|  |  |
| --- | --- |
| Герб МГТУ | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
|  | |
| ФАКУЛЬТЕТ | «Робототехника и комплексная автоматизация» (РК) |
| КАФЕДРА | «Системы автоматизированного проектирования» (РК6) |
|  | |
| **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** | |
| ***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ*** | |
| ***НА ТЕМУ:*** | |
| ***«Методы взаимодействия с большими языковыми моделями для управления виртуальными агентами в Unreal Engine 5»*** | |
|  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | РК6-84Б |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Больных А.С.** |
| (Группа) | (Подпись, дата) | (Фамилия И.О.) |
| Руководитель ВКР | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
| (Подпись, дата) | (Фамилия И.О.) |
| Нормоконтролёр | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Грошев С.В.** |
| (Подпись, дата) | (Фамилия И.О.) |

|  |
| --- |
|  |
| *2024 г.* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** | | | |
|  | УТВЕРЖДАЮ | | |
| Заведующий кафедрой | | | РК6 |
| (Индекс) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Карпенко А.П. | |
| (Фамилия И.О.) | |
| «\_\_» \_\_\_\_\_\_\_ 202\_ г. | | | |

|  |  |
| --- | --- |
| **ЗАДАНИЕ** | |
| **на выполнение выпускной квалификационной работы бакалавра** | |
| Студент группы | РК6-84Б |
| Больных Андрей Сергеевич | |
| (фамилия, имя, отчество) | |
| Тема квалификационной работы: «Методы взаимодействия с большими языковыми моделями для управления виртуальными агентами в Unreal Engine 5» | |
| Источник тематики (НИР кафедры, заказ организаций и т.п.): кафедра. | |
| Тема квалификационной работы утверждена распоряжением по факультету РК № от « »  202\_ г. | |
| **Техническое задание** | |
| **Часть 1.** *Аналитическая часть* | |
| *Изучить средства разработки, предоставляемые трехмерным движком Unreal Engine 5. Провести продуктовое исследование существующих LLM моделей, сравнить их характеристики. Провести анализ реализации взаимодействия выбранной LLM модели с игровыми агентами.* | |
| **Часть 2.** *Практическая часть 1. 3d-моделирование* | |
| *Создать трёхмерную сцену игрового мира с различными параметрами для игровых агентов. Найти трёхмерные модели игровых агентов и настроить их анимации. Выполнить поиск и импорт ассетов в движок Unreal Engine 5.* | |
| **Часть 3.** *Практическая часть 2. Разработка внутриигровых систем.* | |
| *Интегрировать LLM модель в движок Unreal Engine 5. Разработать систему общения LLM модели с игровыми агентами. Провести программную реализацию взаимодействия генеративных игровых агентов с внешним миром. Создать интерфейс, показывающий лог событий и данные о выбранных объектах, для конечного пользователя.* | |
| **Оформление выпускной квалификационной работы** | |
| Расчетно-пояснительная записка на 76 листах формата A4. | |
| Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): | |
| |  | | --- | | *Работа содержит 8 графических листа формата A4.* | |  | |  | |  | | |
| Дата выдачи задания: «10» февраля 2024 г. | |
| В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «21» июня 2024 г. | |

|  |  |  |
| --- | --- | --- |
| **Руководитель квалификационной работы** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | (Фамилия И.О.) |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Больных А.С.** |
|  | (Подпись, дата) | (Фамилия И.О.) |

|  |
| --- |
| Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** | | | | | |
| ФАКУЛЬТЕТ | РК |  | УТВЕРЖДАЮ | | |
| КАФЕДРА | РК6 | Заведующий кафедрой | | | РК6 |
| (Индекс) |
| ГРУППА | РК6-84Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Карпенко А.П. | |
| (Фамилия И.О.) | |
| «\_\_» \_\_\_\_\_\_\_\_ 202\_ г. | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **КАЛЕНДАРНЫЙ ПЛАН** | | | | | | | | |
| **выполнения выпускной квалификационной работы** | | | | | | | | |
| Студент группы | | | РК6-84Б | | | | | |
| Больных Андрей Сергеевич | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | |
| Тема квалификационной работы: «Методы взаимодействия с большими языковыми моделями для управления виртуальными агентами в Unreal Engine 5» | | | | | | | | |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | | 10.02.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 1 часть. Аналитическая часть | | 18.02.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | | 28.02.2024 | \_\_\_\_\_\_\_\_\_ | Заведующий кафедрой | Карпенко А.П. |
|  | 2 часть. Практическая часть 1 | | 21.04.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 3 часть. Практическая часть 2 | | 23.05.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | 1-я редакция работы | | 28.05.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Подготовка доклада и презентации | | 04.06.2024 | \_\_\_\_\_\_\_\_\_ | Студент | Больных А.С. |
|  | Заключение руководителя | | 10.06.2024 | \_\_\_\_\_\_\_\_\_ | Руководитель ВКР | Витюков Ф.А. |
|  | Допуск работы к защите на ГЭК (нормоконтроль) | | 17.06.2024 | \_\_\_\_\_\_\_\_\_ | Нормоконтролер | Грошев С.В. |
|  | Внешняя рецензия | | 17.06.2024 | \_\_\_\_\_\_\_\_\_ |  |  |
|  | Защита работы на ГЭК | | 21.06.2024 | \_\_\_\_\_\_\_\_\_ |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Больных А.С. | Руководитель ВКР | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Витюков Ф.А. |
|  | (подпись, дата) | (Фамилия И.О.) |  | (подпись, дата) | (Фамилия И.О.) |

АННОТАЦИЯ

Выпускная квалификационная работа посвящена подходам для создания генеративных игровых агентов в движке Unreal Engine 5. Описана разработка системы взаимодействия с моделью LLM для получения управляющих команд агентов.

В первой главе рассматривается создание игровой системы с окружающей средой, в которой игровые агенты будет осуществлять свою деятельность. Осуществляется подготовка и импорт трёхмерных моделей в движок Unreal Engine 5, настройка анимации для игровых агентов.

Во второй главе описываются этапы разработки основных компонентов проекта для дальнейшего использования в движке, а именно: окружающая среда с различными параметрами, дерева поведений игровых агентов, графический интерфейс для конечного пользователя с целью мониторинга объектов трёхмерной сцены.

В третьей главе описывается механизм «общения» LLM модели с игровыми агентами, а также анализ эффективности использования LLM для управления ими.

В пояснительной записке 32 страницы, 12 рисунков, 8 графических листов.

Тип работы: выпускная квалификационная работа.

Тема работы: «Методы взаимодействия с большими языковыми моделями для управления виртуальными агентами в Unreal Engine 5».

Объекты исследований: 3d-моделирование, разработка внутриигровых систем, LLM модели.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 5](#_Toc161360708)

[ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ 7](#_Toc161360709)

[ВВЕДЕНИЕ 9](#_Toc161360710)

[1. Создание игровой системы 11](#_Toc161360711)

[1.1. Правила игры 11](#_Toc161360712)

[1.2. Структура проекта 12](#_Toc161360713)

[1.3. Импорт ассетов в Unreal Engine 5 14](#_Toc161360714)

[1.4. Моделирование игрового поля 15](#_Toc161360715)

[1.5. Настройка анимации у моделей 16](#_Toc161360716)

[2. Разработка внутриигровых систем 17](#_Toc161360717)

[2.1 Описание классов 17](#_Toc161360718)

[2.2. Взаимодействие игрового агента с окружающим миром 20](#_Toc161360719)

[2.2.1. Функция CalcTemp 21](#_Toc161360720)

[2.2.2. Функция CalcFatigue 21](#_Toc161360721)

[2.2.3. Функция CalcHP 22](#_Toc161360722)

[2.3. Создание интерфейса 22](#_Toc161360723)

[3. Разработка системы взаимодействия с LLM моделью 23](#_Toc161360724)

[3.3. Обзор существующих моделей 23](#_Toc161360725)

[3.3. Принцип работы LLM 23](#_Toc161360726)

[3.4. Интеграция модели в движок 26](#_Toc161360727)

[3.5. Реализация общения с моделью 26](#_Toc161360728)

[4. Тестирование 28](#_Toc161360729)

[ЗАКЛЮЧЕНИЕ 30](#_Toc161360730)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 31](#_Toc161360731)

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

**UE5** – трёхмерный движок Unreal Engine 5.

**Blueprints (BP) –** это система визуального скриптинга Unreal Engine 5.

**AActor (actor)** – один из основных классов в UE5. Представляет собой любой объект, который может быть размещен на уровне.

