Федеральное государственное автономное образовательное
учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедры ИС

_____ П.П. Дьячук
подпись инициалы, фамилия

«____» _____2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – «Информационные системы и технологии»

Разработка игрового искусственного интеллекта на базе ПО компьютерных игр Unreal Engine 4

Руководитель _____ доцент, к.т.н. И. А. Легалов
подпись, дата инициалы, фамилия

Выпускник _____ Д. О. Сучков
подпись, дата инициалы, фамилия

Нормоконтролер _____ ст. препод. ИС Ю. В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедры ИС

_____ П.П. Дьячук
подпись инициалы, фамилия

«____» _____2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Сучкову Дмитрию Олеговичу

Группа: КИ16-13Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: «Разработка игрового искусственного интеллекта на базе ПО компьютерных игр Unreal Engine 4»

Утверждена приказом по университету № 6499/с от 22.05.2020 г.

Руководитель ВКР: И.А. Легалов, к.т.н., доцент кафедры «Информационные системы» ИКИТ СФУ.

Исходные данные для ВКР: Требования к разрабатываемой системе, методические указания научного руководителя, учебные пособия.

Перечень разделов для ВКР: Введение, теоретическая часть, реализация прототипов игрового искусственного интеллекта, заключение, список использованных источников.

Перечень графического материала: Презентация, выполненная в Microsoft Office PowerPoint 2016.

Руководитель ВКР _____ И. А. Легалов
подпись, дата — инициалы, фамилия

Задание принял к исполнению _____ Д. О. Сучков
подпись, дата — инициалы, фамилия

«____» _____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка игрового искусственного интеллекта на базе ПО компьютерных игр Unreal Engine 4» содержит 54 страницы текстового документа, 34 иллюстраций, 3 приложения, 11 использованных источников.

ИСКУСТВЕНЫЙ ИНТЕЛЛЕКТ, UE4, ПРОТОТИП, EQS, ВОСПРИЯТИЕ ИИ, ДЕРЕВО ПОВЕДЕНИЙ, NPC, MOB, BOT

Актуальность.

Наличие игрового искусственного интеллекта в играх не во всех нужно, но важность при надобности их разработать в игре большая от нее может зависеть чуть ли не весь смысл игры красивая и грамотная подача может привести к увеличению игроков в игре. Есть игры, где взаимосвязь с игровым ИИ и игроком в продолжительность всю игру, если они будут не устойчивыми, то игроку может наскучить.

Объектом исследования является система игрового искусственного интеллекта.

Предметом исследования являются программное обеспечение Unreal Engine 4.

Целью выпускной квалификационной работы является разработка набора прототипов игрового искусственного интеллекта. Разрабатываться будет на про программное обеспечение компьютерных игр «Unreal Engine 4». Разработка должна помочь в дальнейшем для использования в личном проекте.

Задачи:

- изучить концепцию игрового искусственного интеллекта;
- выделить виды игрового искусственного интеллекта;
- рассмотреть примеры создания игрового искусственного интеллекта;

− подготовить игровые ресурсы, которые будут использоваться в разработке;

− создать прототипы игрового искусственного интеллекта.

Практическая значимость в том, чтобы использовать данные прототипы в личных проектах. Научившись на ошибках больше их не повторять. Так как реализация прототипов, то все наработки будут выложены в интернет для общего пользования.

Для достижения цели необходимо изучить предметную область. Разобрать примеры реализаций другие разработчиков. Рассмотреть инструментарий программного обеспечения компьютерных игр «Unreal Engine 4». Подготовить игровые ресурсы, которые будут способствовать успешной разработке.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С появлением искусственного интеллекта (ИИ) произошло много изменений в развитии человечества. Он важен для развития отраслей науки такие как: робототехника, кибернетика, медицина и для более быстрого, удобного доступа к мировым информационным ресурсам. А также искусственный интеллект необходим для развития социальных услуг и для ведения кибервойн и в проведение досуга в компьютерных играх.

Алгоритмы искусственного интеллекта можно применять для решения практически любых задач. Например, они могут использоваться для управления производственной линией в промышленности или для распознавания образов в медицине.

Пока в одном месте искусственный интеллект развивается с большой скоростью, в другой части зарождается другое направление в данной научной сфере, а точнее ее ответвление.

Игровой искусственный интеллект иллюзия на наличии интеллекта в поведении персонажей, управляемых компьютером. Разница игрового ИИ и традиционного ИИ, в том, что игровой ИИ подразумевает наличие ИИ или же отсутствие, но тогда он создает иллюзию его существование. Наличие его в игре не всегда нужна, все зависит от того какую игру собираются делать.

Для реализации игрового искусственного интеллекта будет использоваться программное обеспечение компьютерных игр “Unreal Engine 4”.

Целью выпускной квалификационной работы является разработка набора прототипов игрового искусственного интеллекта. Разрабатываться будет на про программное обеспечение компьютерных игр “Unreal Engine 4”.

Задачи:

- изучить концепцию игрового искусственного интеллекта;
- выделить виды игрового искусственного интеллекта;

– рассмотреть примеры создания игрового искусственного интеллекта;

– подготовить игровые ресурсы, которые будут использоваться в разработке;

– создать прототипы игрового искусственного интеллекта.

Практическая значимость в том, чтобы использовать данные прототипы в личных проектах и выложить их в среду интернет для общего пользования.

1 Анализ предметной области

1.1 Понятие игровой искусственный интеллект

Игровой искусственный интеллект (игровой ИИ) — набор алгоритмов, программных методик для реализации игровых персонажей (или ресурсов) с иллюзией на наличие интеллекта. Иллюзия, почему? Среда, для которой разрабатывается игровой ИИ не является реальной, поэтому не подразумевает использование всегда сложных алгоритмов и подходов традиционного ИИ, так как в играх он не всегда играет первую роль. Игровому ИИ достаточно обмануть человека на наличие интеллекта [1].

Важность развитие в игровой области не меньше, чем в научной сфере. Он также включает себя алгоритмы традиционного ИИ, теории управления, робототехники, компьютерной графики и информатики в целом. Изучение игрового и традиционного ИИ могут друг другу помочь, что развитие обеих областей важно.

Многие эксперты жалуются на то, что «ИИ» в термине «игровой ИИ» преувеличивает свою ценность, так как игровой ИИ не связан с интеллектом и разделяет несколько целей академической области ИИ. Исторически академические проекты игрового ИИ были относительно отделены от коммерческих продуктов, потому что академические подходы были простыми и не масштабируемыми. Коммерческий игровой ИИ разработал собственный набор инструментов, которых было достаточно, чтобы обеспечить хорошую производительность во многих случаях.

Растущее осознание разработчиками игр академического ИИ и растущий интерес академического сообщества к компьютерным играм делает определение того, что считать его в игре, становится менее своеобразным. Тем не менее, существенные различия между различными областями

применения ИИ означают, что игровой искусственный интеллект все еще можно рассматривать как отдельное подполе традиционного ИИ.

Основным ограничением сильного ИИ является внутренняя глубина мышления и чрезвычайная сложность процесса принятия решений. Это означает, что хотя теоретически было бы возможно сделать «умный» ИИ, проблема потребовала бы значительной вычислительной мощности [2].

1.2 Алгоритмы и методы при разработке игрового ИИ

1.2.1 Дерево поведений

Игровой ИИ используются в самых разных областях игры. Наиболее очевидным является контроль над любыми неигровыми персонажами (NPC) в игре, хотя «сценарии» (дерево поведений) в настоящее время являются наиболее распространенным средством контроля. Деревья поведений часто приводят к «искусственной глупости», такой как повторяющееся поведение, потеря погружения или ненормальное поведение в ситуациях, которые разработчики не планировали [3].

Дерево поведений состоит из узлов, в которых описывается действие и от их может пойти другое ответвление в действиях ИИ. Пример дерева поведений в программном обеспечение Unreal Engine 4:

Все начинается с узла ROOT. Зеленым отмечена служба она подключаются к узлам и будут работать с определенной частотой, пока выполняется их ветвь. Они занимают место традиционных параллельных узлов в других системах дерева поведения. Синим обозначены декораторы, также известный как условные выражения определяет, может ли быть выполнена ветвь в дереве или даже один узел. Фиолетовым отмечены задачи, где определяется что нужно делать ИИ.

1.2.2 Статической поиск пути

Поиск пути, еще одно распространенное использование ИИ, широко используется в стратегических играх в реальном времени. Поиск пути – это метод определения того, как доставить NPC из одной точки на карте в другую, принимая во внимание местность, препятствия и, возможно, «туман войны». Коммерческие видеоигры часто используют быстрое и простое «поиск пути на основе сетки», при котором местность отображается на жесткую сетку из равномерных квадратов. В некоторых играх вместо жесткой сетки используются нерегулярные многоугольники, которые собирают сетку навигации вне областей карты, к которым могут идти неигровые персонажи. В качестве третьего метода разработчикам иногда удобно вручную выбирать «путевые точки», которые NPC должны использовать для навигации; смысл в том, что такие путевые точки могут создавать неестественно выглядящие движения. Кроме того, путевые точки, как правило, работают хуже, чем навигационные сетки в сложных средах. Например, в игре Black Desert навигация строится по путевым точкам:

Рисунок 1 – Пример путевых точек в игре Black Desert

1.2.3 Динамический поиск пути

Помимо статического нахождения пути, навигация – это подполе игрового ИИ, которое фокусируется на предоставлении NPC возможности перемещаться в динамической среде, находить путь к цели, избегая столкновений с другими объектами. Навигация в динамических стратегических играх с большим количеством юнитов, таких как Age of Empires (1997) или Civilization V (2010), часто работает плохо; юниты часто мешают другим юнитам. В игре Overlord реализовали очередность отряда по движению, но таким образом нельзя было взять определенного юнита (все в режиме очереди) [4].

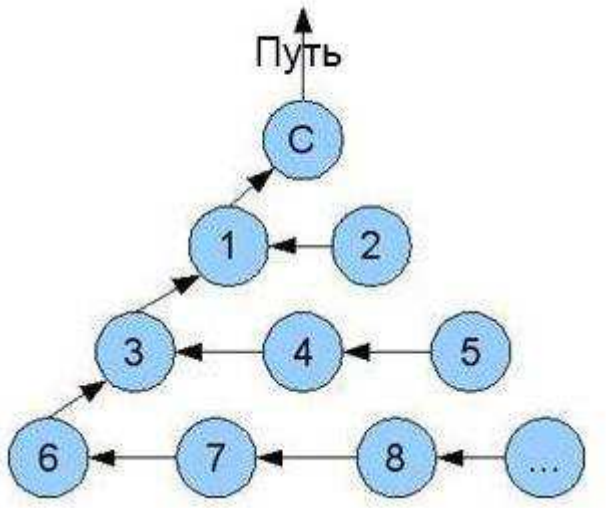


Рисунок 2 – Порядок движение треугольным строем в игре Overlord

Вместо того, чтобы улучшать игровой ИИ для правильного решения сложной проблемы в виртуальной среде, зачастую более выгодно просто изменить сценарий, чтобы он был более гибким. Если поиск пути увязает над конкретным препятствием, разработчик может просто переместить или удалить препятствие.

1.2.4 Система боя

Многие современные видеоигры подпадают под категорию боевиков, шутеров от первого лица или приключений. В большинстве игр такого типа существует определенный уровень боя. Способность ИИ быть эффективным в бою важна в этих жанрах. Общая цель сегодня сделать ИИ более человечным или, по крайней мере, выглядеть так.

Одной из наиболее позитивных и эффективных функций современного игрового ИИ является способность охотиться. ИИ изначально реагировал на событие очень простым способом. Если бы игрок находился в определенной области, то ИИ реагировал бы либо полностью наступательным способом, либо был бы полностью оборонительным. За небольшой период времени идея "охоты" была модернизирована; в этом «охотничьем» состоянии ИИ будет искать реалистичные маркеры, такие как звуки, издаваемые персонажем, или следы, которые они могут оставить позади. Эти события в конечном счете позволяют более сложную форму игры. С помощью этой функции игрок может на самом деле подумать, как приблизиться или избежать врага. Эта особенность особенно распространена в жанре стелс.

