следовала пехота и автомобили.

Во время игры на открытой локации с полем у 10 из 12 человек существенно изменилось значение целевой функции. У разных пользователей оно увеличивалось за итерацию от 6,1% до 20,6%. Результат изменения целевой функции в одном из экспериментов на открытой карте с полем представлен на рисунке 4.5. Красным цветом на графике выделена линия тренда. На каждой из 10 итераций среднее значение целевых функций увеличивалось в среднем на 0,06 единицы, что составляет 10,8%. Оптимизация достигалось у разных игроков разными методами: у одних вперёд выносились юниты с лучшим обзором, чем у танков, у других - танки отводились назад ближе к другой технике, что обеспечивало их лучшее прикрытие, у третьих – часть танков убиралась из первой линии так, что роль разведки оставался выполнять танк. Это повышает вариативность игры и создаёт у разных пользователей разные игровые ситуации. У одного пользователя возникла уникальная ситуация. После того, как он несколько раз использовал обхват с фланга, противник повернул строй боком и разместил с этого фланга автомобиль для разведки.

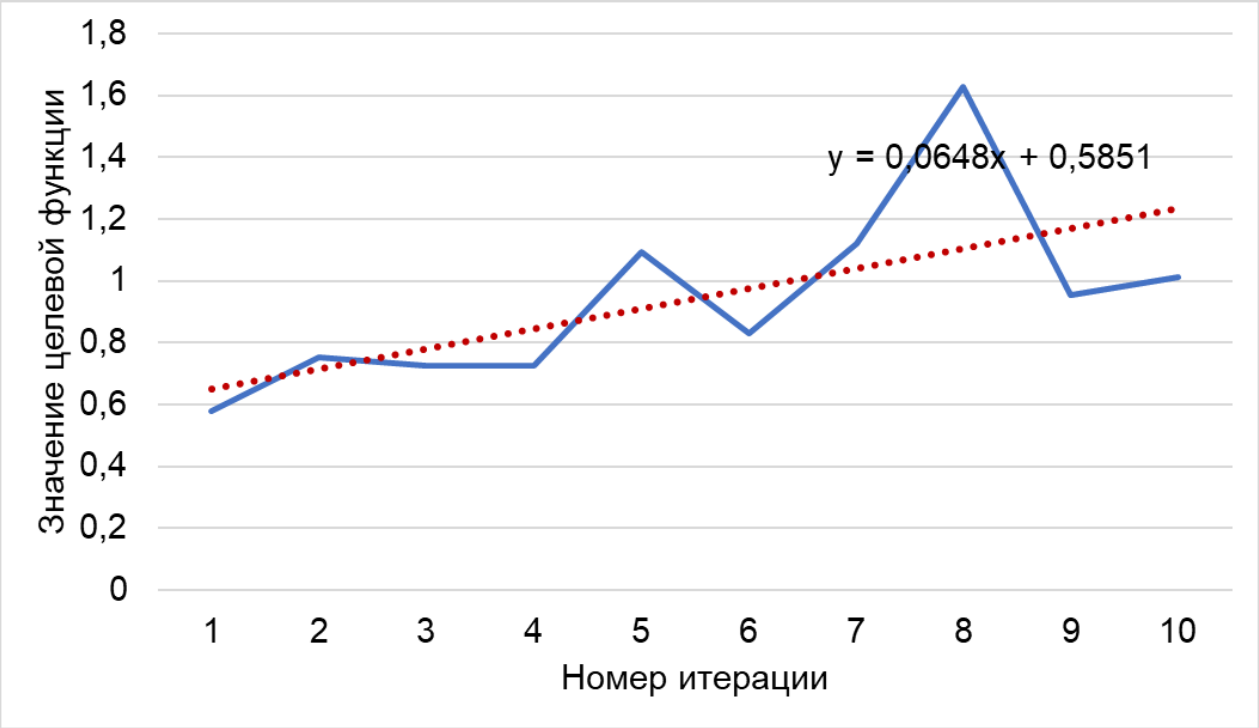


Рисунок 4.5 – изменение значений средней целевой функции

У нескольких людей функция приспособленности почти не изменялась. Это можно объяснить тем, что генетический алгоритм основывается на случайном переборе и не всегда удаётся быстро подобрать нужную комбинацию построения.