**APawn** (пешка)– это класс, который является физическим и визуальным представлением того, чем управляет игрок или искусственный интеллект.

**ACharacter** (персонаж) – это наследник класса APawn, который, по умолчанию, имеет некоторые базовые функции движения. В отличие от пешек, у персонажей есть SkeletalMeshComponent – возможность включить расширенную анимацию, использующую скелет. К классам, производным от персонажа, можно добавлять другие скелеты, но это основной компонент, связанный с персонажем.

**USkeletalMeshComponent** –используются для всего, что имеет сложные анимации и использует скелет. Скелет (внешний вид) имеет внутри сложную конструкцию (взаимосвязанные кости), которая помогает перемещать отдельные вершины скелетной сетки в соответствии с текущей воспроизводимой анимацией.

**UStaticMeshComponent (SMC)** – компонент в UE5, позволяющий использовать Static Mesh.

**UInstancedStaticMeshComponent (ISMC)** – компонент в UE5, позволяющий использовать инстансинг.

**UHierarchicalInstancedStaticMeshComponent (HISMC)** – компонент в UE4, позволяющий использовать инстансинг. В отличие от ISMC, имеет поддержку LOD.

**LOD (Level of Detail, уровни детализации)** – техника, позволяющая подменять разные по детализации версии модели в зависимости от дистанции между камерой и объектом, либо в зависимости от процента площади экрана, занимаемой моделью.

**Полигон** – многоугольник, являющийся базовым компонентом 3D-сетки. Основные типы: треугольник (tri), четырёхугольник (quad) и n-gon (5 или более вершин).

**Polycount** – количество полигонов модели.

**Текстура** – изображение, накладываемое на поверхность 3D-модели. Может содержать различные свойства поверхности, например: цвет, жёсткость (roughness), смещение (displacement), направление нормалей (normal map), и т.д.

**CPU (Central Processing Unit)** – центральный процессор.

**GPU (Graphics Processing Unit)** – графический процессор (видеокарта).

**FPS (Frames per second)** – количество кадров в секунду.

**LLM** **(Large Language Model)** ­– большая языковая модель.

**Intelligent game agent (Игровой агент) –** это программное обеспечение или часть программного обеспечения, которое наблюдает за своей средой, разумно мыслит и действует в соответствии с этой средой.

**Token (токен)** **–** это специальные маркеры, которые используются для обозначения определенных элементов в тексте, например, начала и конца предложения, начала и конца абзаца, маркеров времени и т.д.

**Генеративный игровой агент** **–** это вид агентов, которые создаются на генеративных моделях для моделирования правдоподобного человеческого поведения, они демонстрируют симуляцию как индивидуального, так и группового поведения.

**AnimInstance –** это элемент, который используются для создания и управления сложным поведением анимации.

**AnimGraph** **– BP** элемент сграфом, используемый для выборки, смешивания и управления позами, которые будут применяться к скелетным сеткам с помощью Animation Blueprint (**AnimInstance BluePrint**).

ВВЕДЕНИЕ

LLM – это мощные модели искусственного интеллекта, которые используют возможности обработки естественного языка для эффективного понимания, создания и управления человеческим языком. Эти модели обучаются на огромных объемах текстовых данных, что позволяет им преуспеть в различных языковых задачах, таких как контекстно-зависимое понимание естественного языка, генерация текста, перевод и обобщение.

Данный проект является прототипом внедрения LLM модели внутри трёхмерного движка UE 5. В контексте компьютерных игр можно отметить, что создание ботов - модели программных компонентов для игровых агентов – использует разные подходы и методы. Например, конечно-автоматная модель, модель поведения с применением машинного обучения, структура с расписанием действий. В данной работе, в качестве модели поведения игрового агента используются текстовые запросы и ответы LLM модели.

Для реализации эффективного взаимодействия с LLM моделью необходимо выбрать нужную модель, правильно формировать вопрос и ответ к ней, реагировать на некорректное поведение, продумывать различные сценарии использования модели, исследовать предметную область и рынок, анализировать качество уже придуманных решений.

Для быстрого создания прототипов игр в Unreal Engine 5 существует визуальная система скриптинга – Blueprint. Система Blueprint основана на концепции использования интерфейса на основе узлов для создания элементов игрового процесса из редактора Unreal Editor. Как и многие распространенные скриптовые языки, Blueprint используется для определения объектно-ориентированных классов или объектов в движке. При запуске игры все Blueprint-скрипты переводятся на язык C++, ввиду чего игра на Blueprint может работать менее продуктивно, чем на C++. Система Blueprint чрезвычайно гибкая и мощная, поскольку дает дизайнерам возможность использовать практически весь спектр концепций и инструментов, обычно доступных только программистам. Кроме того, разметка для Blueprint, доступная в реализации Unreal Engine на C++, позволяет программистам создавать базовые системы, которые могут быть расширены дизайнерами [1].

В данном проекте рассматривается разработка элементов взаимодействия LLM модели с движком и пользовательским интерфейсом для игры жанра «Выживание».

Затрагивается и вопрос рациональности применения LLM моделей в UE и игровой индустрии, в частности.

1. Создание игровой системы
   1. Правила игры

**Симулятор выживания** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *survival sim* или [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *survival game*) — жанр [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), разновидность [симуляторов жизни](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D0%B6%D0%B8%D0%B7%D0%BD%D0%B8), в которых основной целью игрока является сохранение жизни виртуального персонажа на фоне множества угрожающих ему опасностей. Элементы выживания содержатся практически во всех компьютерных играх, но в симуляторах выживания эта задача выдвинута на передний план и является главной в игре.

В данном проекте игровой агент имеет следующие параметры:

* Здоровье (HP) – от очков здоровья зависит жив персонаж или нет. Здоровье медленно восстанавливается, если все остальные параметры в норме. Также быстро восстановить здоровье можно находясь в гнезде. Диапазон значений – от 0 до 100.
* Сытость (Satiety) – очки сытости от 0 до 150. Их можно восполнить, находясь на лесном блоке - значение food у области декрементируется, а игровой агент получает +15 к сытости. При значении сытости меньше 0, персонаж теряет здоровье.
* Усталость (Fatigue) – очки усталости от 0 до 100. Усталость равная 100 означает, что персонаж бодрствует. При 0 и меньше – игровой агент теряет здоровье. Усталость можно пополнить, находясь в гнезде.
* Температура (Temperature) – очки температуры. Максимальная температура равна 60 – это температура пустынного региона (блока). Минимальная температура равна -20.

Изначально персонаж начинает с температурой равной 35. В зависимости от местонахождения, игровой агент понижает или повышает свою температуру. Температура меняется по закону охлаждения Ньютона:

, где

означает конечную температуру агента через 1 сек., – начальную температуру персонажа, – температура блока, k – коэффициент теплообмена. В проекте он равен 0,19. При перегреве (больше 40), переохлаждении (меньше 14) персонаж теряет здоровье.