Еще одним событием в развитии игровом ИИ стало "инстинкт выживания". Игровые компьютеры могут распознавать различные объекты в окружающей среде и определять, является ли это полезным или вредным для его выживания. Как и пользователь, ИИ может искать укрытия в перестрелке, прежде чем предпринимать действия, которые могут сделать его уязвимым, например перезарядить оружие или бросить гранату. Могут быть установлены маркеры, которые сообщают, когда нужно реагировать определенным образом. Например, если ИИ получает команду для проверки его здоровья на протяжении всей игры, то можно установить дополнительные команды, чтобы он реагировал определенным образом с определенным процентом здоровья. Если здоровье ниже определенного

порога, тогда ИИ может быть настроен убегать от игрока и избегать его, пока не сработает другая функция. Другим примером может быть, если ИИ заметит, что у него нет пуль, он найдет объект прикрытия и будет прятаться за ним, пока не перезарядится. Подобные действия делают ИИ более человечным. Тем не менее, в этой области все еще существует потребность в улучшении.

Рисунок 3 – ИИ в укрытии

Другой побочный эффект боевого ИИ возникает, когда два контролируемых ИИ персонажа сталкиваются друг с другом; впервые получившая распространение в игре id Software Doom, в определенных ситуациях могут вспыхнуть так называемые «сражения с монстрами». В частности, агенты ИИ, которые запрограммированы реагировать на враждебные атаки, иногда атакуют друг друга, если атаки их когорты приземляются слишком близко к ним. В случае Doom, опубликованные руководства по геймплею даже предлагают использовать преимущества борьбы с монстрами, чтобы пережить определенные уровни и настройки сложности.

1.3 Важность традиционного ИИ в игровом

Разработчики предполагают, что академические разработки искусственного интеллекта могут играть роль в игровом искусственном интеллекте помимо традиционной парадигмы искусственного интеллекта, контролирующей поведение NPC. Он выделяет четыре других потенциальных области применения:

1. Моделирование опыта игрока: распознавание способностей и эмоционального состояния игрока, чтобы соответствующим образом адаптировать игру. Это может включать в себя динамическую балансировку сложности игры, которая заключается в настройке сложности видеоигры в режиме реального времени на основе способностей игрока. Игровой ИИ также может помочь определить намерения игрока (например, распознавание жестов).

2. Генерация процедурного контента: автоматическое создание элементов игровой среды, таких как условия среды, уровни и даже музыка. Методы ИИ могут генерировать новый контент или интерактивные истории.

3. Интеллектуальный анализ данных о поведении пользователей: это позволяет разработчикам игр исследовать, как люди используют игру, какие части они играют чаще всего и что заставляет их прекращать играть, позволяя разработчикам настраивать игровой процесс или улучшать монетизацию.

4. Альтернативные подходы к неигровым персонажам. Они включают в себя изменение игровой настройки для повышения правдоподобия неигровых персонажей и изучение социального, а не индивидуального поведения NPC.

1.4 Виды игрового искусственного интеллекта

Виды игрового ИИ:

− неигровые персонажи (NPC) — как правило, эти персонажи являются дружественными или нейтральными к человеческому игроку;

− боты (Bot) — враждебные к игроку персонажи, приближающиеся по возможностям к игровому персонажу; против игрока в любой конкретный момент сражаются небольшое количество ботов. Боты наиболее сложны в программировании;

− мобы (Mob) — враждебные к игроку «низкоинтеллектуальные» персонажи. Мобы убиваются игроками в больших количествах ради очков опыта, артефактов или прохождения территории.

Также есть еще нечестные «обманный» игровые ИИ (сленг «читерские»), которые помогают или же наоборот усложняют игроку прохождение игры. В игре Need for Speed (игра жанра гонки) такие ИИ применятся для обоих случаях. Когда игрок отстает вражеская машина притормаживает что дает шанс игроку его догнать, но есть триггеры, при которых враг едет намного быстрее чем раньше, что может позволит преодолеть возможности машины противника. Так же встречаются противники, которые едут по ранее заданной траектории и их невозможно остановить или как-то повлиять на его траекторию.

1.5 Вывод к первой главе

Игровой искусственный интеллект хоть и молод, но развивается не хуже, чем традиционный искусственный интеллект. Было много споров ученых традиционного ИИ, связанные с принадлежностью игрового ИИ к их области науки.

Так как среда разработки игрового ИИ игровая, а в ней по идеи нет чего-то не известного для ИИ, если это только не умышленно им не показано, то и решение задачи может быть упрощена. Игровой ИИ может как простым в представление, так и сложным в се зависит от игры, которую разрабатывают. Вспомнить нейронную сеть которое в интернете обучалась, играя в шахматы с игроками, сама собой и другими нейронными сетями. Его в теории можно назвать игровым искусственным интеллектом, но из-за требование большого требования к компьютерным ресурсам он будет очень требовательным к компьютерам игроков. И данная нейронная сеть была построена отдельно от игры она как игрок играет в игру, получается это уже игрок – бот.

Игра в имитацию или тест Тьринга — знаменитый тест, в котором робот должен обмануть человека, убедить его что он не робот [5]. Создать иллюзию интеллекта, поведение человека. Игровой ИИ тоже иллюзия в наличие интеллекта. Есть игры, которые обманывают сознания человека, и игрок начинает сопереживать ИИ подсознательно думая, что это реально, хоть они и убеждены в том, что это просто игра. Пока традиционный искусственный интеллект решает проблему имитации разумного поведение человека игровой искусственный интеллект уже это делает. Еще сыграла важность игр в жизни людей. Некоторые люди считаю ее второй жизнью. Делают то, что в реальной бы они этого не сделали.

Игровой искусственный интеллект использует все что только есть в нашем мире, но не все вместе реализуется и за сложности в разработке. В нем есть реализация поиск путей, дерево поведений также могут использоваться в механизмах, таких как интеллектуальный анализ данных и генерация процедурного контента.

Разработка игрового искусственного интеллекта сильно зависит от способностей в творческом подходе разработчика.

2 Игровой движок Unreal Engine 4

Unreal Engine 4 (UE4) — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Данный движок полностью бесплатный с 2015 года, при установке данного движка не нужно ничего докупать. По лицензионному соглашению (п. 5 ч. 10) при доходе больше $3000 за квартал. Поддерживает множество платформ и операционных систем. Поддержка языков: C++ и для написания скриптов – Python [6,7].

Blueprints — это система визуального скриптинга UE4. Она является быстрым способом создания прототипов игр. Вместо построчного написания кода всё можно делать визуально: перетаскивать ноды (узлы), задавать их свойства в интерфейсе и соединять их «провода». Пример соединение узлов на (Рисунок 4).

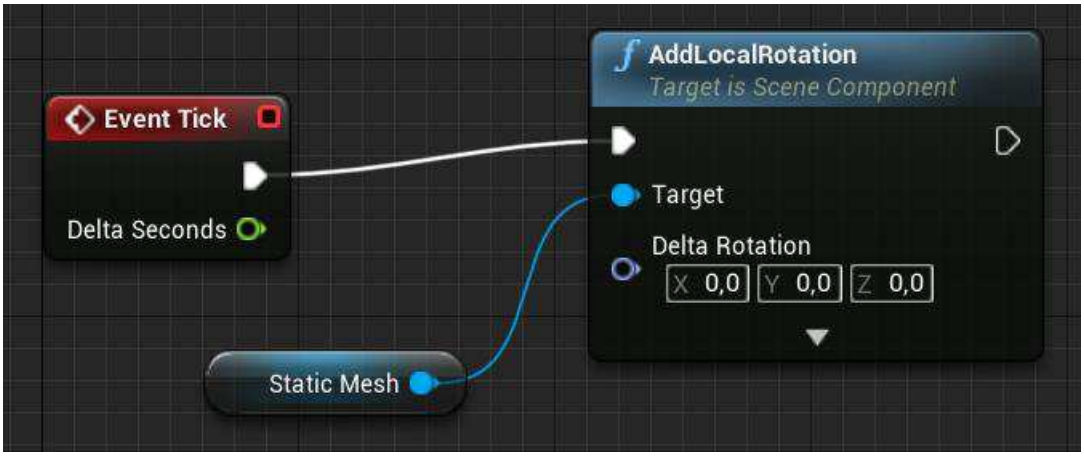


Рисунок 4 – Соединение узлов

В данном примере Event Tick это событие, которое срабатывает каждый раз, когда обрабатывается кадр в игре, то есть срабатывает с кадровой частотой. AddLocalRotation — функция поворота компонента объекта. На вход принимает: объект, который мы поворачиваем (Target, компонент Static Mesh), и координаты поворота в трёхмерной проекции (Delta Rotation).

Из таких узлов в Bluprints можно создавать как лёгкие игровые ресурсы, так и сложные в обработке. Таким образом используя blueprints, можно создать игру не разрабатывая игру с нуля используя язык программирования.

Игровые ресурсы (ассеты) — цифровой объект, состоящий из различных не значимых по отдельности компонентов. Примеры игровых ресурсов: персонаж, дерево, трава. При создание игровых ресурсов можно использовать в добавление к игровому движку другие программы как 3D max для моделирования, анимирование и Visual Studio для программирования.

Так как работа предполагает создание игровых ресурсов использование blueprints время на моделирование анимирование время не тратилось. Брались примитивы из игрового движка как цилиндр, шар так же применялась модель стола (ранее сделанная).

Создание ИИ для персонажей или других объектов в проектах на UE4 выполняется с помощью нескольких систем, работающих вместе. От дерева поведения (Behavior Trees), которое связи между различными решениями или действиями, запуска запроса для получения информации об окружающей среде через систему запросов среды (Environment Query System), до использования системы восприятия ИИ (AI Perception) для получения сенсорной информации, такой как зрение, звук или информация о повреждениях; Все эти системы играют ключевую роль в создании правдоподобного ИИ в проектах. Кроме того, все эти инструменты можно отлаживать с помощью инструментов отладки ИИ (AI Debugging), что дает вам представление о том, что ИИ думает или делает в любой момент.

Для того чтобы начать создавать ИИ нужно будет создать два ресурса движка:

− Blueprints класс родителя Character. Нужен для добавления ИИ на уровне Класс Blueprint, часто сокращенно называемый Blueprint, является активом, который позволяет создателям контента легко добавлять функциональные возможности поверх существующих классов игрового

процесса. Чертежи создаются внутри редактора движка визуально, а не путем ввода кода, и сохраняются в виде ресурсов в пакете содержимого. По сути, они определяют новый класс или тип Actor, который затем может быть помещен на уровни как экземпляры, которые ведут себя как любой другой тип Actor;

− AIController — Класс Blueprint наследуется от родителя AIController. В то время как PlayerController полагается на игрока, который принимает решение о том, что ему делать, AIController больше ориентируется на реагирование на информацию из окружения и игрового мира. Задача AIController – наблюдать за окружающим миром, принимать решения и соответственно реагировать без явного участия человека.

2.1 Дерево поведений

Дерево поведений (Behavior Trees) — игровой ресурс, используется для создания искусственного интеллекта для неигровых персонажей в проектах. Дерева поведения используется для выполнения ветвей, содержащих логику, чтобы определить, какие действие должны быть выполнены. Опирается на другой актив, называемый доской (Blackboard), который служит «мозгом» для Дерева Поведения [8].

Доска содержит несколько пользовательских клавиш, которые содержат информацию, используемую деревом поведения для принятия решений. Например, у вас есть логический ключ IsRun, на который может ссылаться Дерево Поведения, чтобы увидеть, изменилось ли значение. Если значение равно true, оно может выполнить ветвь, которая заставит NPC убежать. Если оно ложно, то можно выполнить другую ветвь, в которой NPC может случайно перемещаться по окружению.