В городе значение целевой функции в ходе эксперимента повысилось у половины людей. Рост составлял от 2% до 18,9% за итерацию. У двух людей незначительно уменьшилось, у остальных четырёх осталось почти неизменным. В городе юниты в отделении группировались в более маленькие подгруппы, чтобы находиться ближе друг к другу, что положительно сказывалось на их функции приспособленности. Но на это требовалось больше итераций, чем при тестах на открытой карте. Результат одного из таких тестов представлен на рисунке 4.6, в нём среднее значение целевой функции увеличивалось на каждой итерации на 0,07 единиц, что составляло 18,9%. Примечательно то, что строй на двух разных картах на равнине и в городе в конечном итоге принимал разные построения, что говорит о том, что модель может приспосабливаться под разные ландшафты.

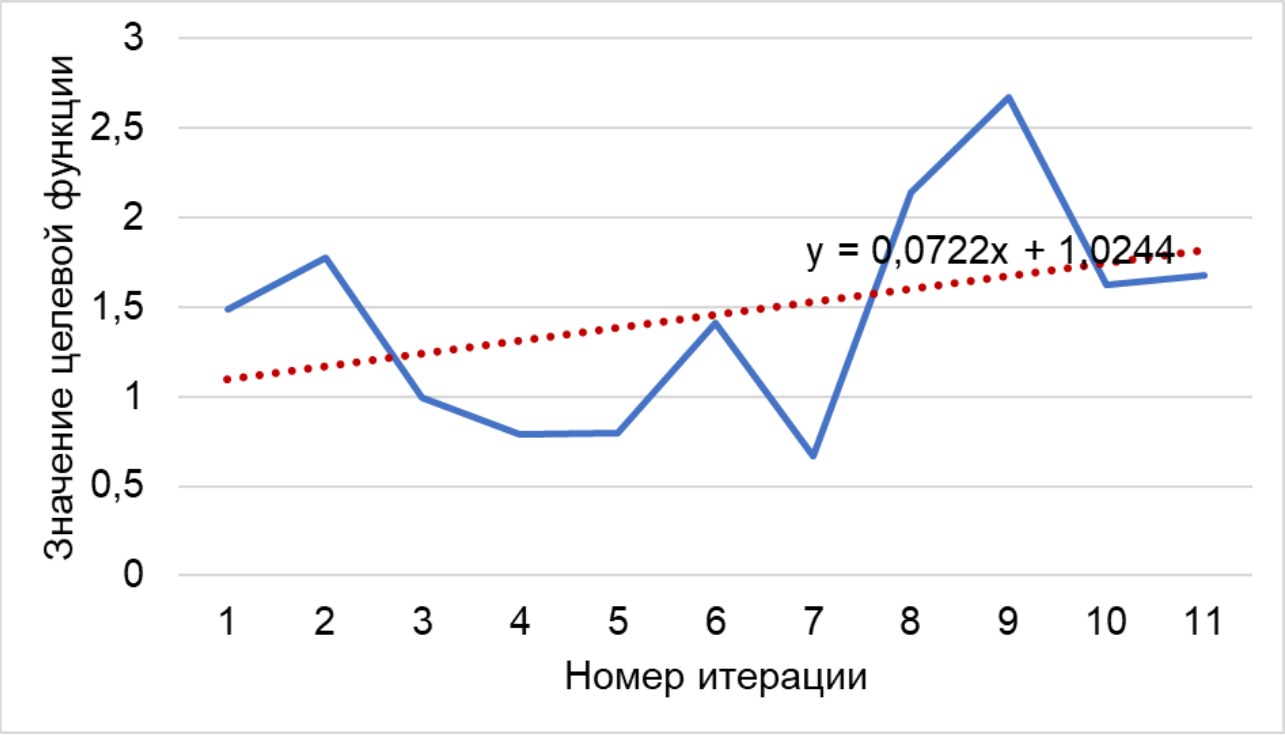


Рисунок 4.6 – Изменение значений средней целевой функции

4.4 Заключение по главе

Таким образом, опытным путём установлено, что предложенная модель действительно повышает эффективность игры искусственного интеллекта. К плюсам использования генетического алгоритма для данных целей можно отнести следующие пункты:

1. Относительная простота реализации.
2. У разных пользователей генерируются разные построения, что создаёт более уникальный игровой опыт.
3. Модель достаточно универсальна и может работать в разных игровых ситуациях без доработок алгоритма.