* Скорость – скорость перемещения персонажа. Она зависит от его местонахождения на игровом поле, например, в пустыне игровой агент будет замедлен. Этот параметр не влияет на здоровье и прочие значения.

Все данные в кратком виде передаются модели в первом текстовом запросе. Также ей передаётся список возможных функций игрового агента:

1. go\_near\_<имя\_игровой\_области> – находится ближайший от агента блок нужного типа. Затем персонаж начнёт к нему движение по прямой.
2. go\_random – персонаж выберет произвольную доступную точку на игровом поле и пойдёт к ней.
3. rest – игровой агент отправится к гнезду.
4. eat – персонаж начнёт охоту и поест. Для этого, предварительно, нужно находиться на лесном блоке.
5. go\_maxforest – персонаж отправится в лесную область с самым большим число пищи.

Как видно, данная игра мало похожа на другие игры жанра выживания, так как у агента отсутствуют прямые угрозы. Также у игры нет конечной цели или путей развития персонажа. Однако модель LLM может допустить ряд ошибок, неправильно отвечать на запросы, что приведёт к гибели управляемого персонажа.

* 1. Структура проекта

Файлы проекта имеют строгую иерархию. Все файлы с исходным кодом расположены в папке «Source», файлы конфигурации – в папке «Config», основной контент проекта и Blueprint-ы – в папке «Blueprints».

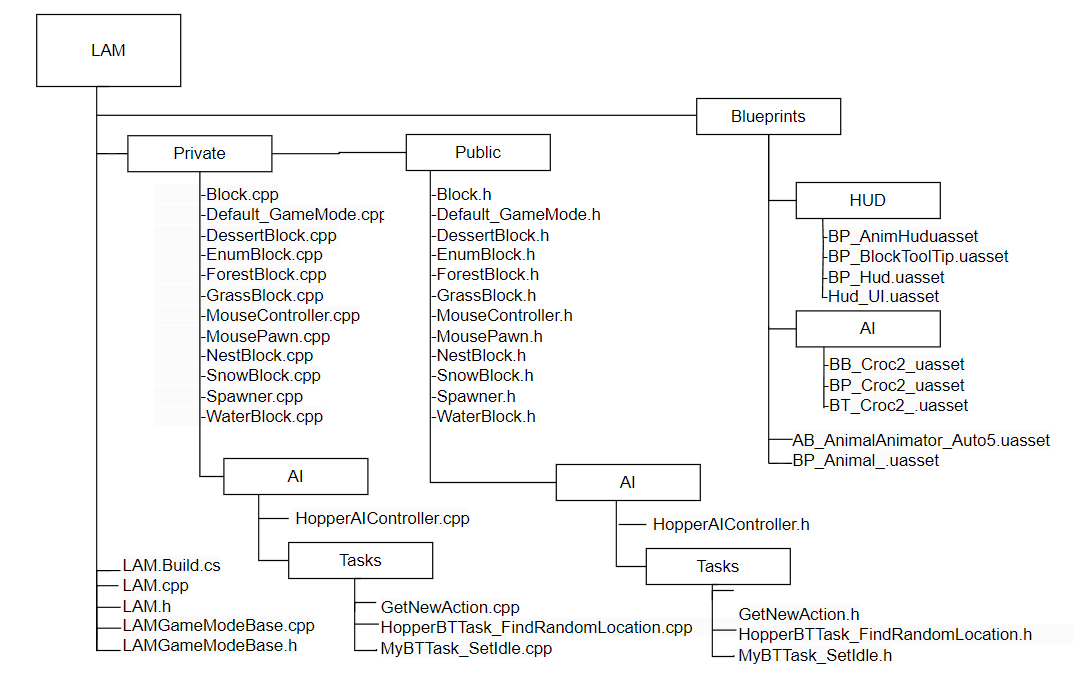
Иерархия файлов с исходным кодом изображена на рисунке 1:

Рисунок 1. Структура корневой папки проекта

В папке «Public» содержатся заголовочные файлы (.h), в «Private» – исходные (.cpp). Файлы в папке «Blueprints» содержат основные классы проекта, обязательные для его функционирования.

Файлы внутри папки «Blueprints» разделены по следующему принципу:

1. HUD: файлы с классами, отвечающими за интерфейс;
2. AI: файлы с классами, хранящие данные и дерево поведения ИИ;
3. Camera: файлы с классами, отвечающими за управление камерой;
4. Прочие файлы – класс игрового агента BP\_Animal и классAB\_AnimalAnimator типаAnimInstance с внутренними файлами типа AnimGraph.

Аналогичный подход применяется к разделению контента проекта. К ранее описанному добавляются папки, содержащие различные ассеты, например: «Maps» (карты, уровни), «Widgets» (UMG-виджеты), и т.д.

* 1. Импорт ассетов в Unreal Engine 5

Для визуального обогащения трехмерной сцены необходимы ассеты для игровых агентов и окружающей среды. Они были найдены в официальном магазине от Epic Games. Теперь их необходимо импортировать в движок. Для этого воспользуемся встроенной в Unreal Engine 5 утилитой. Вызвать ее можно через меню Add/Import в Content Browser’е (рис. 2):

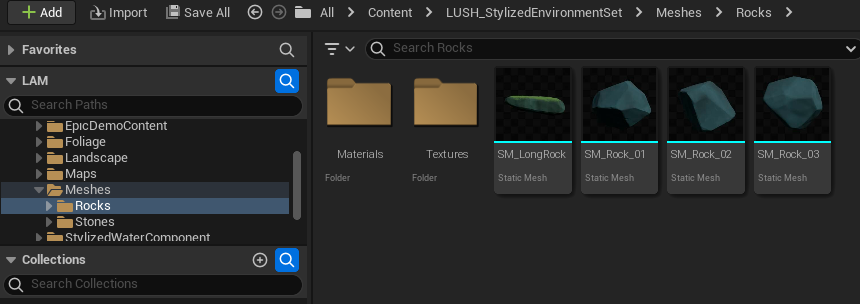


Рисунок 2. Вызов утилиты импорта ассетов в Unreal Engine 5

Далее необходимо выбрать все необходимые файлы. В данном случае нас интересуют текстуры (набор .png файлов), файлы Static Mesh-ей, а также файлы моделей (.fbx).

Результат после импорта изображен на рисунках 3,4:

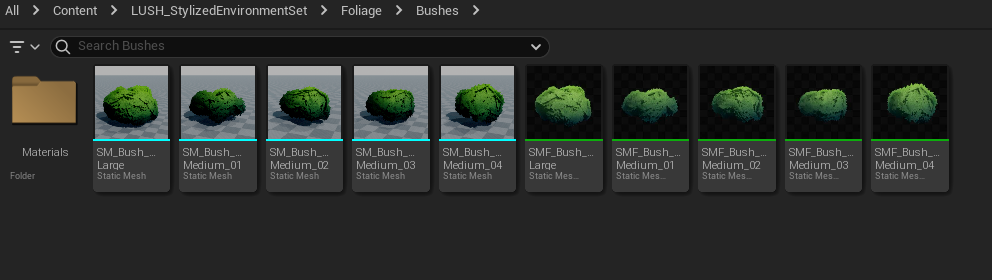


Рисунок 3. Ассеты для окружающего мира в UE Content Browser после импорта.