Деревья поведения могут быть такими же простыми, как приведенный пример NPC, или такими же сложными, как симуляция другого игрока в

многопользовательской игре, которая находит укрытие, стреляет в игроков и ищет предметы.

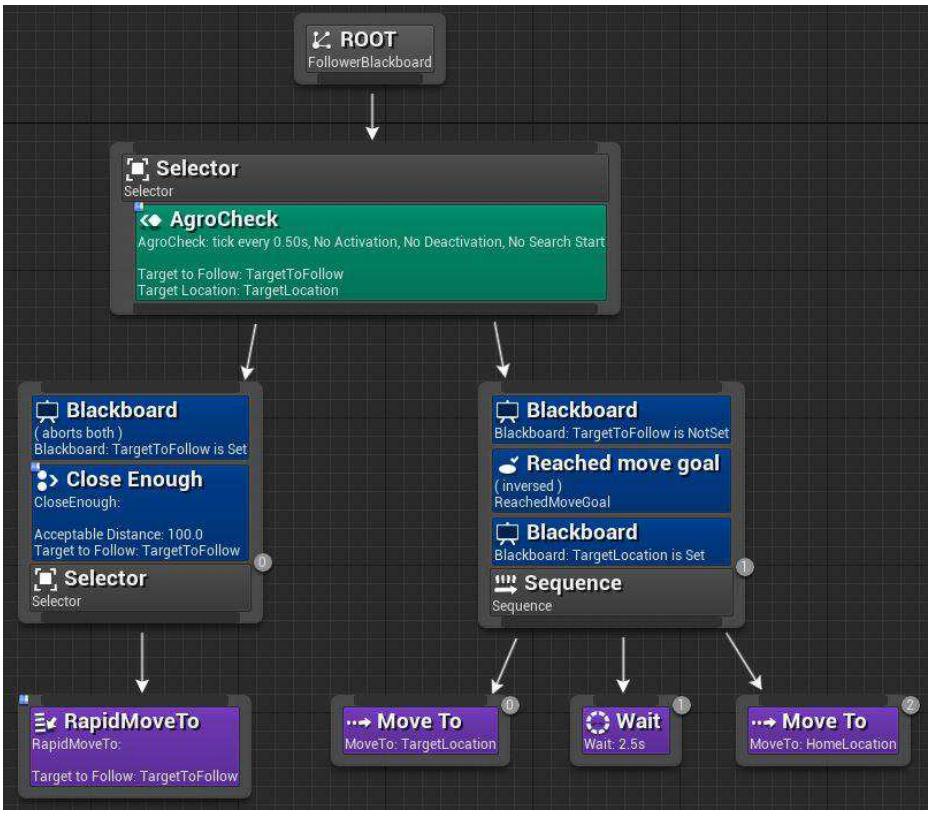


Рисунок 5 – Дерево поведений

В дереве поведения есть типы узлов такие как:

– ROOT — первый узел, от чего все начинается (Рисунок 5);

– Composite — узел который определяет корень ветви и основные правила выполнения этой ветви. Selector, sequence на (Рисунок 5);

– Task — узел задачи, определяет какое действие надо проделать. Он не имеет выходного соединения. Move to, Wait и другие с таким же цветом (Рисунок 5);

– Description — узел условия, присоединяется к другому узлу. Принимает решение будет ли выполняться данная ветвь (к которой присоединен) в дереве или даже один узел. Blackboard, Close Enough и другие с таким же цветом (Рисунок 5);

– Service — узел с частотой выполнения, присоединяется к составным узлам. Действует пока выполняется их ветвь. AgroCheck в Selector (Рисунок 5).

При использование данных узлов и создается дерево поведений

2.2 Система запросов среды

Система запросов среды (Environment Query System или EQS) — это функция инструментов ИИ в UE4 которую можно использовать для сбора данных об окружающей среде [9]. Затем с помощью генератора система может задавать вопросы об этих данных с помощью различных пользовательских тестов, возвращая лучшую оценку, который соответствует типам задаваемых вопросов [10]. Он состоит из нескольких систем:

– Generator — генерирует элементы которые будут, фактически проверены и взвешены;

– Context — предоставляет систему координат для различных тестов и генераторов;

– Тесты — определяет, как запрос среды решает, какой элемент из генератора является наилучшим вариантом.

Рисунок 6 – Узел в дереве поведений

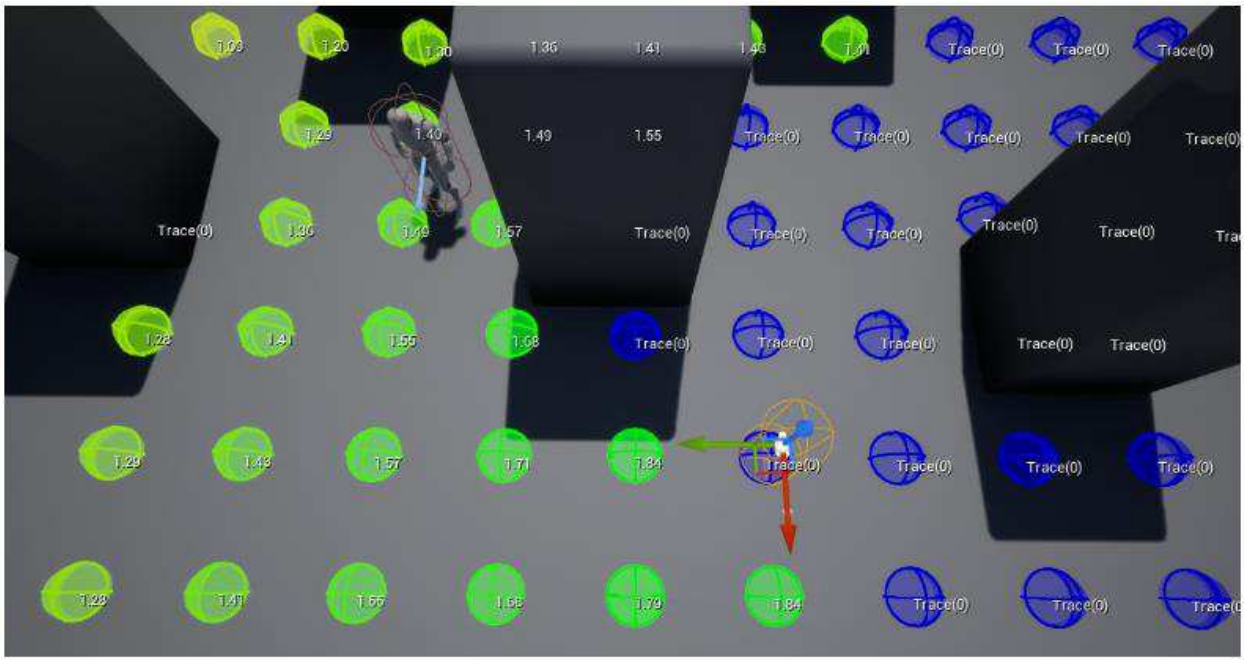


Рисунок 7 – Проведения теста EQS на уровне игры

2.3 Восприятие ИИ

Восприятие ИИ (AI Perception) — компонент, который может быть добавлен к чертежу пешки (Pawn Blueprint). В окне компонента определяется, какие чувства следует прослушивать, параметры этих чувств и как реагировать, когда чувство обнаружело что либо [11]. Также можно использовать несколько различных функций, чтобы получить информацию о том, что было воспринято, какие актеры были восприняты, или даже отключить или включить определенный тип чувства. Например, можно включить чувства зрения назначить сектор видимости, дальность видимости при которой он увидит объект и его потеряет. Для отладки AI Perception можно использовать AI Debugging (Рисунок 8).

Рисунок 8 – Зрительное восприятие ИИ

2.4 Отладка ИИ

Отладка ИИ (AI Debugging) — описывает различные способы отладки вашего ИИ с помощью инструментов отладки (появилась в версии Unreal Engine 4.21).

Создав объект ИИ, вы можете диагностировать проблемы или просто просматривать действия ИИ в любой момент, используя инструменты отладки ИИ. После включения вы можете переключаться между просмотром деревьев поведения, система запросов среды (EQS) и системы восприятия AI в одном и том же централизованном месте (Рисунок 9).

Рисунок 9 – Режим отладки при включение всех функций

2.5 Подготовка игровых ресурсов для разработки

При разработке прототипов игрового искусственного интеллекта будут использоваться анимации и их 3D моделей, а также уровень деревни взятые из магазина игрового движка UE4 как бесплатный ресурс для разработки.

2.5.1 Лиса и волк

В источники от куда данные животные были взяты были взяты два набора игровых ресурса такие как лиса и волк. В этот набор из каждого из животных входили такие ресурсы как:

- 3D модель;
- скелет;
- набор анимаций.

Рисунок 10 – Модель лисы

Рисунок 11 – Модели волка

У лисы был один окрас модели (Рисунок 10), когда у волка их три (Рисунок 11).

Данный набор ничем не примечательный и больше в них ничего нету.

2.5.2 Модель гуманоида

Данный набор набирал в себе очень много элементов для разработки таких как:

- две 3D модели. Одна с настраиваемой расцветкой, другая с одной;
- анимации. Большой спектр на все случае кроме атаки;
- скелет с множественными костями реализованные по подобию человека;

– различное поведение при изменении элементов связанные с предметами в руках и характера.

Данная модель, отображённая на рисунке, ниже была взята из стандартного пакета игрового движка UE4.

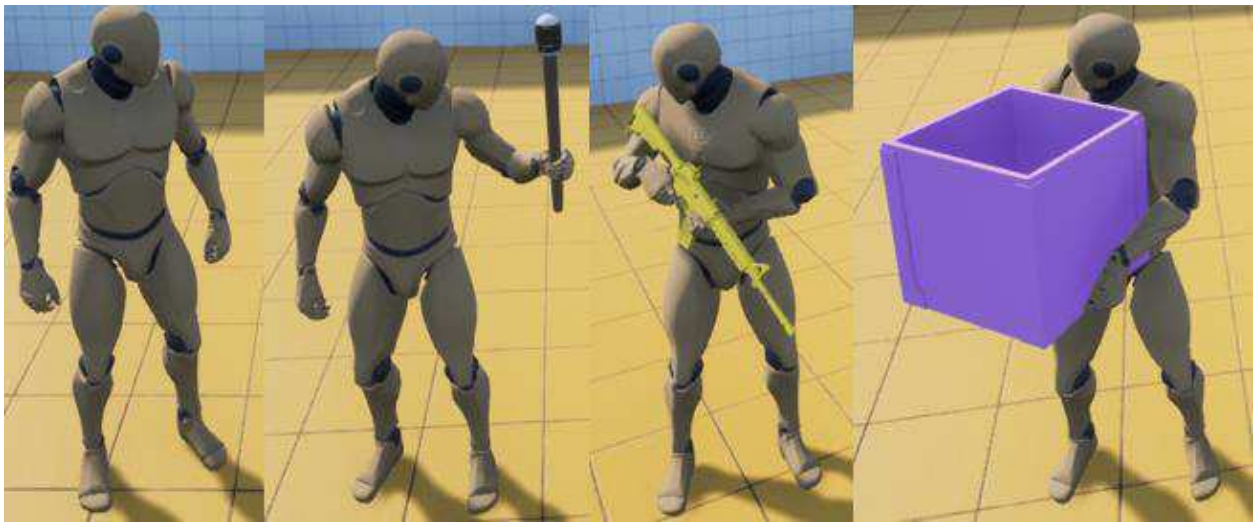


Рисунок 12 – Модель гуманоида

2.5.3 Игровой уровень

В наборе идет два уровня:

– уровень, где расставлены все ресурсы данного набора для создания деревни;

– уровень, где используя ресурсы создана деревня.

Так же в данном наборе можно взять по отдельности каждый ресурс и можно создать свою деревню.

