|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***«Методы взаимодействия с LLM для получения управляющих команд для игровых агентов в трехмерном движке Unreal Engine 5»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-74Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Больных А.С.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель курсового проекта | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

*2023 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

А.П. Карпенко

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_Модели и методы анализа проектных решений\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_РК6-74Б\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Больных Андрей Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсового проекта Методы взаимодействия с LLM для получения управляющих команд для игровых агентов в трехмерном движке Unreal Engine 5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КП (учебный, исследовательский, практический, производственный, др.) учебный\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения проекта: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

***Задание.*** Разработать игровую систему моделирующую взаимодействие LLM-модели и игрового агента (персонажа) с игровыми параметрами (здоровье, температура, усталость, сытость). Создать 3D-сцену с объектами, включающие себя различные параметры. Управляемый агент должен взаимодействовать с окружающей средой, удовлетворяя свои игровые потребности. Разработать механизм приёма/передачи сообщений между LLM-моделью и игровым агентом/окружающим миром.

***Оформление курсового проекта:***

Расчетно-пояснительная записка на 32 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

3 графических листа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «23» октября 2023 г.

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

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

АННОТАЦИЯ

В данной работе рассмотрены подходы для создания генеративных игровых агентов. Описана разработка системы взаимодействия с моделью LLM для получения управляющих команд. Рассмотрено создание игровой системы с окружающей средой, включающую в себя различные параметры, в которой игровой агент будет осуществлять свою деятельность. Также описаны этапы разработки основных компонентов проекта для дальнейшего использования в движке. Проведен анализ эффективности использования LLM для управления игровыми агентами.

В расчётно-пояснительной записке 32 страницы, 21 рисунок, 3 графических листа.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 3](#_Toc154693602)

[ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ 5](#_Toc154693603)

[ВВЕДЕНИЕ 7](#_Toc154693604)

[1. Структура проекта 8](#_Toc154693605)

[2. Описание классов 10](#_Toc154693606)

[3. Разработка системы взаимодействия с LLM моделью 14](#_Toc154693607)

[3.1. Обзор существующих моделей 14](#_Toc154693608)

[3.2. Интеграция модели в движок 14](#_Toc154693609)

[3.3. Реализация общения с моделью 15](#_Toc154693610)

[4. Правила игры 17](#_Toc154693611)

[5. Моделирование игрового поля 19](#_Toc154693612)

[6. Тестирование 20](#_Toc154693613)

[ЗАКЛЮЧЕНИЕ 22](#_Toc154693614)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 23](#_Toc154693615)

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.

**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 экспериментом по внедрению 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.

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

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

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».

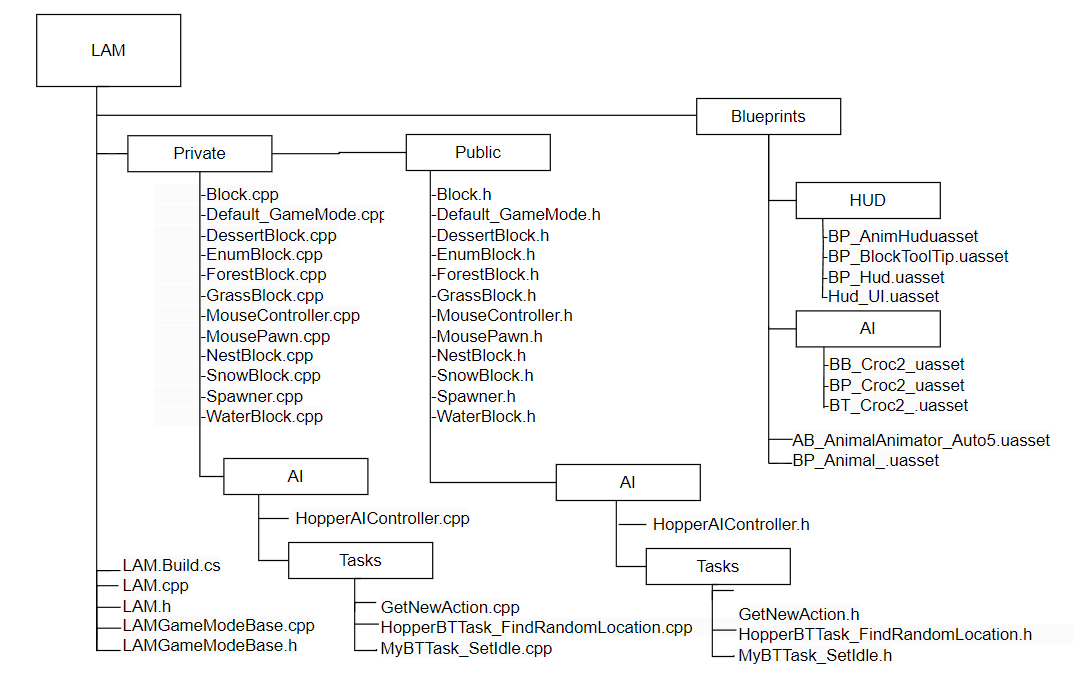
Иерархия файлов с исходным кодом изображена на рисунке 1:

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. 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. 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).

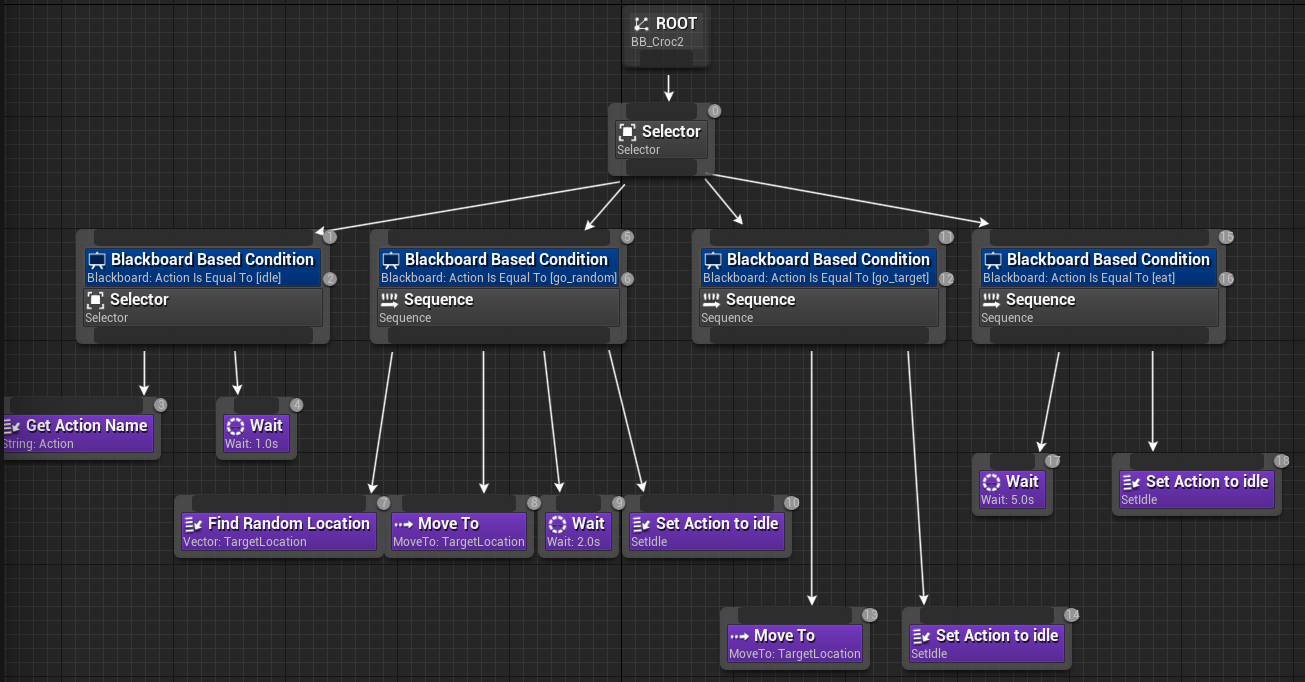


Рисунок 3. Дерево поведения игрового агента

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. Разработка системы взаимодействия с LLM моделью