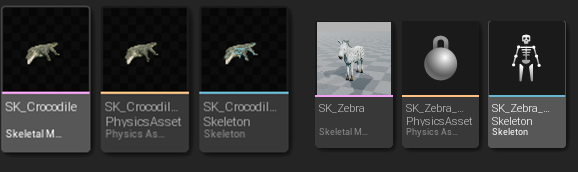


Рисунок 4. Ассеты игровых агентов в UE Content Browser после импорта

* 1. Моделирование игрового поля

За генерацию игрового поля отвечает функция GenerateMap() объекта класса Spawner. По заранее заготовленной двумерной строке WorldMap создаются блоки различных классов. Все mesh-ы блоков имеют тип ISMC.

Ниже даны эквиваленты символами строки карты:

“g” – зеленый блок травы класса AGrass.

“w” – синий блок воды класса AWater.

“f” – зеленый блок с лесом класса AForest. Создаётся вместе с моделями деревьев на нём. Высота дерева зависит от числа еды.

“d” – желтый блок пустыни класса ADesert.

“s” – белый блок снега класса ASnow.

“n” – оранжевый блок гнезда класса ANest.

Размеры карты настраиваемые, как и строка отвечающая за генерацию карты. Можно генерировать карты разных форм, если вместо на определенной позиции в строке стоит неразрешенный символ. Тогда куб не поместится в сцену на нужной позиции.

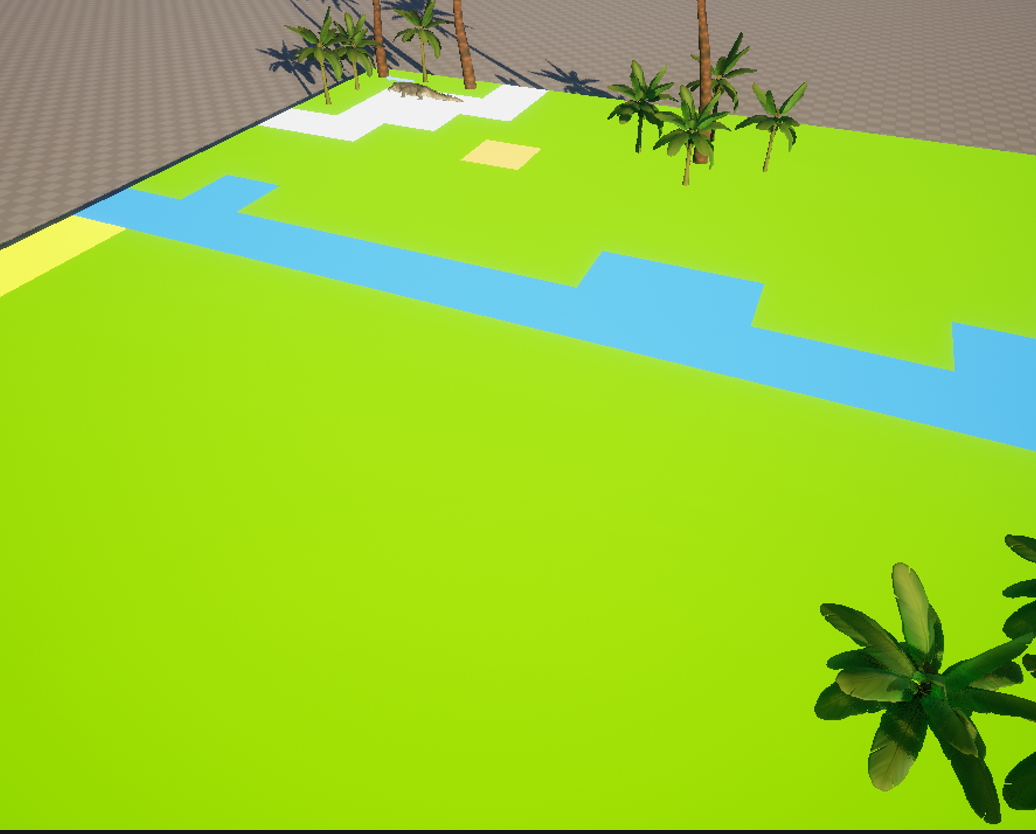


Рисунок 5. Модель карты с различными блоками

* 1. Настройка анимации у моделей

МНОГО ТЕКСТА И КАРТИНОК

1. Разработка внутриигровых систем
   1. Описание классов

Рассмотрим классы, созданные в рамках проекта.

Классы, размещаемые на 3D сцене (производные от AActor):

1. ASpawner – класс, генерирующий игровое поле из кубов. Основными его целями являются размещение различных игровых областей в зависимости от символа из текстовой карты, а также ожидание ответа от LLM модели. Текстовая карта представляет собой многострочную строку с определёнными символами, обозначающие типы игровых территорий. Объект не имеет модели и существует в единственном экземпляре на сцене. Функция ожидания ответа состоит из таймера на 20 сек., который запускается каждый раз, когда модель выдаёт новый токен. После того как модель сформирует ответ, таймер завершится – а полученный текст отправится в объект класса ABP\_Animal;
2. ABP\_Animal – класс генеративного игрового агента. Содержит в себе модель, капсулу (capsule component) , скелет (skeletal mesh), анимацию движения, функции передвижения, ИИ контроллер (AIController). Капсула представляет собой сферу, отвечающую за коллизию модели с землёй и другими объектами. Внутри капсулы также создан маленький параллелепипед, который будет соприкасаться с игровым блоком и отсылать информацию агенту о том, в какой территории он сейчас находится. Анимация движения (AB\_AnimalAnimator), в свою очередь, контролирует тип перемещения агента (ходьба или плавание) в зависимости от информации от капсулы. Также этот компонент обеспечивает плавность анимации передвижения и поворота в зависимости от скорости агента. Данный подобъект несёт исключительно декоративную функцию;
3. ABlock – класс, производный от ISMC. Представляет собой блок, по умолчанию область с травой, от которого наследуются все остальные классы блоков. Блок имеет параметры max\_speed, max\_temperature – первый отвечает за максимально допустимую скорость игрового агента, второй за его температуру воздуха. Далее описаны все типы блоков.

Классы, производные от ABlock:

1. AGrassBlock – класс-наследник от ABlock. Это обычная область со степью. Не имеет штрафов к максимальной скорости, температура чуть ниже начального значения персонажа;
2. AForestBlock – класс-наследник от ABlock. Это область с лесом. Не имеет штрафов к максимальной скорости, температура такая же как у области с травой. Содержит в себе целочисленную переменную food, сгенерированную заранее со значениями от 1 до 15. От этой переменной зависит модель дерева, которая расположена на этом блоке. Дерево несёт декоративную функцию, также позволяет визуально отличить блоки травы от блоков с лесом. Игровой агент, находясь на этом блоке, может пополнить запас пищи – значение food декрементируется;
3. ADesertBlock – класс-наследник от ABlock. Это пустынная область. Температура гораздо выше начальной температуры игрового агента. Также имеет штраф к максимальной скорости. Является основным препятствием, находясь на ней агент может терять здоровье из-за перегрева;
4. ASnowBlock – класс-наследник от ABlock. Это область заснеженная область. Не имеет штраф к скорости. Имеет минусовую температуру, игровой агент может терять здоровье из-за переохлаждения;
5. AWaterBlock – класс-наследник от ABlock. Это область с водой. Персонаж, находясь в этой области, начинает плавать. Скорость плавания чуть ниже максимальной скорости ходьбы;
6. ANestBlock – класс-наследник от ABlock. Это гнездо для игрового агента. Находясь в нём, персонаж восстанавливает здоровье и усталость. Не имеет штрафов к скорости. Имеет температуру равную начальной температуре персонажа.