Рисунок 13 – Игровой уровень деревни

2.6 Вывод по второй главе

Инструментов для разработки игрового искусственного интеллекта множество, а как их использовать прерогатива разработчика. Исходя из набора найденных игровых ресурсов будет строится свои прототипы искусственного интеллекта по возможности реализации будет выбираться характер и поведение данных ИИ.

Реализация ИИ в игровом движке по осмыслению проста. Реализация сводится к прописанную сценария действий игрового ИИ, но не все можно прописать в сценарии поэтому лучше всего будет поведение анимации прописывать в Blueprint. При разработке следует учесть AI Perception для восприятие окружающих пешек. EQS можно использовать для реализации входа или выхода из поле зрения противника.

3 Реализация прототипов игрового искусственного интеллекта

3.1 Гуманоидный тип игрового ИИ

3.1.1 Blueprint класс

При реализации уже имелось два Blueprint класса наследника родителя Character (входит в пакет игровых ресурсов). Для того чтобы их случайно не поломать при разработке было решено создать свой наследуя класс, который был в наборе. В нем были реализованы функции такие как:

- DoWalk — функция смены передвижения на ходьбу;
- DoRun — функция смены передвижения на бег;
- DoSprinting — функция смены передвижения на еще больший бег;
- DoDamageHP — функция с входным параметром – количества урона;
- DoDead — функция смерти пешки;
- AppenedStringHP — функция для отображение здоровья;
- DoBehavior — функция для смены экипировки (оружия) и для фиксации урона сколько нанесет урон данный предмет;
- UpdateIsAttack — функция обновление переменной IsAttack в Blackboard;
- SetStringAttack — функция для визуального отображения выстрела;
- SetStringDamage — функция для отображение количества урона;
- UpdateTargetEnemyController — функция для смены цели. Используется, когда ИИ не знал о противнике, но после получения урона он должен его узнать.

- DoAttack — функция которая вызывает событие при котором ИИ начинает стрелять. Для анимации прицеливания.

Используемые переменные:

- IsDead — bool значение, мертва ли пешка или нет;
- IsWalk — bool значение, идет ли пешка или нет;
- IsRun — bool значение, бежит ли пешка или нет;
- IsSprinting — bool значение, сильный бег или нет;
- IsAttack — bool значение, может атаковать ли пешка или нет;
- IsTargetAttack — bool значение, есть ли цель для атаки в поле видимости или нет;
- IsDefender — bool значение, цель для защиты есть или нету;
- HP — integer значение, отображает здоровье;
- Damage — integer значение, отображает урон;
- TargetPoint — array enum, хранит точки для патрулирования;
- OverlayStateBP — enum значения для хранения в себе то оружие которое было ему назначено.

При разработки было решено проверять каждый тик о том, что ИИ умер, обновление визуального отображение здоровья, и обновление о том надо ли сейчас ИИ доставать оружия в зависимости от того есть у него враг полез зрения или нет.

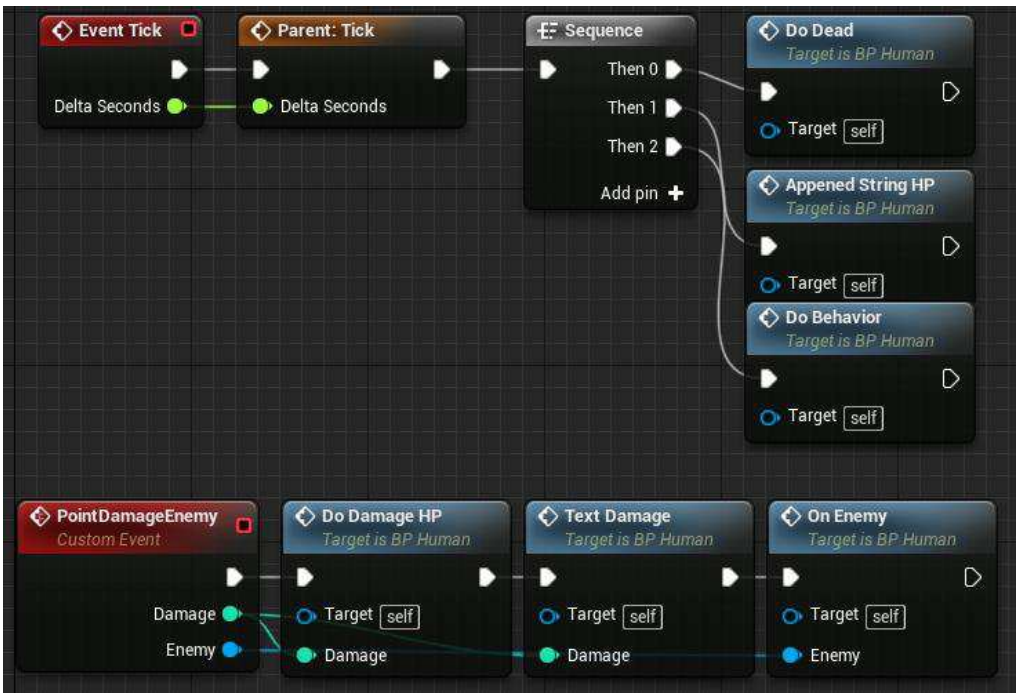


Рисунок 14 – Проверка переменных и получение урона

PointDamageEnemy — событие которое срабатывает когда ИИ получает урон. Передается информация о враге и урон.

Реализация атаки с визуальным отображением что произведена атака показана ниже.

Рисунок 15 – Событие атаки

При атаке ИИ наводит на врага оружие и появляется надпись SHOT после прохождение 1 секунды ИИ убирает ружье.

Пример отображения здоровья, урона и выстрела на (Рисунок 16).

Рисунок 16 – Визуальное отображение информации

3.1.2 Контроллеры ИИ

Для управление пешкой используется AIController. В реализация в зависимости от выбранного вида ИИ реализован свой контроллер.

Для ИИ который ходит с игроком используется такие функции как:

- UpdatetagetActorFriend — функция с входным параметром Actor значение Player, для обновление ключа TargetActorPlayer в blackboard;
- UpdateSightKey — функция с входным параметром bool значение HasLineOfSight, для обновление ключа HasLineOfSight в blackboard;
- UpdateTargetEnemyKey — функция с входным параметром Actor значение TargetEnemy, для обновление ключа TargetEnemy в blackboard;
- UpdateGoToPlayer — функция описывающее стиль ходьбы ИИ при следовании за игроком и сама функция для движения за игроком;
- UpdateIsTargetAttack — функция обновляет значение IsTargetAttack в blackboard;
- UpdateAttacking — функция с входным параметром IsAttacking обновляет значение IsAttack в Blueprint классе;

– IsDead — функция с входным параметром типом Actor для проверки жив он или нет, возращает параметр bool;

– UpdateIsAttack — функция для проверки приказа игрока о нападении и обновление значения IsAttack в blackboard;

– CheckTargetEnemyIsDead — функция для оценки состояния врага есть он или нет, мертв или нет. Возвращает два bool значения: есть ли на данный момент враг у действующей пешки и мертва ли она;

– UpdateFocus — функция обновление фокусировки внимание пешки.

При постоянном обновлении фокусировки внимания ИИ (Рисунок 17) будет смотреть на актуального на данный момент (Рисунок 18).

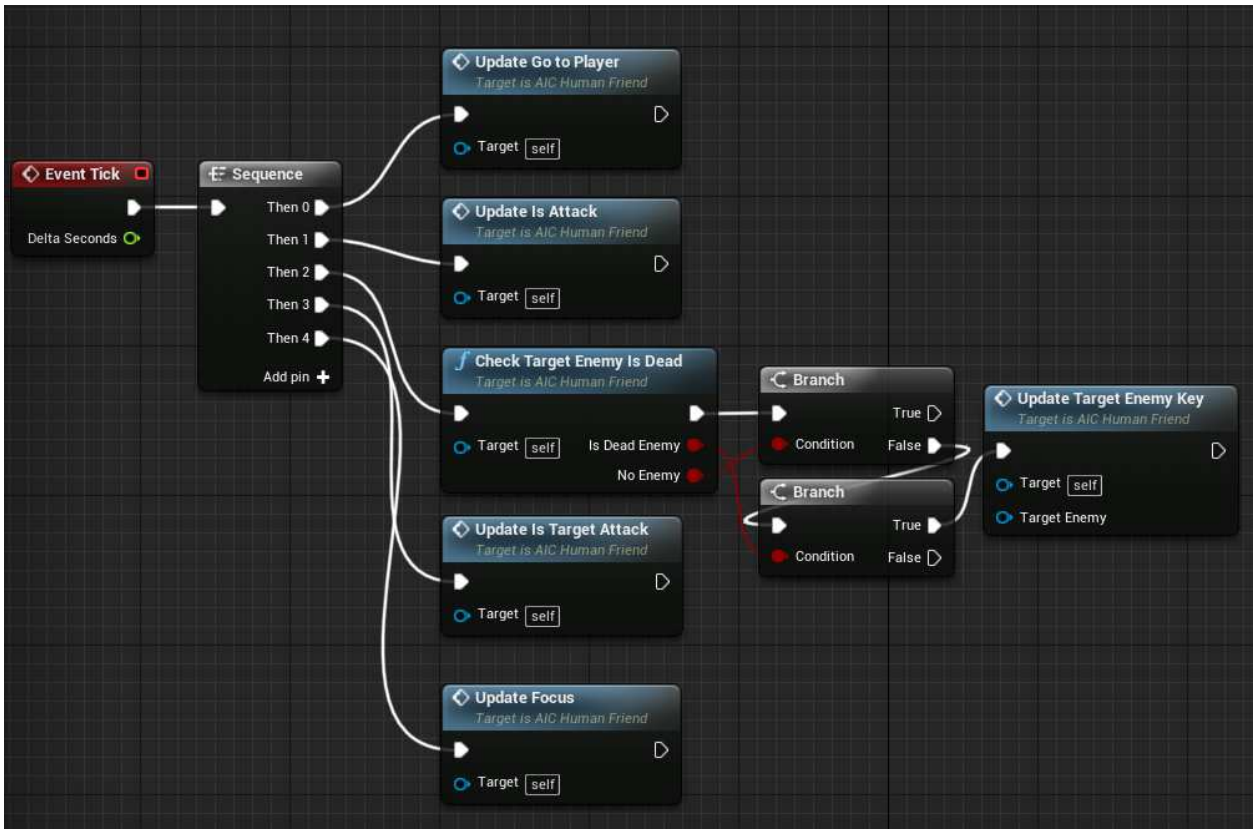


Рисунок 17 – Функции, которые срабатывают каждый такт

Рисунок 18 – Фокусировка внимания

В AIController также добавлен компонент AI Perception для добавления зрения ИИ, которым он и замечает врагов. На рисунке ниже отображено как видит ИИ.

Рисунок 19 – Восприятие ИИ

Сектор видимости (угол 90 градусов) расстояние видимости отображено розовым цветом. Зеленый периметр — это зона, в которой он сможет увидеть новую пешку и в случае, если зайдет за пределы розовой зоны, то он ее потеряет. Зеленые сферы показывают, где последний раз видел ИИ пешку.

Восприятие привязано к кости head скелета если фокусировка измениться, то изменится и положение сектор восприятия. В контроллере реализовано событие, которое обрабатывает видимость пешек (ПРИЛОЖЕНИЕ А).

Восприятие у всех прототипов все идентичные за исключением вида NPC – заложник.

У вражеского ИИ такое же восприятие, как и дружественного за исключение тега пешки, которую он будет воспринимать как враг. Так же используются те же функции, как и у дружеского ИИ. При старте игры и создания экземпляра контроллера, событие, которое срабатывает (ПРИЛОЖЕНИЕ А) было подвержена изменению так как он не ходит за игроком. Было оставлена только функция Run Behavior Tree и сменено дерево поведения. Из события Event Tick (Рисунок 17) были убраны функции такие как: UpdateGoToPlayer, UpdateIsAttack.

При создании NPC – проводник, за которым игрок движется было убрана множество функций из прошлых контроллеров. UpdateGoToPlayer уже была переделана под изменением скорости игрока в обратном порядке. Из-за того, что дружеский ИИ должен поспевать за игроком, а NPC должен направлять (Рисунок 20).