3.1. 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-3 от OpenAI
* BERT 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 модель LLaMa 2-7b. Это простая модель, настройки которой позволяют задавать системный контекст (в нашем случае – это роль игрока, управляющего игровым агентом). Как и другие модели, llama позволяет общаться в json формате и других грамматиках. В данной работе json формат не предусмотрен, так как модель адекватно отвечает на текстовые запросы. Стоит отметить, что на этапе создания проекта был найден единственный доступный плагин в UE 5 с оффлайн моделью llama.

3.2. 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.

3.3. 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.

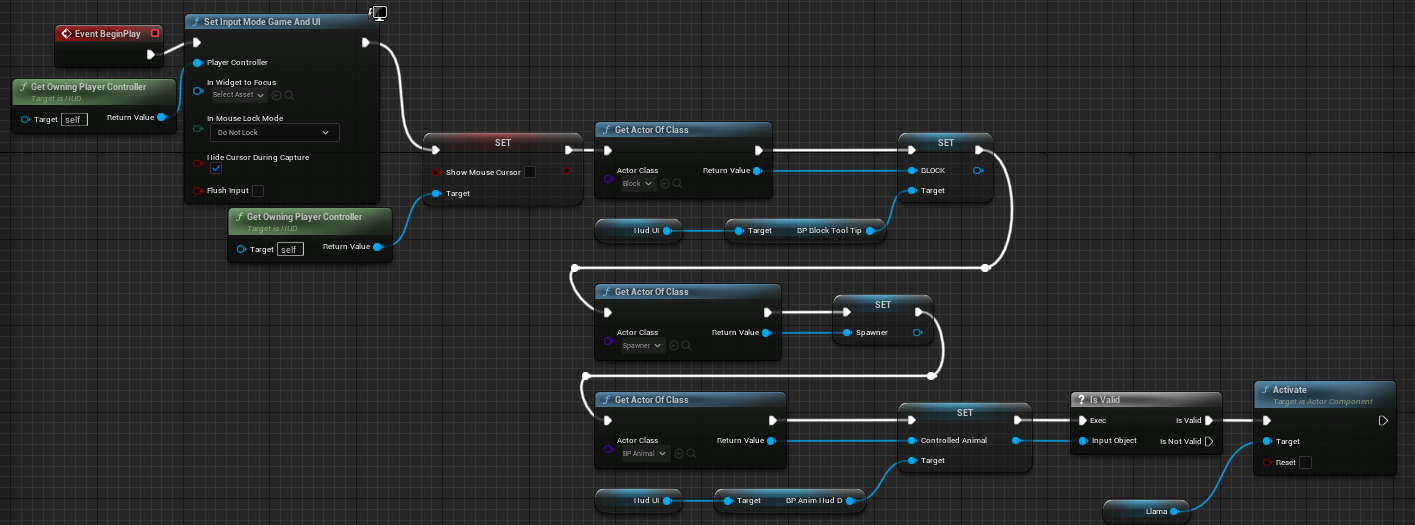


Рисунок 3. Конструктор BP\_HUD

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. Правила игры

Исходя из режима общения с LLM моделью, удобно спроектировать игру жанра «Выживание» с пошаговой системой. Ниже представлено описание жанра.

**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 значение 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. 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.

“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.