Классы, расширяющие функциональность игрока:

1. BP\_HUD – класс интерфейса. Содержит в себе объект типа UI и компонент LLM модели. Обеспечивает основной бизнес процесс по взаимодействию с нейросетью.
2. BP\_AnimHud – класс UMG Widget, позволяет в режиме реального времени выводить в правом верхнем углу состояние игрового агента ( его показатели).
3. BP\_BlockToolTip – класс UMG Widget, появляется при нажатии левой кнопкой мыши по игровому блоку. Позволяет в режиме реального времени выводить имя, температуру и др. показатели куба.
4. Hud\_UI - класс UI. Содержит в себе scroll поле с textbox-ом, отображающим все входные и выходные текстовые запросы в виде чата. Под scroll box-ом находится поле text input, которое тоже позволяет корректировать запросы модели вручную, по желанию.
5. AI\_Anim\_Controller – класс типа AIController. Взаимодействует с объектами типа BlackBoard и BehaviourTree. BlackBoard – хранилище данных для игрового агента, управляющее поведением в BehaviourTree. BlackBoard содержит vector TargetLocation – координаты конечной точки маршрута, string Action – выбранное действие персонажа, actor SelfActor – указатель на персонажа BP\_Anim. Сам контроллер имеет в себе строковый массив Actions, с очередью действий. BehaviourTree – это компонент, позволяющий создавать дерево поведения персонажа (см. рис. 2).

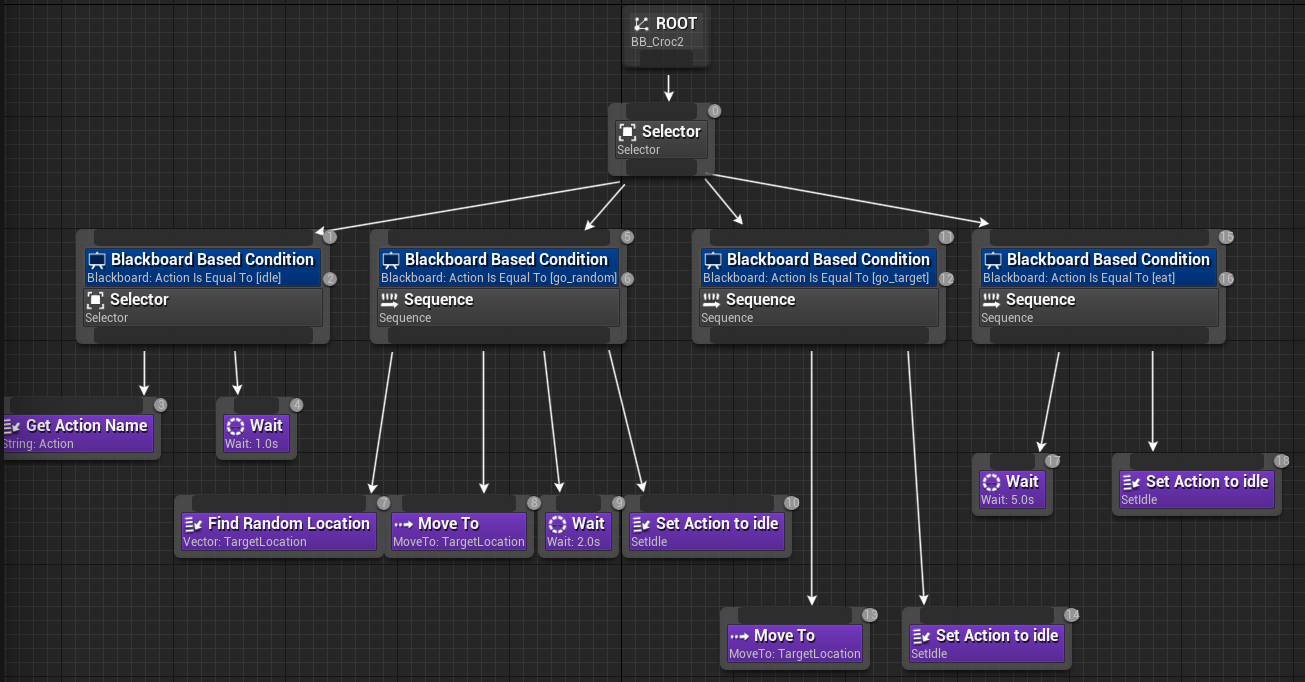


Рисунок 3. Дерево поведения игрового агента

Данное дерево помимо стандартных компонентов MoveTo и Wait имеет созданные узлы GetActionName, FindRandomLocation, SetActionToIdle. Стоит отметить, что большую часть времени агент находится в состоянии idle – и ждёт следующее действие. Было принято решение поставить на паузу всю игровую логику, пока модель формирует ответ на запрос с текущим состоянием персонажа. Параметры игрового агента, не меняются в состоянии idle.

* 1. Взаимодействие игрового агента с окружающим миром

Так как жанр игры «Выживание», игровой агент должен терять очки своих потребностей исходя из условий внешней среды. Ниже представлены функции, которые отвечают эти параметры.

* + 1. Функция CalcTemp

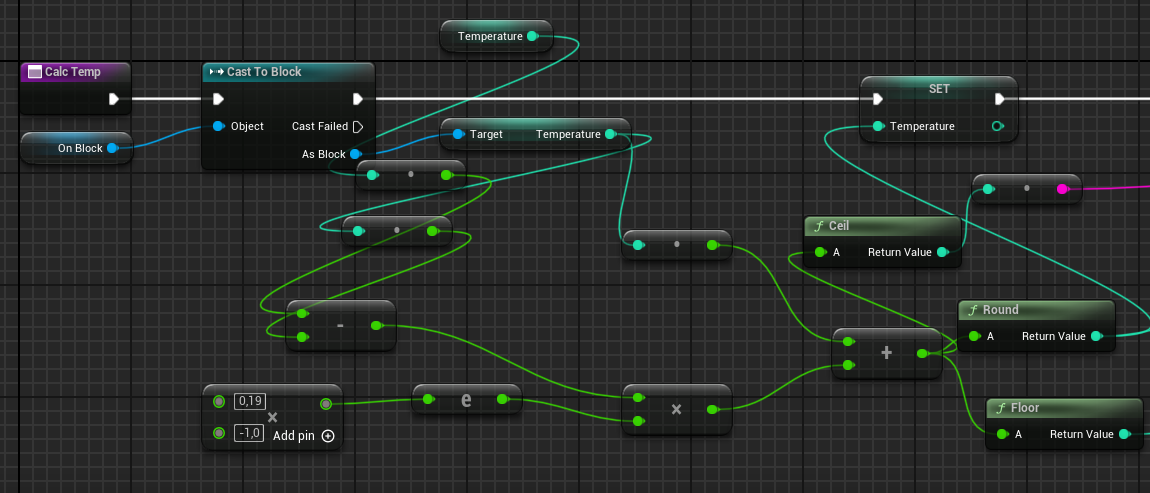


Рисунок 6. BP функции CalcTemp

Данная функция считает температуру игрового агента через 1 игровой тик по закону охлаждения Ньютона:

, где

означает конечную температуру агента через 1 сек., – начальную температуру персонажа, – температура блока, k – коэффициент теплообмена. В проекте он равен 0,19.