Рисунок 20 – Изменение передвижение в зависимости от дистанции

При создании NPC – заложник, были так же убраны почти все функции, расширен сектор восприятия и уменьшена дальность зрения. Событие, которое работает при восприятии объектов было переделана (ПРИЛОЖЕНИЕ А). Внешне как выглядит заложник показано ниже:

Рисунок 21 – NPC – заложник

Тэг пешек, которые был изменен на тэг игрока для того, чтобы, он был спасен, когда игрок подходил близко к нему. Переделана концовка события (Рисунок 22).

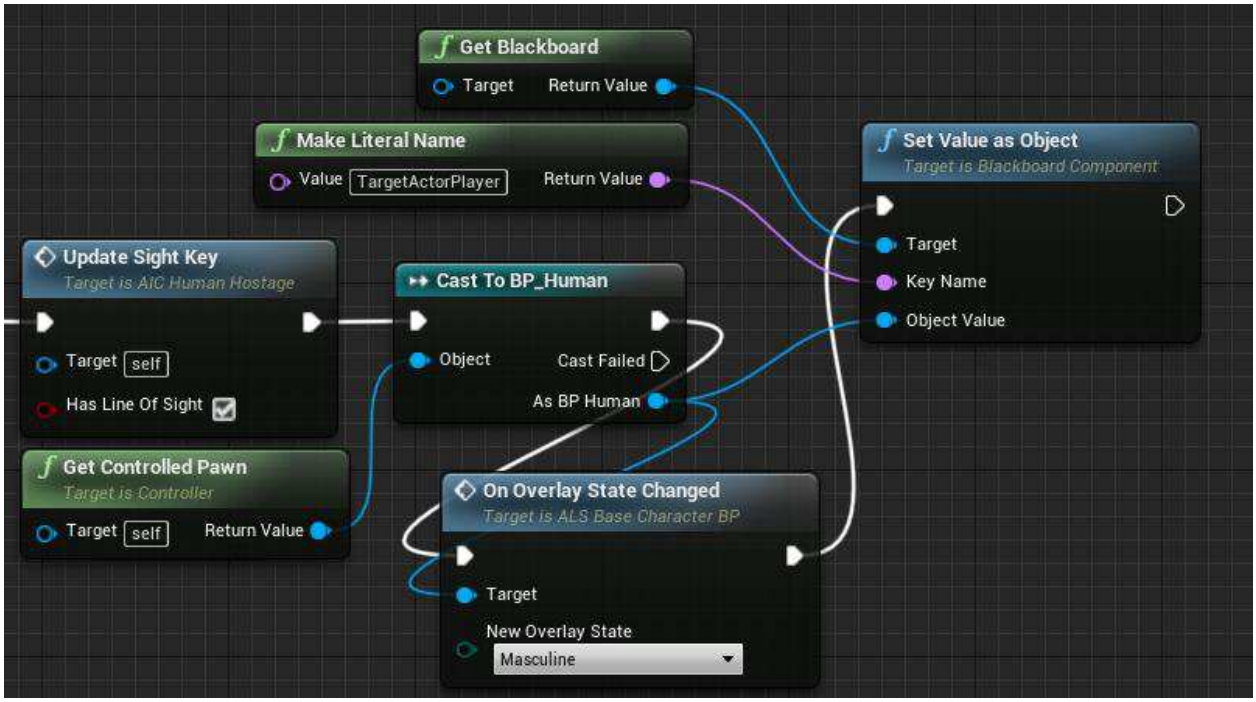


Рисунок 22 – Изменение в событие восприятия

В нем заложник в случае, если игрок появился в поле зрении выходит из состояния заложника в blackbord передается ссылка игрока, тем самым запуская работу дерева поведения

3.1.3 Дерево поведений

Так как дружественный ИИ был для того, чтобы он просто ходил рядом с игроком и атаковал по приказу игрока то дерево поведение выглядит:

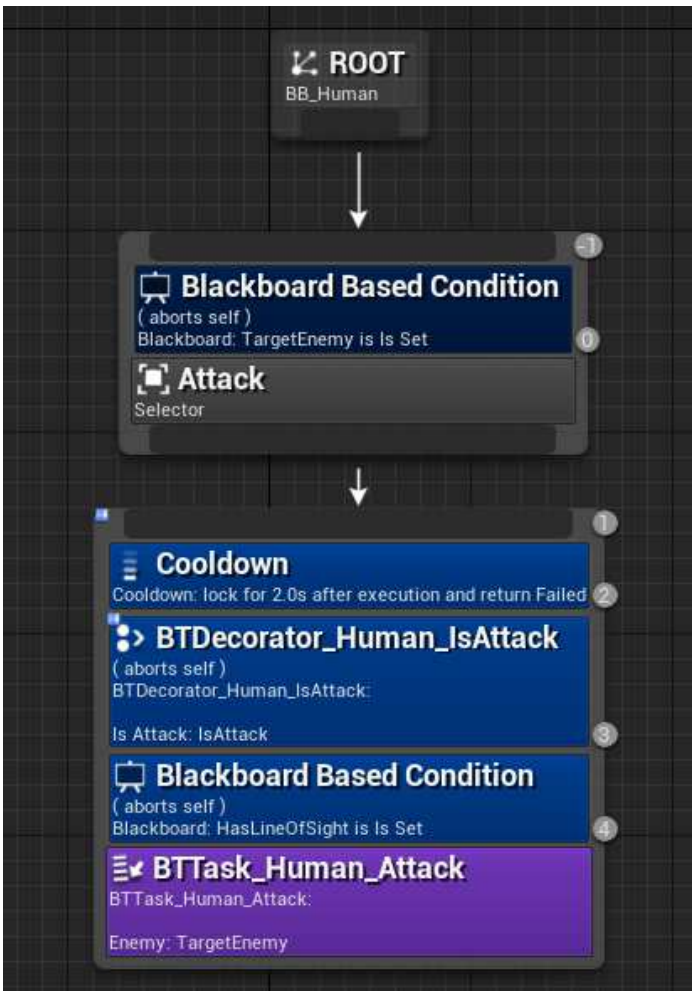


Рисунок 23 – Дерево поведения дружественного ИИ

В нем реализована только атака (ходьба за игроком задана в контроллере). С декораторами такие как:

- Blackboard Based Condition — условие что в blackbord ключ TargetEnemy не пустой;
- Cooldown — задержка перед повторной атаки;
- BTDecorator_Human_IsAttack — условие что игрок отдал приказ атаки;
- Blackboard Based Condition — условие что враг находится в прямой видимости.

BTTask_Human_Attack — задача на выполнение атаки противника (Рисунок 24).

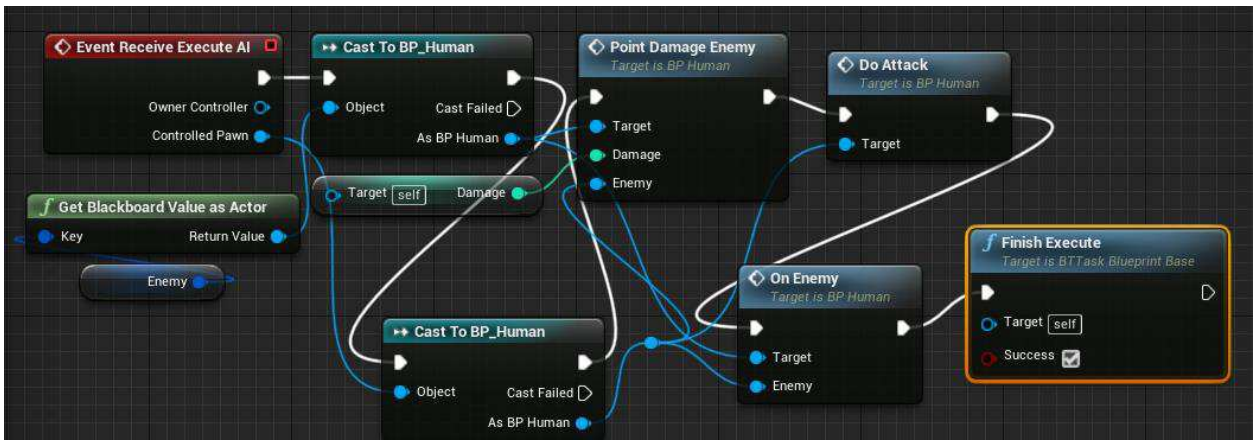


Рисунок 24 – BTTask_Human_Attack

Дерево поведений у вражеского бота имеет более сложное представление (ПРИЛОЖЕНИЕ Б). В случае если враг есть, то он будет его атаковать и будет передвигаться в то положение, где враг находится в прямой видимости. Если врага нету, то он будет патрулировать. Патрулирование происходит по двум случаям:

– если есть контрольные точки для патрулирования будет патрулировать их;

– если нету контрольных точек патрулирует в случайном направлении.

Были добавлены новые декораторы и задачи, и запрос EQS.

– BTDecorator_Human_NoneTargetPoint — декоратор для проверки есть ли контрольные точки.

– Run EQS Query — EQS, генерирует координаты куда лучше пойти для видимости противника.

– BTTask_Human_GetTargetPoint — задача для получение следующей контрольной точки.

Дерево поведение у NPC – проводник, в случае если рядом игрока нету он будет случайно бродить. По появлению игрока он начнет перемещения по контрольным точкам (Рисунок 25).

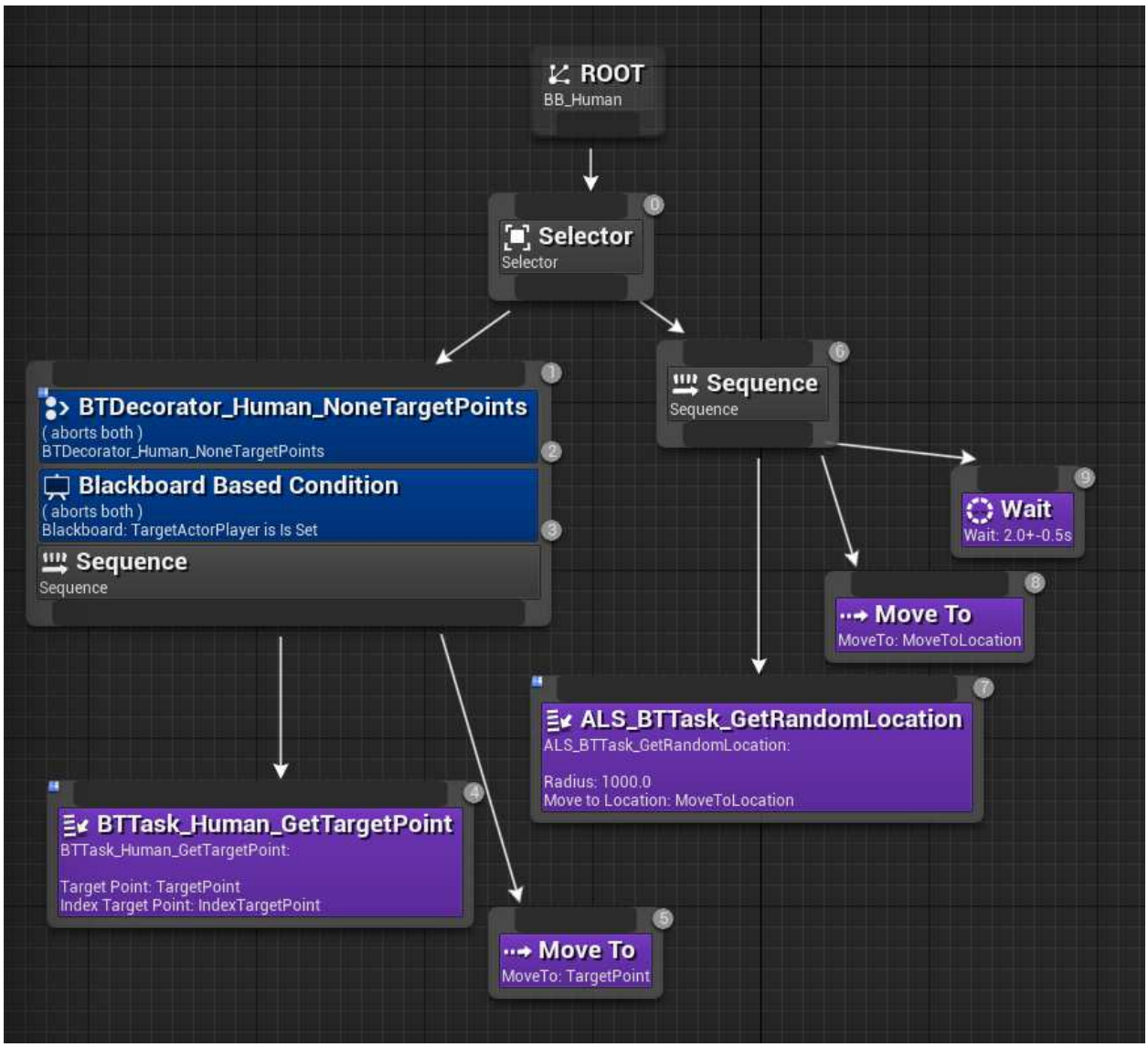


Рисунок 25 – Дерево поведения у NPC – проводник

Дерево поведение у NPC – заложника тоже, как и у проводника не очень большое (Рисунок 26). Поведение заложника заключается в блокировке поведения используя декоратор. По появлению игрока в поле восприятия дерево поведений раскрывается полностью. В качестве основного поведение было выбрано случайное перемещение.

