|  |  |
| --- | --- |
| 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 НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***«Методы взаимодействия с LLM для получения управляющих команд для игровых агентов»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-74Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Больных А.С.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **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 научно-исследовательской работы**

по теме: методы взаимодействия с LLM для получения управляющих команд для игровых агентов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_РК6-74Б\_\_\_

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

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) \_учебная\_\_\_

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

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

Техническое 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 научно-исследовательской работы:***

Расчетно-пояснительная записка на 17 листах формата А4.

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

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

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

Дата выдачи задания «23» октября 2023 г.

|  |  |  |
| --- | --- | --- |
| **Руководитель НИР** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **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

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc152976178)

[ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ 6](#_Toc152976179)

[1. Большие языковые модели 8](#_Toc152976180)

[2. Применение LLM в играх 10](#_Toc152976181)

[2.1.Генерация текстов и реплик для неигровых персонажей 10](#_Toc152976182)

[2.2.Генерация игровых уровней. 10](#_Toc152976183)

[3.1.Создание игровых агентов. 11](#_Toc152976184)

[ЗАКЛЮЧЕНИЕ 15](#_Toc152976185)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc152976186)

ВВЕДЕНИЕ

Цели НИРС:

1. Изучить методы взаимодействия с LLM для получения управляющих команд для симуляции ИИ "личности"
2. Найти и рассмотреть иные перспективные AI-модели для ИИ в играх.
3. Изучить другие сферы применения AI-моделей в играх.

С момента появления первых компьютерных игр возникла необходимость в развитии инструментов противостояния действиям игроков. Игровой процесс, заключающийся только в достижении игроком поставленных задач без каких-либо затруднений или конкуренции со стороны игровой программы, может быть скучным, линейным и однообразным, хотя и существуют игры, где вся сложность заключается в геометрии уровней.

Первые модели программных компонентов – ботов – появились вместе с первыми играми. Они были максимально просты и могли выполнять автоматически и/или алгоритмизированные по расписанию действия. Такие ограничения были продиктованы вычислительными мощностями компьютеров того времени, а кроме того, не слишком притязательные игроки, пока ещё не избалованные возможностями игрового искусственного интеллекта, не требовали большего.

Такая ситуация сохранялась с момента зарождения компьютерных игр в середине XX в. до начала XXI в. С течением времени модели поведения ботов в играх стали всё стремительнее совершенствоваться. Одновременно улучшалась и реалистичность компьютерной графики. Основные причины: рост количества компьютеров и иных устройств, используемых населением; совершенствование их технических характеристик; увеличение конкуренции между разработчиками компьютерных игр. [6]

В 2005 г. произошёл прорыв – вышла игра F.E.A.R. , боты в которой были способны демонстрировать очень реалистичное поведение, в основе которого лежала новая конечно-автоматная модель. С тех пор игровая индустрия сильно изменилась, но необходимость в создании более совершенных моделей поведения неигровых персонажей (англ. Non-playable character (NPC) – персонаж, управляемый компьютером) сохраняется. Ведущим направлением на данный момент является применение методов машинного обучения (ММО) для получения наиболее реалистичной модели поведения интеллектуальных агентов. [7]

Искусственный интеллект продолжает добиваться огромных успехов в различных отраслях, и одним из ключевых событий последних лет является рост больших языковых моделей (LLM). В данной работе рассмотрены различные LLM и их применение в управлении игровыми агентами.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

**UE5** – трёхмерный движок 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.

1. Большие языковые модели

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. [5]

LLM могут произвести революцию в игровом мире несколькими способами, в том числе:

* Создание динамичных и захватывающих диалоговых систем
* Создание уникального внутриигрового контента, такого как квесты, сюжетные линии или описания.
* Автоматизация дизайна и генерации игровых уровней
* Персонализация игрового процесса на основе поведения и предпочтений игрока.
* Усиление внутриигровых обучающих программ и вспомогательных систем.

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)

Все эти 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]

1. Применение LLM в играх
   1. Генерация текстов и реплик для неигровых персонажей

Одним из основных применений LLM в разработке игр является генерация текстов и реплик для неигровых персонажей (NPC).

Интеграция LLM в игры может изменить взаимодействия с NPC. Эти объекты ИИ можно обучать на огромных наборах данных, обеспечивая спонтанность и реализм. Игроки могут участвовать в сложных диалогах или стратегических беседах, развивая прочные отношения с NPC, которые влияют на их положение в игре. Модели на большом языке позволяют добиться беспрецедентного уровня погружения, поскольку каждый NPC может развить свою особую личность, которая расширяется и развивается по мере развития путешествия игрока. Неигровые персонажи могут представлять героев или злодеев, союзников или врагов, их моральные ориентиры и личные линии определяются индивидуальным влиянием каждого игрока. NPC, управляемые искусственным интеллектом, можно даже переносить в разные игровые вселенные, функционируя как API-интерфейсы Plug-and-Play. [5]

Ранее Replica Studios, голосовая платформа искусственного интеллекта для игр и фильмов, [представила умных NPC на базе искусственного интеллекта](https://venturebeat.com/games/replica-unveils-ai-powered-smart-npcs-for-unreal-engine/?ref=hackernoon.com) для платформы Unreal Engine. Эти умные NPC будут работать на базе OpenAI (разработчика ChatGPT) и библиотеки Replica, состоящей из более чем 120 голосов искусственного интеллекта, преобразующих текст в речь, причем эта функция будет доступна в виде облачного плагина. [9]

* 1. Генерация игровых уровней.