* + 1. Функция CalcFatigue

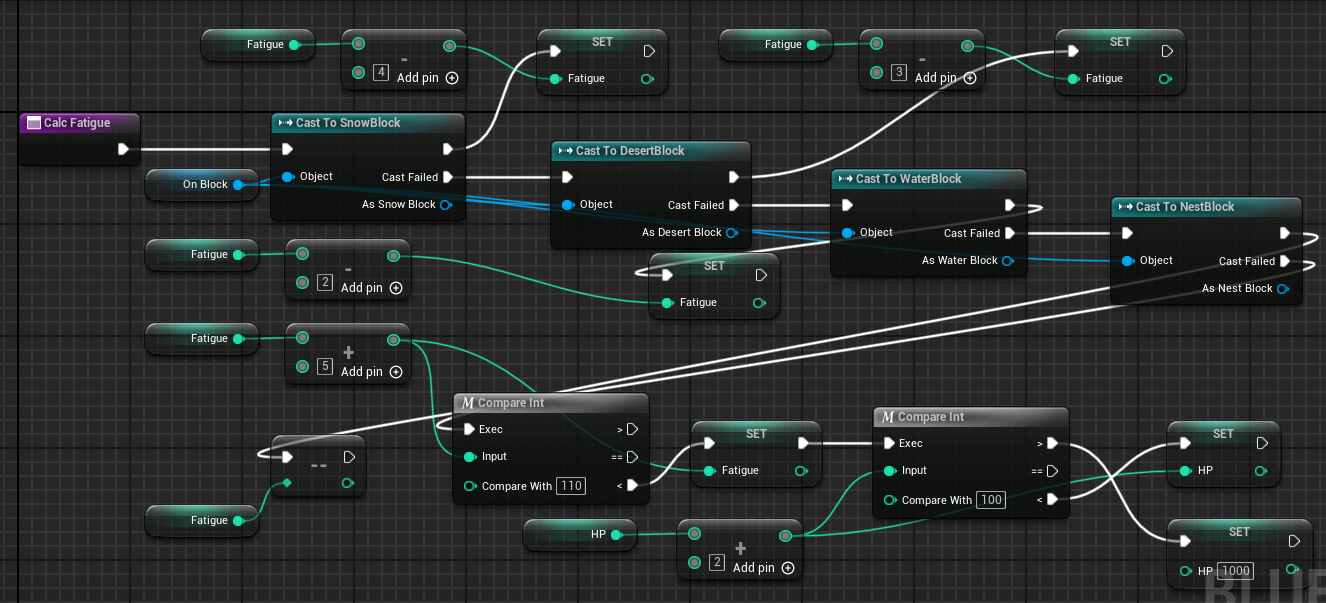


Рисунок 7. BP функции CalcFatigue

Данная функция вычитает из параметра усталости игрового агента значение каждый игровой тик. В зависимости от типа блока, на котором стоит агент (Вода, Снег, Пустыня, Гнездо) меняется вычитаемая/прибавляемая величина. Также эта функция из-за обращения к типу блока Гнездо (Nest), может инкрементировать значение здоровья персонажа. Сделано в целях минимизации Cast-ов.

* + 1. Функция CalcHP

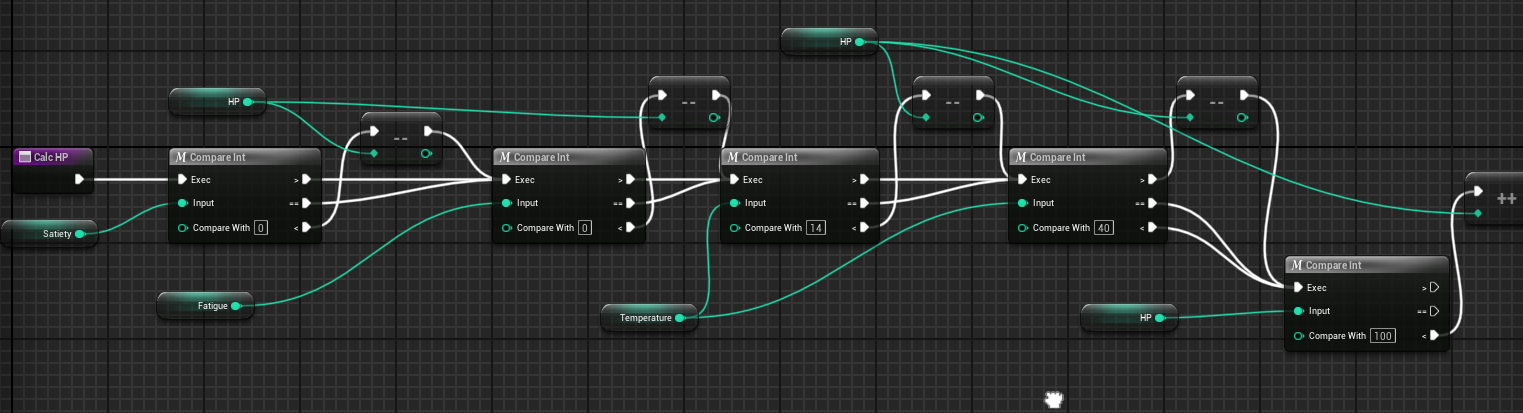


Рисунок 8. BP функции CalcHP

Данная функция исходя из значения всех трёх величин потребностей игрового агента, а именно (Усталость, Сытость, Температура) изменяет значение здоровья каждый игровой тик. Стоит отметить, что игровой персонаж регенерирует 1 очко здоровья вне гнезда всегда.

* 1. Создание интерфейса

Для удобства тестирования проекта и улучшения понимания игровой ситуации у конечного пользователя были разработаны несколько оконных интерфейсов.

МНОГО ТЕКСТА И КАРТИНОК

* 1. Реализация камеры

Для удобства тестирования проекта и улучшения визуальной составляющей проекта была создана подвижная камера. Перемещение её осуществляется при помощи клавиш клавиатуры W,A,S,D; поворот против часовой стрелки на Q, по часовой – на E. Колесо мыши отвечает за приближение/отдалении камеры. При нажатии на X камера перемещается на начальное положение, поворот и приближение сбрасывается до значений по умолчанию. Привязки на нажатия клавиш можно поменять в файлах типа Data Asset (рисунок 14).

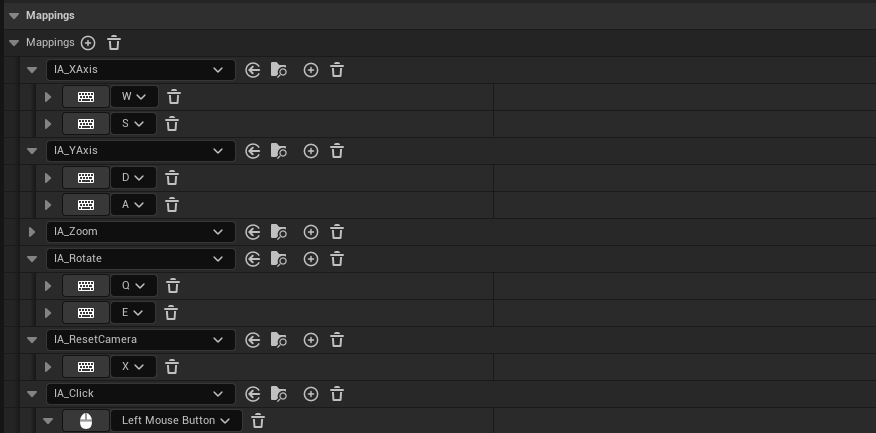


Рисунок 14. Изменение конфигурации управления камерой.

Также в функционал камеры добавлена возможность левой кнопкой мыши выделять цветные блоки из трёхмерной сцены. Результат данного действия описан в предыдущем пункте про интерфейс.

За программную реализацию камеры отвечает класс ARTSPawn. Для удобного изменения параметров внутри UE используются производные от C++ классов компоненты BP\_CustomCamera, BP\_PlayerController, BP\_GameModeBase. В последнем из них выбирается нужный класс игровой сессии, интерфейса игрока, контроллера игрока, класс камеры по умолчанию и другие (рисунок 15).