Минусы модели следующие:

1. Для оптимизации построения требуется время. Алгоритм делает выводы только по результатам прошедшего боя, но не реагирует на происходящую ситуацию в моменте. Поэтому эффективно использовать его возможно только в играх с длительной кампанией, состоящей хотя бы из нескольких миссий, таких как «Steel Division».
2. Иногда попытка оптимизировать боевое построение может приводить к ухудшению боеспособности. Этого можно избежать, если создать механизм, останавливающий работу алгоритма в ситуациях, когда ему не удаётся повысить целевую функцию.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана модель, оптимизирующая боевой порядок юнитов в компьютерных играх жанра стратегия под окружающую среду и предпочитаемую тактику игрока. Работа модели основана на генетическом алгоритме – оптимизация происходит методом перебора с выбором наиболее приспособленных особей, их скрещиваний и мутаций.

Также предусмотрена работа алгоритма в ряде ситуаций, возникающих в компьютерных играх. Предусмотрены несколько условий выхода из алгоритма в обычной ситуации и в случаях, когда оптимизация не удаётся. Условия выхода заданы такими, чтобы минимизировать влияние на них выбросов, вызванных случайным повышением значения функции приспособленности. Предусмотрена возможность переноса результатов оптимизации строя с одного отделения на другое при изменении состава армии.

Для апробирования работы подобного алгоритма был разработан прототип стратегии на игровом движке Unreal Engine 5, в котором в упрощённом виде реализованы ключевые игровые механики, важные для варгеймов. Модель в нём была реализована как отдельная игровая механика. Для тестирования алгоритма было реализовано несколько локаций с разным преобладающим типом местности.

Модель состоит из двух подсистем. Одна работает всё время и указывает юнитам, где они должны находится в тот или иной момент времени. За счёт этого войска удерживают строй. Вторая подсистема отвечает непосредственно за генетический алгоритм. Она срабатывает по одному разу при старте боя и в конце и собирает информацию о наиболее эффективных отрядах, отвечает за их отбор, мутации и скрещивания.

Для оценки работы алгоритма были проведены два этапа экспериментов и сформулированы критерии оценки их результатов. В первом тесте

проводились серии автоматических боёв двух команд, обе из которых управлялись искусственным интеллектом. На втором этапе тестирование проводилось с привлечением реальных людей, играющих против искусственного интеллекта. В ходе экспериментов было установлено, что предлагаемая модель действительно может исправно выполнять свою задачу. Причём модель подбирала разный строй для разных карт с разными преобладающими типами местности. Модель имеет ряд недочётов: она даёт положительный результат не всегда и не всегда приводит к изменениям у некоторых игроков, но с её использованием эффективность действий противника в среднем увеличивалась. Значение, на которое увеличивается целевая функция за бой, может варьироваться у разных игроков, также как и число необходимых итераций. Это объясняется их разным игровым опытом и используемыми тактиками. Таким образом эффективность игры искусственного интеллекта с используемой моделью действительно повышается. Заявленные задачи работы были полностью выполнены.

Разработанное решение в виде проекта Unreal Engine 5.2 опубликовано в репозитории сервиса GitHub и доступно по следующей ссылке: https://github.com/YaK-571/Genetic_Algorithm_Wargame.git