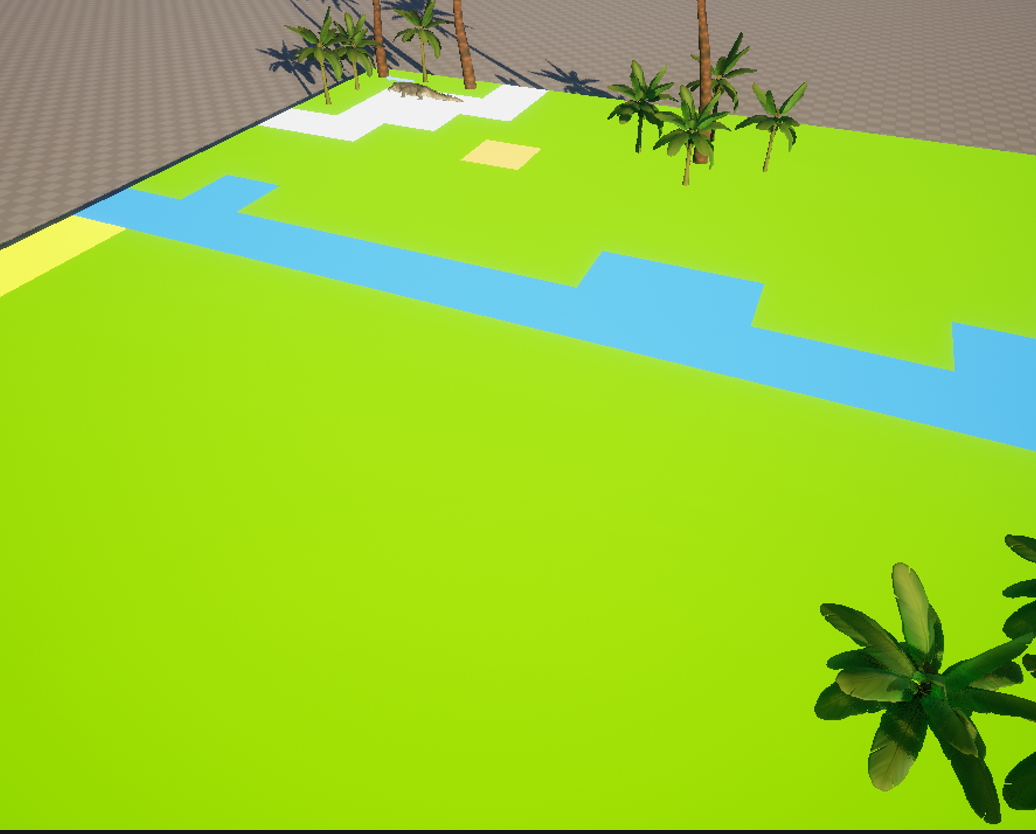


Рисунок 4. Модель карты с различными блоками

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. Неправильная компиляция исходных файлов .cpp плагина UELlama.
2. Неправильная 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 базовые навыки 3D-моделирования, анимирования, создания дерева поведения. Изучены способы взаимодействия, настройки и интеграции модели LLM в движок Unreal Engine 5.

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;

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Unreal Engine 5 Documentation // Unreal Engine Documentation URL: <https://docs.unrealengine.com/>. Дата обращения: 11.11.2023.
2. Использование токенов в LLM // URL: https://dzen.ru/a/ZBKqdO5TiGGPDjJN. Дата обращения: 05.11.2023.
3. Основа геймплея игры на C++ для Unreal Engine // URL: https://habr.com/ru/articles/357976/. Дата обращения: 11.11.2023.
4. Display aspect ratio // Wikipedia, the free encyclopedia URL: <https://en.wikipedia.org/wiki/Display_aspect_ratio/>. Дата обращения: 05.12.2023.
5. Geometry instancing // Wikipedia, the free encyclopedia URL: <https://en.wikipedia.org/wiki/Geometry_instancing>. Дата обращения: 01.12.2023.
6. This dude put LLaMA 2 inside UE5 URL: https://www.youtube.com/watch?v=j\_r5xWm3Xl8&ab\_channel=Mika. Дата обращения: 01.12.2023.
7. UELlama URL: https://github.com/mika314/UELlama/tree/main. Дата обращения: 01.12.2023.
8. Using Large-Language Models (LLM) in Game Development // 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 12.11.2023.
9. Generative Agents: Interactive Simulacra of Human Behavior // URL: <https://arxiv.org/abs/2304.03442> Дата обращения 11.11.2023.