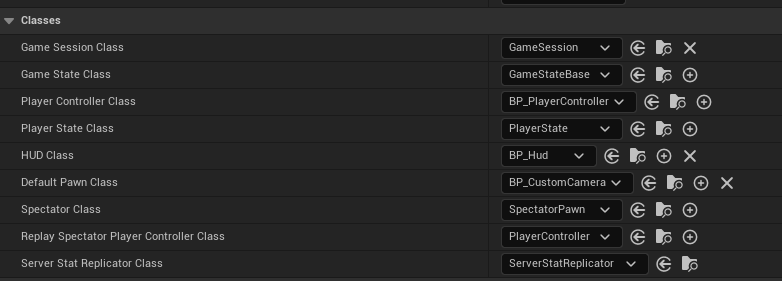


Рисунок 15. Настройки класса GameModeBase.

1. Разработка системы взаимодействия с LLM моделью
   1. Обзор существующих моделей

В настоящее время доступно несколько LLM, и новые разрабатываются по мере роста спроса на возможности обработки естественного языка. Некоторые популярные примеры включают в себя: [12]

* GPT-4 от OpenAI
* BERT от Google
* Gemma от Google
* RoBERTa от Facebook
* Microsoft Turing-NLG
* LLaMA (Large Language Model Meta AI)
* Mistral (Mistral AI)

Некоторые из моделей требуют постоянное подключение к интернету. Исходя из потребностей проекта, принято решение скачать ряд моделей, а именно: LLaMa2-7b, Gemma-7b, LLaMa3-8b.

LLaMa2-7b - это простая модель, настройки которой позволяют задавать системный контекст (в нашем случае – это роль игрока, управляющего игровым агентом). Gemma-7b – открытая модель от компании Google, разработанная на базе закрытой Gemini. Она более «тяжёлая», т.к. она обучена на большем числе параметров. LLaMa3-8b, как утверждают разработчики: «Самая мощная из открытых LLM на сегодняшний день».

Все модели умеют общаться в json формате и других грамматиках. В данной работе в общении предусмотрен формат json для повышения качества ответа у LLM модели. Стоит отметить, что на этапе создания проекта был найден единственный доступный плагин в UE5 с офлайн моделью llama. Однако практически любую модель в офлайн и онлайн версиях, работающую с API, можно интегрировать в UE.

* 1. Принцип работы LLM

LLM Модели работают с помощью текстовых запросов. Принцип работы состоит в следующем:

1. на вход модели подается контекст (любой текст);
2. модель анализирует весь контекст и генерирует всего один токен (токен - слово, часть слова или символ);
3. модель добавляет токен к контексту и передает новый контекст в пункт 1, до тех пор, пока не будет получен ответ.

Языковая модель имеет ограничение на суммарное количество токенов, которое мы можем передать на вход и получить на выходе. В основном, это зависит от размера модели. Токен это слово или часть слова, символ и тд. 1 токен не всегда равен 1 слову. Для английского языка 1000 токенов в среднем равны 750 словам. Для русского языка 1000 токенов это всего около 375 слов. [8]

Также стоит отметить различные параметры LLM моделей:

1. **Temperature** (Температура). Чем ниже температура, тем более детерминированными являются результаты в том смысле, что всегда выбирается наиболее вероятный следующий токен. Повышение температуры может привести к большей случайности, что будет способствовать более разнообразным и творческим результатам. По сути, вы увеличиваете вес других возможных токенов. С точки зрения применения вы можете использовать более низкое значение температуры для таких задач, как контроль качества на основе фактов, чтобы стимулировать более фактические и краткие ответы. Для создания стихотворений или других творческих задач может быть полезно увеличить значение температуры.
2. **Top P** — метод выборки с температурой, называемый выборкой ядра, при котором можно контролировать степень детерминированности модели. Если нужны точные и фактические ответы, данный параметры должен быть на низком уровне. Если требуются более разнообразные ответы, значение следует увеличить. Если в модели есть параметр Top P, это означает, что для ответов учитываются только токены, составляющие вероятностную массу top\_p, поэтому низкое значение top\_p выбирает наиболее уверенные ответы. Это означает, что высокое значение top\_p позволит модели просмотреть больше возможных слов, в том числе менее вероятных, что приведет к более разнообразным результатам. Общая рекомендация – изменить температуру или Top P, но не оба сразу.
3. **Max Length (**максимальная длина). Вы можете управлять количеством токенов, генерируемых моделью, регулируя максимальную длину. Указание максимальной длины помогает предотвратить длинные или ненужные ответы и контролировать затраты.
4. **Стоп-последовательности**. Это строка, которая останавливает модель от генерации токенов. Указание последовательности остановки — это еще один способ контролировать длину и структуру ответа модели. Например, вы можете указать модели генерировать списки, содержащие не более 10 элементов, добавив «11» в качестве стоп-последовательности.
5. **Frequency Penalty** (штраф за частоту). Штраф за частоту накладывает штраф на следующий токен, пропорциональный тому, сколько раз этот токен уже появлялся в ответе и подсказке. Чем выше штраф за частоту, тем меньше вероятность того, что слово появится снова. Этот параметр уменьшает повторение слов в ответе модели, назначая более высокий штраф за лексемы, которые кажутся более выраженными.
6. **Presence Penalty (**штраф за присутствие). Штраф за присутствие также применяет штраф к повторяющимся токенам, но, в отличие от штрафа за частоту, штраф одинаков для всех повторяющихся токенов. Токен, который появляется дважды, и токен, который появляется 10 раз, наказываются одинаково. Эта настройка не позволяет модели слишком часто повторять фразы в ответе. Если требуется, чтобы модель генерировала разнообразный или креативный текст, можно использовать более высокий штраф за присутствие. Чтобы модель оставалась детерминированной, используется более низкий штраф за присутствие.
   1. Интеграция модели в движок

Для внесения модели внутри UE 5 используется плагин UELlama. Он включает в себя Blueprint компонент с моделью, который находиться в классе BP\_HUD. Компонент имеет следующий функционал – activate(), deactivate(), addprompt(). Первые 2 ф-ии включают/выключают llama 2-7b. Последняя – отправляет модели prompt (текстовый запрос). Чтобы модель корректно отвечала на сообщения, а не дописывала их сама – сообщение оборачивается в следующую конструкцию [INST]…[/INST].

Важным моментом является, то что модель не сразу выдаёт полный ответ, а генерирует токены (слова и др. символы) последовательно. В следующем пункте описан процесс общения с моделью при таком режиме работы.

* 1. Реализация общения с моделью

Как было сказано ранее модель отвечает постепенно, также существует задержка после получения входных текстовых данных. Чтобы дождаться полного ответа, в классе Spawner реализована ф-ия GetAnswer(). Внутри неё реализован 20-ти секундный таймер класса FTimerDelegate. Если текст ответа модели изменится, то таймер запустится снова. По окончанию работы, Spawner изменит свою переменную ready на значение true, которую впоследствии считает объект BP\_HUD.

В конструкторе класса BP\_HUD инициализируются режим ввода (Input mode), контроллер игрока (Player Controller, который обеспечивает обработку нажатий кнопок), объект HUD\_UI и его производные, а также сам игровой агент – с помощью функции GetActorOfClass(), которая вёрнет первого актёра заданного класса. После инициализации включается сама модель llama.