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Загарских А.С., Хорошавин А.А., Александров Э.Э., Введение в разработку компьютерных игр– СПб: Университет ИТМО, 2019. – 79 с.
2. По образу и подобию: почему ИИ в играх «читерит» [Электронный ресурс]. – 2020. – URL: https://dtf.ru/gamedev/139091-po-obrazu-i-podobiyu-pochemu-ii-v-igrah-chiterit (дата обращения 15.02.2024)
3. Не совсем человек: искусственный интеллект в играх [Электронный ресурс]. – 2021. – URL: https://skillbox.ru/media/gamedev/iskusstvennyy-intellekt-v-igrakh/ (дата обращения 15.02.2024)
4. Седых И. А. – Индустрия компьютерных игр 2020 [Электронный ресурс]. – 2020. – URL: https://dcenter.hse.ru/data/2020/07/27/1599127653/Индустрия%20компьютерных%20игр-2020.pdf (дата обращения 06.01.2023).
5. Итоги 2020 года: рынок игровой индустрии [Электронный ресурс]. – 2021. – URL: https://hsbi.hse.ru/articles/itogi-2020-goda-rynok-igrovoy-industrii/ (дата обращения 06.01.2023).
6. Анализ рынка игр в России: кризисные полгода 2020 vs стабильные полгода 2019 [Электронный ресурс]. – 2020. – URL: https://dtf.ru/gameindustry/247402-analiz-rynka-igr-v-rossii-krizisnye-polgoda-2020-vs-stabilnye-polgoda-2019 (дата обращения 06.01.2023)
7. Индустрия видеоигр [Электронный ресурс]. – 2020. – URL: https://www.tinkoff.ru/invest/research/strategy/2021-videogames/ (дата обращения 06.01.2023)
8. Куда движется жанр 4X-стратегий [Электронный ресурс]. – 2021. – URL: https://dtf.ru/u/943-konstantin-sakhnov/909129-kuda-dvizhetsya-zhanr-4h-strategiy (дата обращения 06.01.2023).

9. Савченко, А. Игра как бизнес. От мечты до релиза / Савченко Алексей – М.: Бомбора, 2022 – 336 с.
10. GamesIndustry.biz presents… The Year In Numbers 2022 [Электронный ресурс]. – 2022. – URL: https://www.gamesindustry.biz/gamesindustrybiz-presents-the-year-in-numbers-2022 (дата обращения 06.01.2023).
11. Гейминг в России. Социальные и экономические эффекты – М.: НАФИ – 2022 – 68 с.
12. Стратегикон. База стратегий [Электронный ресурс]. – 2022. – URL: https://strategycon.ru/database/ (дата обращения 01.05.2023).
13. Фрейермут, Г. Игры. Геймдизайн. Исследование игр / Фрейермут Гундольф – Харьков: Гуманитарный центр Харьков, 2021 – 250 с.
14. Что есть стратегия реального времени (RTS) и как с ней быть? [Электронный ресурс]. – 2022. – URL: https://dtf.ru/games/1422003-chto-est-strategiya-realnogo-vremeni-rts-i-kak-s-ney-byt (дата обращения 01.05.2023).
15. Классификация жанров стратегий [Электронный ресурс]. – 2022. – URL: https://strategycon.ru/strategy-video-game-classification/ (дата обращения 01.05.2023).
16. Мейер, С. Сид Мейер: Жизнь в мире компьютерных игр / Сид Мейер – М.: Манн, Иванов и Фербер, 2021 – 304 с.
17. Саймон, Д. Алгоритмы эволюционной оптимизации / Саймон Дэн – М.: ДМК Пресс, 2020 – 940 с.
18. Грегори, Д. Игровой движок. Программирование и внутреннее устройство / Грегори Джейсон – Спб.: Питер - 2022 г. – 1136 с.
19. Гундольф, Ф. Игры. Геймдизайн. Исследование игр / Фрейермут Гундольф – Гуманитарный центр, 2021 г. – 250 с.
20. Сирия: Русская буря. Обзор игры. [Электронный ресурс]. – 2017. – URL: https://media.vkplay.ru/articles/review/sirija_russkaja_burja/ (дата обращения 21.04.2024)