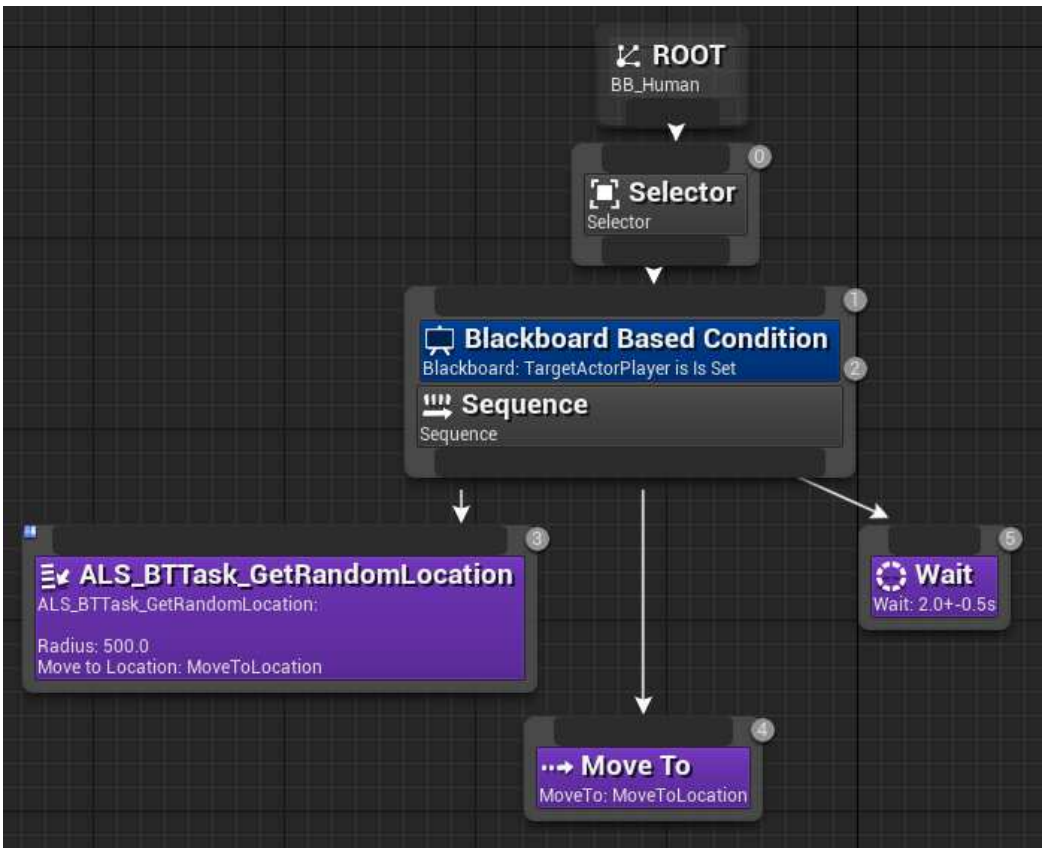


Рисунок 26 – Дерево поведения у NPC – заложник

3.1.4 Blackboard общий

В blackboard хранятся ключи, которые используется в дереве поведения. Он является общий для всех деревьев поведений ИИ гуманоидного типа.

Ключи:

- HasLineOfSight — bool значение, которое показывает цель в прямой видимости или нет;
- MoveToLocation — vector значение, мировые координаты. Используется для перемещения;
- TargetActorPlayer — actor значение, ссылка на пешку игрока;
- TargetEnemy — actor значение, ссылка на пешку врага;
- IsTargetAttack — bool значение, показывает есть ли цель – враг;

– IsDead — bool значение, показывает мертва ли действующая пешка;

– IsAttack — bool значение, показывает может ли атаковать действующая пешка;

– TargetPoint — TargetPoint значение, ссылка на TargetPoint в мире;

– IndexTargetPoint — integer значение, хранит индекс TargetPoint в массиве TargetPoints в blueprint.

3.2 Лиса и волк

3.2.1 Blueprint класс

При реализации Blueprint класса было реализовано меньшее количество функций чем у гуманоидного типа, потому что в пакете игрового ресурса его не было. Пришлось создавать класс Animation родителя AnimInstance, в котором реализован переход поведений ИИ (Рисунок 27).

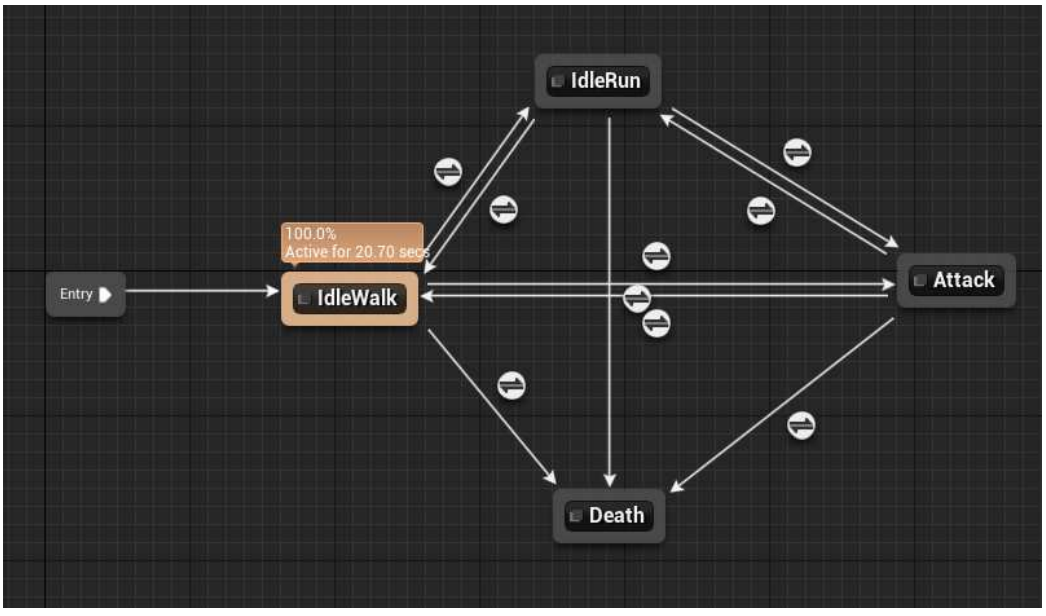


Рисунок 27 – Граф перехода анимации

Для того чтобы знать какая была анимация задействована были реализованы bool значение которые описывали это:

- IsDead — значение говорящая мертв моб или нет;
- IsAttacking — атакует ли моб или нет;
- IsRun — бежит или идет.

Эти переменные дублировались в Blueprint классе. И при изменении в этом классе выше указанных значений происходил переход между анимациями. Несколько кадров которые показывают, как выглядят анимации.

Рисунок 28 – Волк и лиса бегут на друг друга

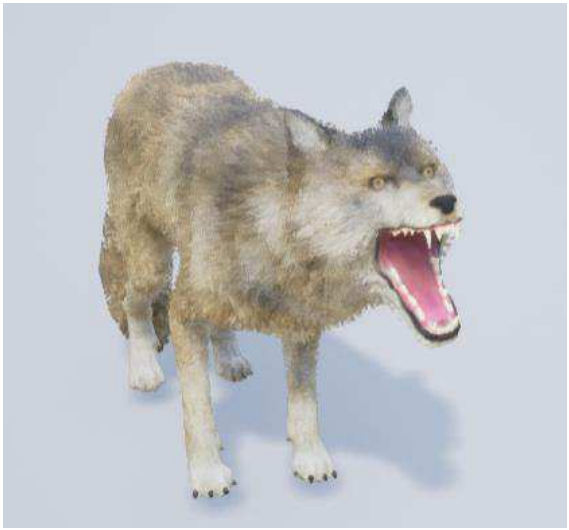


Рисунок 29 – Волк атакует

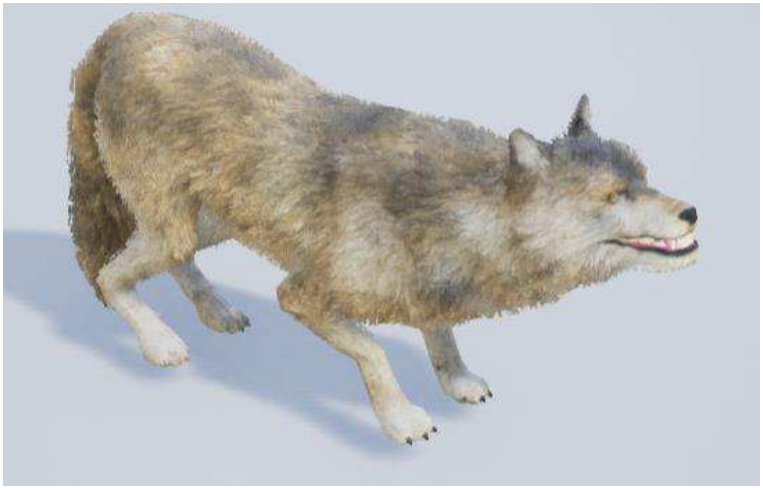


Рисунок 30 – Волк в агрессивной стойке

Так как волк и лиса сражаются с ближнего расстояния то при реализации было добавлена коллизии сферы при соприкосновении с противником наносила урон:

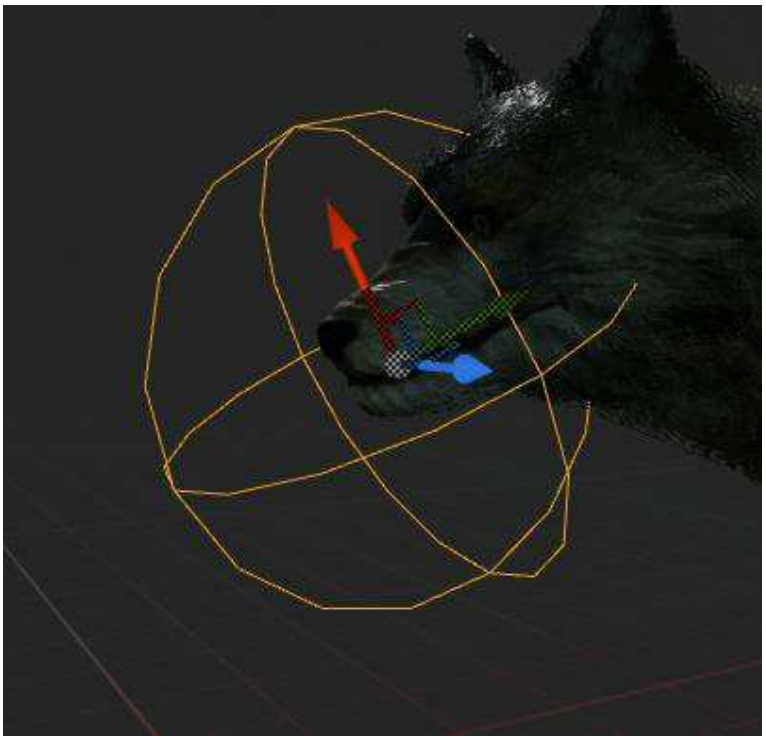


Рисунок 31 – Коллизия для нанесения урона

При соприкосновении коллизии срабатывало событие:

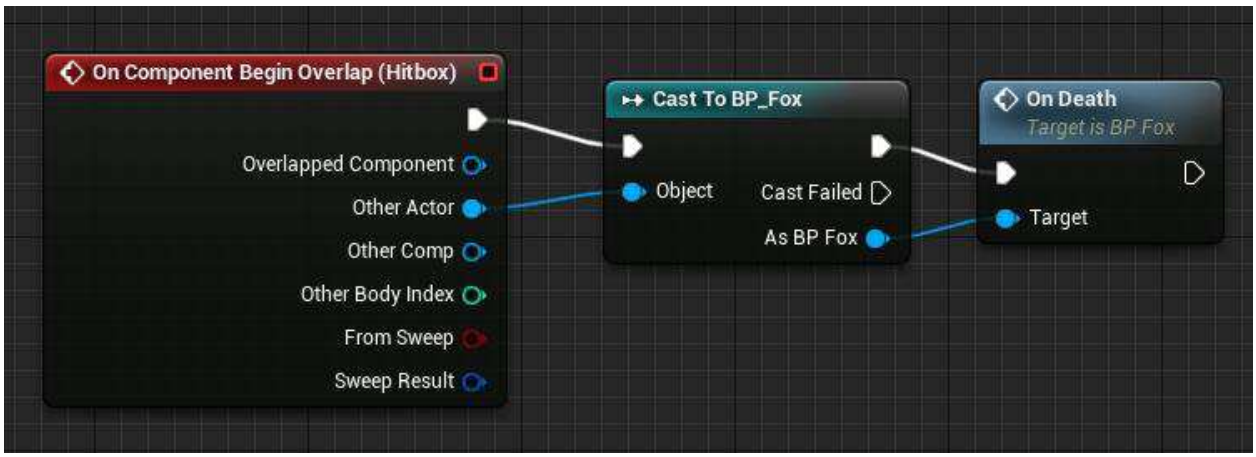


Рисунок 32 – Событие пресечение коллизии

OnDeath — запуск события. Чтобы случайно не сработала используется bool переменная IsAttacking. Чтобы пешка была только тогда, когда значение true. Были реализованы такие функции как:

- DoAttack — функция для атаки;
- DoDamageHP — функция для получение урона. Отнимает здоровье;
- DoRun — функция для увеличение скорости перемещения до уровня бега;
- DoWalk — функция для уменьшения скорости перемещения до уровня ходьбы.

После удара срабатывает событие (OnDeath), которое обрабатывает данный удар: наносит урон и, если нет здоровья — смерть пешки (ПРИЛОЖЕНИЕ В).

3.2.2 Контроллер ИИ

Контроллер схож с ИИ гуманоидного типа, а именно с восприятием. В нем есть такие функции как:

- UpdateSightKey — функция для обновление HasLineOfSight ключа в blackbord;

– UpdateTargetKey — функция для обновление TargetActor ключа в blackboard;

– UpdateRunKey — функция для обновление IsRun ключа в blackboard;

– ClouseTargetKey — функция для обновление (очистки) TargetActor ключа в blackboard.

Восприятие у лисы и волка было сделано большое по расстоянию видимости, но сектор видимости 90 градусов:

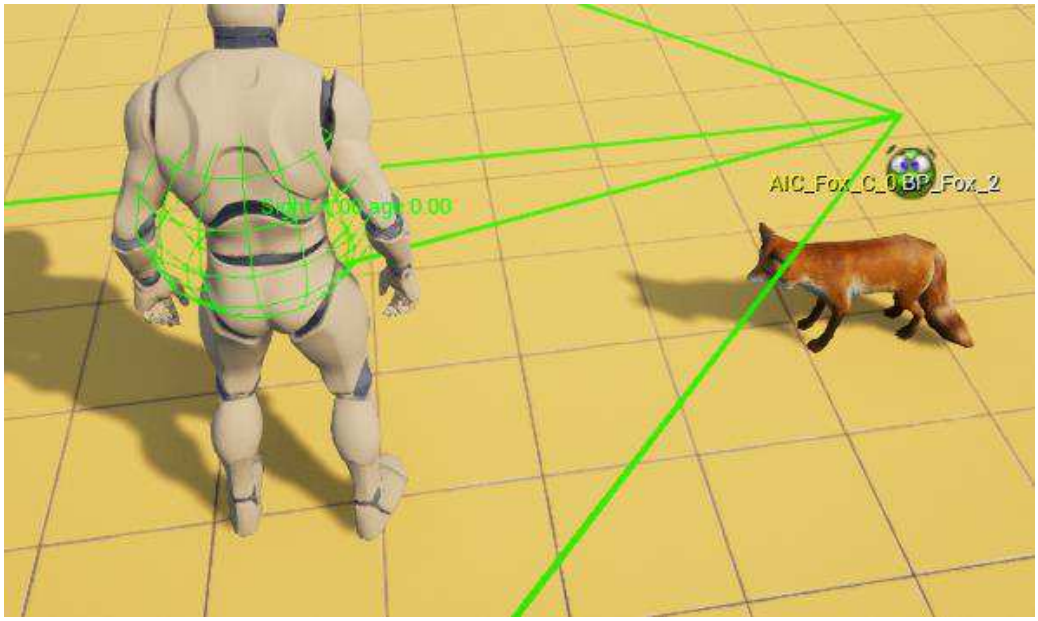


Рисунок 33 – Восприятие у лисы

Большого описание функционала в контроллере лисы и волка нет. При запуске игры запускается их дерево поведений.

3.2.3 Дерево поведения и blackboard

Дерево поведения у волка и лисы одинаковые только для каждого ИИ был создан свой экземпляр всего.

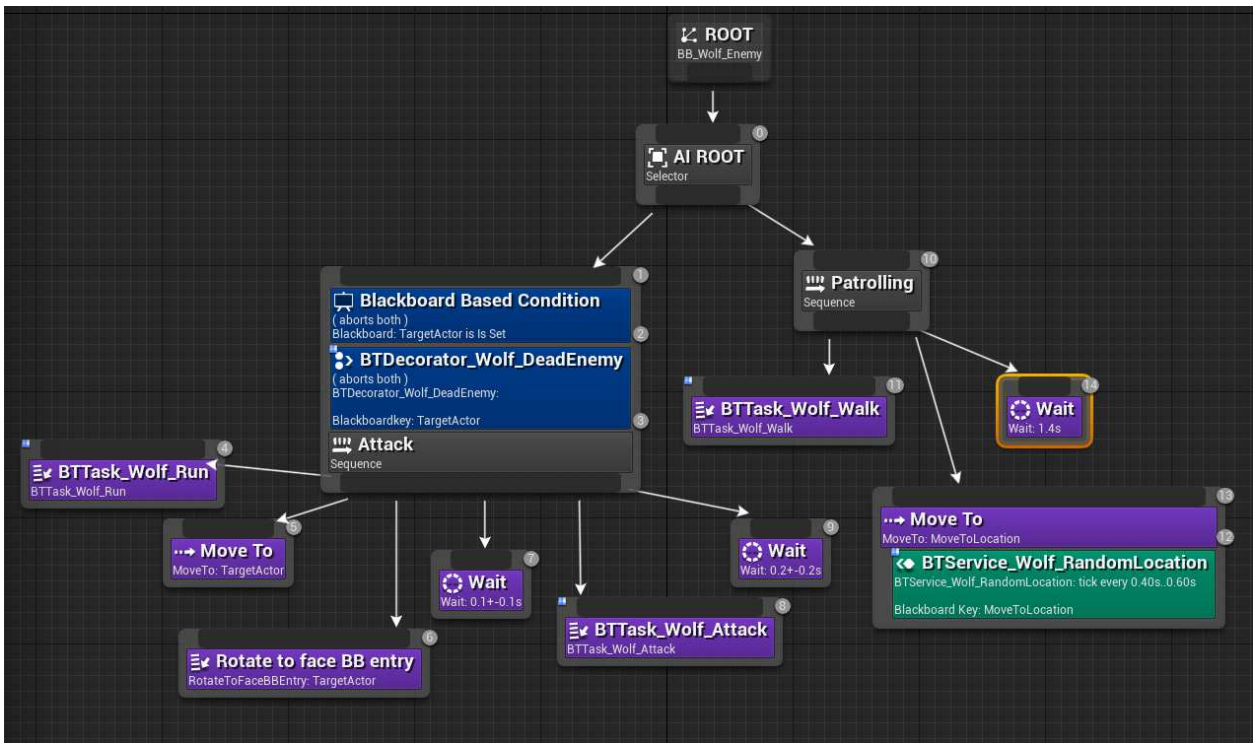


Рисунок 34 – Дерево поведения волка

– BTTask_Wolf_Run — задача, которая дает знать пешки что ей надо бежать;

– Rotate to face BB entry — задача из стандарта движка. Служит для поворота пешки в указанную сторону;

– BTTask_Wolf_Attack — задача для выполнение атаки;

– BTTask_Wolf_Walk — задача, которая дает знать пешки что ей надо идти пешком;

– BTService_Wolf_RandomLocation — служба для вычисления случайных координат. Служат после чего координатами для перемещения;

– BTDecorator_Wolf_DeadEnemy — декоратор, проверка на смерть врага.

Работает данное дерево поведений так:

– если пешка видит врага, то она переходит в режим бега и бежит к противнику, поворачивается к нему и наносит урон;

– если пешка не видит врага, то просто ходит в случайном положении.

В blackboard есть такие ключи как:

- HasLineOfsight — bool значение хранит в себе находится ли враг в прямой видимости;
- MoveToLocation — vector значение, хранит мировые координаты для передвижения;
- TargetActor — Actor значение, хранит ссылку на врага;
- IsRun — bool значение, хранит бежит ли пешка или нет.

3.3 Вывод к третьей главе

При реализации гуманоидного ИИ было реализована система фокуса на цели благодаря чего ИИ мог перемещаться, не теряя врага из виду. Атака проводится с дальнего расстояния, когда у мобов ближнее. При дальней атаке учтена обстановка, а именно если враг не в прямой видимости, то атака не совершится. Боты, как и мобы могут потерять врага из виду продолжая свое перемещения.

У гуманоидного типа есть небольшой арсенал оружия которой в зависимости от выбора будет наносить разный урон.

Список оружия:

- пистолет;
- автомат;
- лук.

Разработки игрока для взаимодействия с ИИ не проводилось. Единственная что игрок может сделать отдать приказ дружественному ИИ для совершения атаки. ИИ не воспринимают игрока. Игрок не может навредить игровыми ИИ. Он является только целью для передвижения дружественного ИИ.

Так как в наборе не было анимации выстрела было реализовано текстовое уведомления о атаки, количестве здоровья и нанесенного урона.