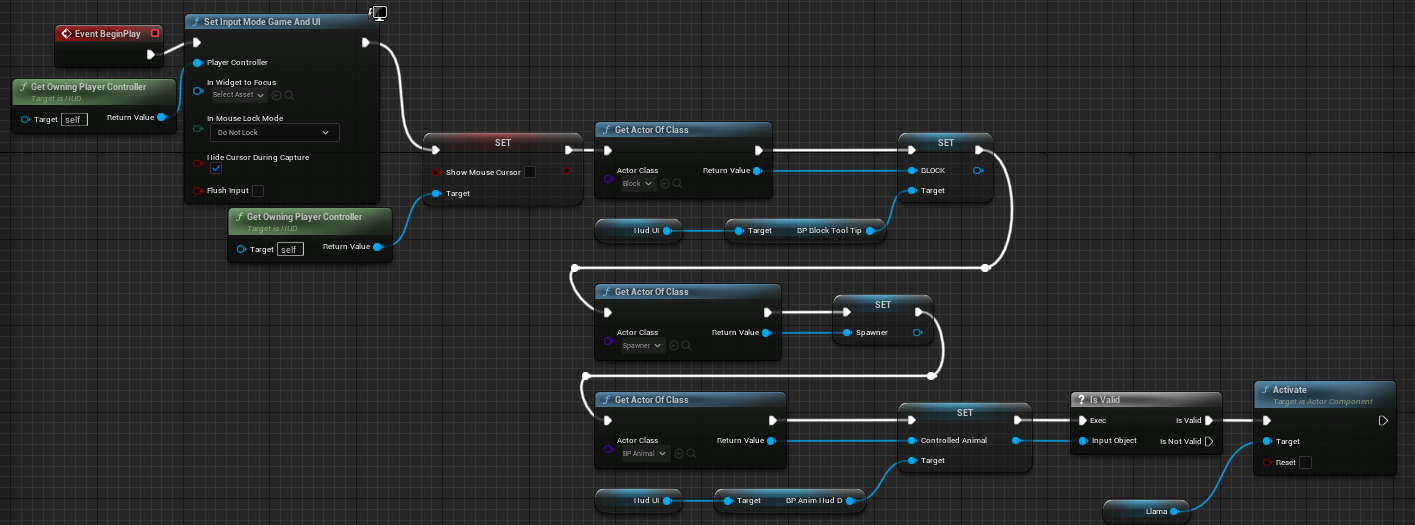


Рисунок 12. Конструктор BP\_HUD

Внутри класса игрового агента BP\_Animal первым делом, вызывается функция PresentGame(), принадлежащая Spawner. Она возвращает строку, в которой описаны приветствие игрока и введение его в правила игры. После этого ожидается ответ от модели. При его получении, входные данные фильтруются и извлекается первое вхождение подстроки, означающую следующее действие игрового агента.

Если действия не указаны, то будет послан ещё один запрос с текущими параметрами персонажа и просьбой сделать ход.

Если действие написано неверно, то персонаж отправится в случайную точку на поле.

Соответствующее действие добавляется в конец очереди Action у объекта класса AIController. Как только персонаж войдёт в состояние idle(бездействие), BP\_Animal отправит модели текущие параметры персонажи и просьбу сделать ход.

1. Тестирование

Проект запускался на ПК со следующими характеристиками:

Процессор Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 3.30 GHz

Оперативная память 16,0 ГБ (доступно: 15,2 ГБ)

Видеокарта NVIDIA GeForce GTX 1050 Ti

Тип системы 64-разрядная операционная система, процессор x64

В процессе активации модели возникали проблемы с нехваткой видеопамяти, при этом диспетчер ресурсов показывал, что видеокарта почти не используется.

Возможными причинами являются:

1. Некорректная компиляция исходных файлов .cpp плагина UELlama.
2. Недостаточно тонкая настройка среды UE 5.
3. Недостаточные мощности ПК.

Стоит отметить, что автор плагина, запуская модель в своём демонстрационном видеоролике, не испытывал проблем с моделью. При этом она довольно-таки быстро отвечала на запросы. Однако модель всё равно генерирует ответ последовательно и с задержками.

В результате можно сказать, что применение LLM в играх может быть обусловлено жанром разрабатываемого проекта. По понятным причинам, модель LLM не подойдут для интеграции с активными играми (например, FPS жанра). Также нежелательно использование таких моделей в играх с реальным течением игрового времени. LLM является универсальным инструментом, который, в свою очередь, нужно правильно настраивать и корректировать, чтобы не получать на выходе неадекватные ответы и действия. Модели можно дообучать на больших данных для конкретизации её назначения. Для LLM можно настроить хорошее воображение и креативность, чтобы выдавать необычное поведение для игровых агентов. Однако любая модель имеет лимит на память предыдущих токенов, поэтому ей периодически придётся напоминать о правилах игры, предыдущих действиях и т.п.

Стоит упомянуть, что LLM на данный момент чаще используют для генерации реплик, картинок, текстов, уровней; нежели чем для более строгих и математически обусловленных операций.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были получены навыки работы с Unreal Engine 5, получен опыт разработки на C++. Освоены навыки 3D-моделирования, анимирования, создания дерева поведения. Изучены способы взаимодействия, настройки и интеграции модели LLM в движок Unreal Engine 5.

В работе описан принцип работы LLM моделей, способы и особенности взаимодействия с ними. Рассмотрены популярные модели LLM, которые можно использовать для генерации управляющих команд игровых агентов. Изучены возможности, достоинства, недостатки применения LLM в игровой индустрии.

В процессе работы выполнены следующие задачи:

1. Создание игровой модели из ISMC кубов;
2. Создание персонажа с различными характеристиками;
3. Интеграция LLM модели llama-2-7b в движок;
4. Работа с анимацией и BlendSpacing-ом;

Итогом выпускной квалификационной работы стал проект, симулирующий деятельность генеративных игровых агентов в разнообразном мире под контролем LLM модели. Такой же принцип взаимодействия с LLM можно использовать и в других, будущих работах.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Unreal Engine 5 Documentation [электронный ресурс] // Unreal Engine Documentation URL: <https://docs.unrealengine.com/>. Дата обращения: 11.03.2024.
2. Использование токенов в LLM [электронный ресурс] // URL: https://dzen.ru/a/ZBKqdO5TiGGPDjJN. Дата обращения: 05.03.2024.
3. Основа геймплея игры на C++ для Unreal Engine [электронный ресурс] // URL: https://habr.com/ru/articles/357976/. Дата обращения: 11.03.2024.
4. Настройка параметров БЯМ [электронный ресурс] // URL: <https://obnimorda.ru/guides/prompt/introduction/settings/>. Дата обращения: 11.03.2024.
5. Display aspect ratio [электронный ресурс] // Wikipedia, the free encyclopedia URL: <https://en.wikipedia.org/wiki/Display_aspect_ratio/>. Дата обращения: 05.02.2024.
6. Geometry instancing [электронный ресурс] // Wikipedia, the free encyclopedia URL: <https://en.wikipedia.org/wiki/Geometry_instancing>. Дата обращения: 01.02.2024.
7. This dude put LLaMA 2 inside UE5 [электронный ресурс] // URL: https://www.youtube.com/watch?v=j\_r5xWm3Xl8&ab\_channel=Mika. Дата обращения: 01.03.2024.
8. UELlama [электронный ресурс] // URL: https://github.com/mika314/UELlama/tree/main. Дата обращения: 01.02.2024.
9. Using Large-Language Models (LLM) in Game Development [электронный ресурс] // URL: <https://gamedevacademy.org/using-large-language-models-llm-in-game-development-tutorial-list/#How_Can_Large_Language_Models_Be_Used_for_Games> Дата обращения 12.03.2024.
10. Generative Agents: Interactive Simulacra of Human Behavior [электронный ресурс] // URL: <https://arxiv.org/abs/2304.03442> Дата обращения 11.11.2023.
11. Инстансинг [электронный ресурс] // Habr: интернет-портал. URL: <https://habr.com/ru/post/352962/>. Дата обращения: 03.04.2024.