21. Доусон, М. Изучаем C++ через программирование игр. – Спб.: Питер – 2022. – 352 с.
22. Мартин, Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. – Спб.: Питер, 2022 – 464 с.
23. Костер Р. Разработка игр и теория развлечений. – М.: ДМК Пресс, 2018 г. – 270 с.
24. Протодьяконов, А.В. Математические и программные методы построения моделей глубокого обучения. Учебное пособие / А.В. Протодьяконов, А.В. Пылов, П.А. Дягилева – Вологда: Инфра-Инженерия, 2023 г. – 176 с.
25. Как сбалансировать стратегию: опыт разработчика Warcraft III [Электронный ресурс]. – 2018. – URL: https://dtf.ru/gamedev/32811-kak-sbalansirovat-strategiyu-opyt-razrabotchika-warcraft-iii (дата обращения 15.05.2023).
26. Новак, Д. О. Искусственный интеллект как инструмент создания компьютерных игр = Artificial intelligence as a tool for creating computer games / Новак Д. О., Попова Е. С., Василькова А. Н. // Электронные системы и технологии : сборник материалов 59-й научной конференции аспирантов, магистрантов и студентов БГУИР, Минск, 17–21 апреля 2023 г. / Белорусский государственный университет информатики и радиоэлектроники ; редкол.: Д. В. Лихаческий [и др.]. – Минск, 2023. – С. 793-795.
27. Лоспинозо, Д. C++ для профи. – СПб.: Питер, 2021. – 816 с.
28. Девятков, В.В. Системы искусственного интеллекта. / В.В. Девятков – Издательство МГТУ им. Н.Э.Баумана, 2023 г. – 280 с.
29. Куксон, А. Разработка игр на Unreal Engine 4 за 24 часа / Арам Куксон, Райан Даулингсока, Клинтон Крамплер, пер с англ М.А. Райтмана – М.: Эскмо, 2019 г. – 528 с.

30. Кугаевских, А.В. Классические методы машинного обучения / А.В. Кугаевских, Д.И. Муромцев, О.В. Кирсанова. – СПб: Университет ИТМО, 2022. – 53 с.
31. Дэн, С. Алгоритмы эволюционной оптимизации / Дэн Саймон пер. с англ. А. В. Логунова. - М.: ДМК Пресс, 2020. - 940 с.
32. Основы машинного обучения: учебное пособие / О.В. Лимановская, Т.И. Алферьева; Мин-во науки и высш. образования РФ. – Екатеринбург: Издательство Урал. университета, 2020. – 88 с.
33. Хлопцев, А. А. Сравнение эффективности различных операторов выбора родительской пары в генетическом алгоритме с дискретной рекомбинацией / А. А. Хлопцев // Электронные системы и технологии: сборник материалов 58-й научной конференции аспирантов, магистрантов и студентов БГУИР, Минск, 18-22 апреля 2022 г. / Белорусский государственный университет информатики и радиоэлектроники ; редкол.: Д. В. Лихаческий [и др.]. – Минск, 2022 г. – С. 39–42
34. Steel Division II: Обзор [Электронный ресурс]. – 2019. – URL: https://strategycon.ru/steel-division-2-review/ (дата обращения 01.02.2024)
35. Анфёров М.А. Генетический алгоритм кластеризации / М.А. Анфёров // Russian Technological Journal. 2019;7(6):134-150.
36. Александер, М. Excel 2019. Библия пользователя. Исчерпывающее руководство, М.: Вильямс, 2019. – 1136 с.
37. Бабешко, Л. Эконометрика и эконометрическое моделирование в Excel и R. Учебник, М.: Инфра-М, 2020. – 300 с.
38. История жанра RTS, часть 3. Как Warcraft спасла Blizzard [Электронный ресурс]. – 2021. – URL: https://www.ixbt.com/live/games/istoriya-zhanra-rts-chast-3-kak-kovalsya-warcraft.html (дата обращения 21.04.2024)

39. Зарицкий, В.Н. Общая тактика: учебное пособие / В.Н. Зарицкий, Л.А. Харкевич. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2007. – 184 с.
40. Воробьёв, И.Н. Тактика – искусство боя, М.: Общевойсковая академия ВС РФ, 2002. – 889 с.
41. Учебник сержанта танковых войск / Г.П. Волотова, О.В. Норунова, И.Г. Подкопаева, А.В. Игнатова, П.М. Бодуна, Б.Н. Москвина, А.С. Масленникова, С.П. Кочешева, В.И. Мясникова, под ред. А.В. Квашнина и А.И. Скородумова, М.: Военное издательство Министерства обороны Российской Федерации, 2004. – 480 с.
42. Маркин, А.В. Альбом тактических схем для самостоятельной подготовки в малых и средних группах – М.: Фонд "КУОС-Вымпел". Центр стратегической конъюктуры, 2022. – 92 с.
43. Тюрин, Ю. Анализ данных на компьютере. Учебное пособие. – М.: Издательство МЦНМО, 2020. – 368 с.