ЗАКЛЮЧЕНИЕ

По окончанию реализации прототипов игрового искусственного интеллекта было реализовано:

- NPC. Гуманоидный тип игрового ИИ. Первый, который лишь направлял игрока в заданный пункт при этом он двигался так чтобы игрок поспевал за ним. Второй вид – заложник. Находится в заложниках у вражеский ИИ. При подходе на достаточное расстояние игроком к заложнику он освобождался;
- бот. Первый вражеский тип при попадании в его область видимости проводит атаку. Второй тот же самый вражеский тип только он патрулирует подготовленные заранее ключевые точки. Третий дружелюбный к игроку ходит за игроком, подстраивается под движения игрока (замедляется или ускоряется). Игрок может отдать приказ атаки дружелюбному боту. Он не атакует если игрок не дал приказ, даже если атакован;
- мобы: лиса и волк. Простая реализация случайного перемещения и атаки если увидят врага.

Цель достигнута, задачи решены

СПИСОК СОКРАЩЕНИЙ

ИИ – Искусственный интеллект

UE4 – Unreal engine 4

AI – Artificial intelligence

EQS – Environment query system

NPC – Non-player character

Mob – Mobile object

Bot – сокращение от чешского robot

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Igromania.ru [электронный ресурс] Статья от Игромании “Лучший искусственный интеллект в играх, или Почему ИИ – это подделка” – Режим доступа: https://www.igromania.ru/article/29712/Luchshiy_iskusstvennyy_intellekt_v_igrah_ili_Pochemu_II-yeto_poddelka.html

2. Newtonew.com [электронный ресурс] Сергея Маркова (специалист по искусственному интеллекту), статья “Словами специалиста: вся правда об искусственном интеллекте” – Режим доступа: https://newtonew.com/tech/slovami-specialista-vsya-pravda-ob-iskusstvennomintellekte

3. Uengine.ru [электронный ресурс] Дерево поведений в Unreal engine 4 – Режим доступа: https://uengine.ru/site-content/docs/ai/behavior-tree

4. Gamer.ru [электронный ресурс] Тимур Бухараев. Искусственный интеллект в Heroes of Might and Magic V – Режим доступа: http://www.gamer.ru/heroes-of-might-and-magic-v-poveliteliordy/iskusstvennyy-intellekt-v-heroes-of-might-and-magic-v.

5. The Trouble with the Turing Test (англ.)// The New Atlantis: журнал. – Center for the Study of Technology and Society, 2006. – No. 11. – P. 42–63.

6. Unrealengine.com [электронный ресурс] Официальный сайт Unreal engine 4 – Режим доступа: https://docs.unrealengine.com/enUS/Engine/ArtificialIntelligence/index.html

7. Uengine.ru [электронный ресурс] Русскоязычный сайт Unreal engine 4 – Режим доступа: https://uengine.ru

8. Uengine.ru [электронный ресурс] Дерево поведений в Unreal engine 4 – Режим доступа: https://uengine.ru/site-content/docs/ai/behavior-tree

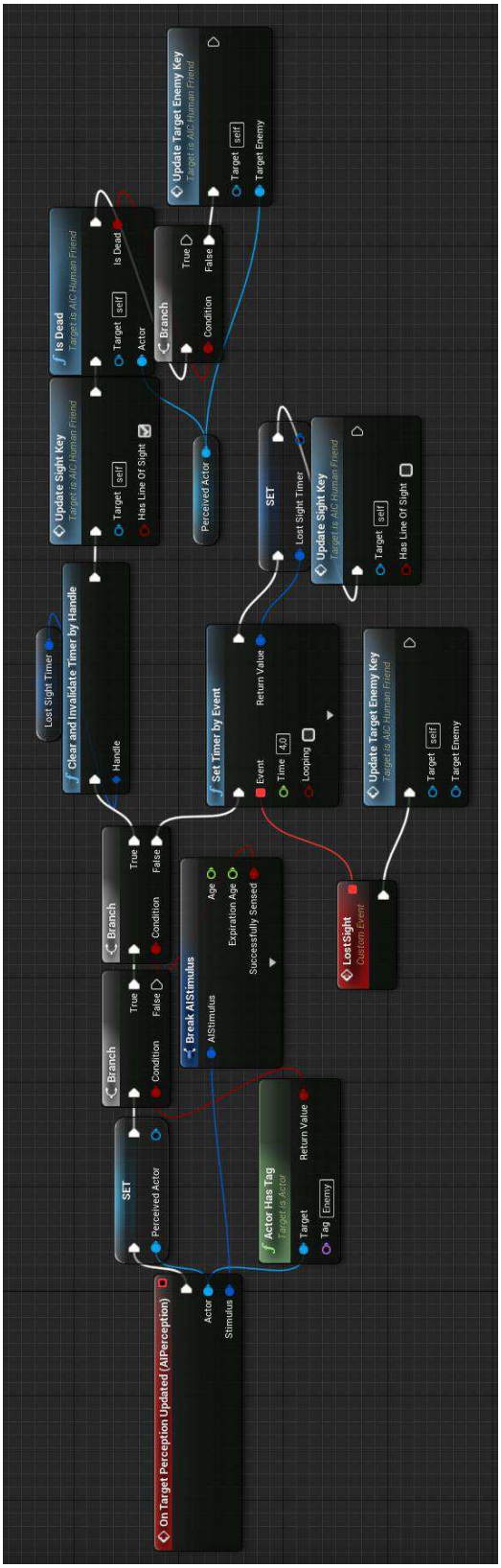
9. Unrealengine.com [электронный ресурс] Environment Query System в Unreal engine 4 – Режим доступа: https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/EQS/index.html

10. Uengine.ru [электронный ресурс] Blueprint в Unreal engine 4 – Режим доступа: https://uengine.ru/site-content/docs/blueprints-docs/blueprint

11. Unrealengine.com [электронный ресурс] AI and Behavior Trees в Unreal engine 4 – Режим доступа: https://docs.unrealengine.com/en-US/Gameplay/AI/index.html

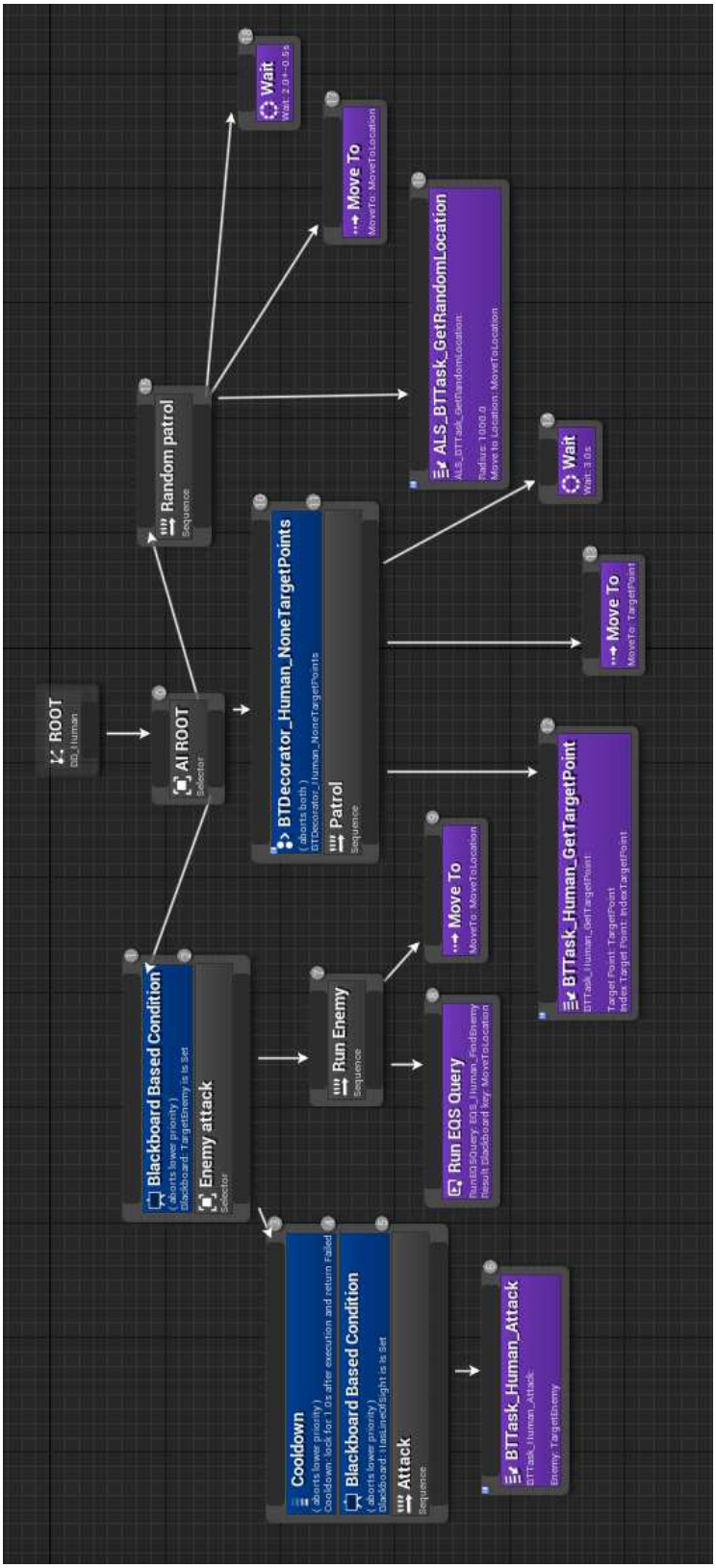
ПРИЛОЖЕНИЕ А

Событие восприятия у дружеского бота



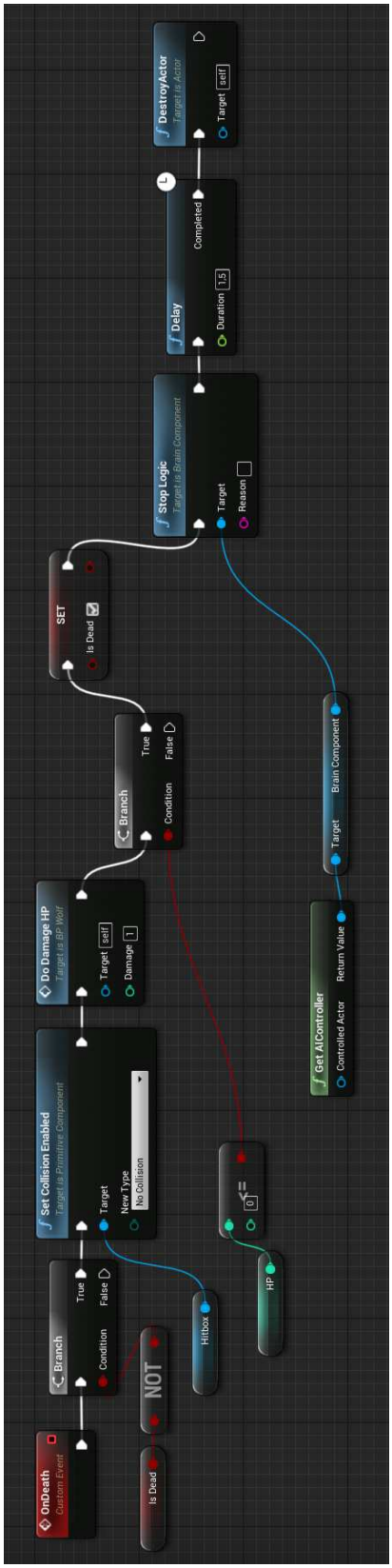
ПРИЛОЖЕНИЕ Б

Дерево поведений вражеского бота



ПРИЛОЖЕНИЕ В

Событие для обработки атаки и последующей смерти пешки



Федеральное государственное автономное образовательное
учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и информационных технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедры ИС

_____ П.П. Дьячук
подпись инициалы, фамилия

«26» 06 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – «Информационные системы и технологии»

Разработка игрового искусственного интеллекта на базе ПО компьютерных игр Unreal Engine 4

Руководитель _____ 26.06.20 доцент, к.т.н. И. А. Легалов
подпись, дата инициалы, фамилия

Выпускник Сучков 26.06.2020 Д. О. Сучков
подпись, дата инициалы, фамилия

Нормоконтролер _____ 26.06.20 ст. препод. ИС Ю. В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2020