Также LLM может применяться для генерации игровых уровней. Уровни игры, с их сложными функциональными ограничениями и пространственные отношениями более чем в одном измерении, сильно отличаются от видов данных, которые LLM обычно видят во время обучения. Наборы данных игровых уровни также трудно найти, что потенциально обременяет возможности этих моделей, требующих больших объемов данных. LLM обрабатывают входные данные и генерируют выводные данные линейным образом. [5]

Уровни игры должны быть представлены как последовательность токенов для ввода в модель, и сгенерированные результаты должны быть переосмыслены как пространственные данные для того, чтобы его можно было использовать. Далее, токенизация переменной длины схемы, используемые современными LLM, означают, что два уровня один и тот же размер может быть представлен разным количеством базовых токенов. Поэтому поддержание регулярности и пространственных отношений (например, попытка разместить врага прямо под игроком в двух измерениях) требует большего, чем просто подсчитывать количество токенов. [5]

* 1. Создание игровых агентов.

Примером создания игровых агентов с помощью LLM является проект Smalville, принадлежащий учёным из Корнельского Университета. [10]

Генеративные агенты — это агенты, которых создаются на генеративных моделях для моделирования правдоподобного человеческого поведения, они демонстрируют симуляцию как индивидуального, так и группового поведения.

Генеративные агенты располагают широким спектром умозаключений о себе, других агентах и их окружении. Они создают ежедневные планы, отражающие их характеристики и опыт, реализуют эти планы, реагируют и меняют планы, когда это уместно; они реагируют, когда конечный пользователь меняет свой окружающей среды или командует ими на естественном языке. Например, генерирующие агенты выключают плиту, когда видят, что их завтрак сгорает; ждут возле ванной, если она занята, и останавливаться, чтобы поболтать, когда они встречают другого агента, с которым хотят поговорить.

Общество, полное порождающих агентов, характеризуется появлением социальной динамики, в которой формируются новые отношения, распространяется информация, и возникает координация между агентами. Чтобы использовать генеративные агенты, мы описываем архитектуру агента, который хранит, синтезирует и применяет соответствующие воспоминания для создания правдоподобное поведение с использованием большой языковой модели.

Архитектура включает в себя три основных компонента.

Первый — это поток памяти, модуль долговременной памяти, который записывает на естественном языке полный список опыта агента. Модель сочетает в себе релевантность, новизну и важность для выявления записи, необходимые для информирования о текущем поведении агента.

Второй — рефлексия, которая с течением времени синтезирует воспоминания в выводы более высокого уровня, позволяя агенту делать выводы. о себе и других, чтобы лучше управлять своим поведением.

Третий планирование, которое преобразует эти выводы и текущую ситуацию в планы действий высокого уровня, а затем рекурсивно в детальное поведение для действий и реакций. Эти размышления и планы передаются обратно в поток памяти, чтобы повлиять на действия агента. будущее поведение.

Эта архитектура предлагает использование в разных областях: от ролевых и социальных игр до виртуальных миров. В сценарии социальных ролевых игр (например, подготовка к собеседованию), пользователь может безопасно генерировать сложные, конфликтные разговоры.

В проекте использовалась LLM ChatGPT и общество из двадцати пяти агентов в игровой среде. Конечные пользователи могут наблюдать за этими агентами и взаимодействовать с ними.

Если конечный пользователь или разработчик хотел, чтобы в городе была размещена внутри игровая вечеринка, то потребовалось бы вручную писать сценарии поведения десятков персонажей. В проекте демонстрируется, что с помощью генеративных агентов достаточно просто сказать одному игровому агенту, что он хочет устроить вечеринку. Несмотря на множество потенциальных точек неудачи, организатор вечеринки должен не забывать приглашать других агентов на вечеринку, участники должны запомнить приглашение, те, кто помнит, должны решить приходить ли им, и многое другое… Агенты распространяют информацию о вечеринке, а затем появляются, причем один агент даже спрашивает другого о свидании на вечеринке. Это все создается из одного исходного предложения, созданного пользователем для единственного агента (см. рис. 1).

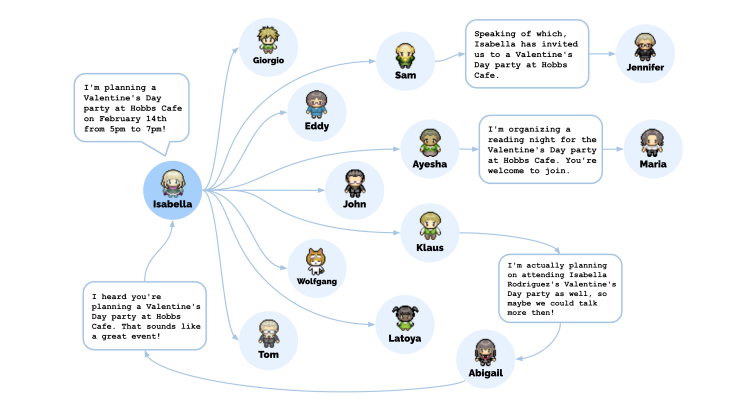


Рисунок 1. В распространении приглашения Изабеллы Родригес на вечеринку в честь Дня святого Валентина участвовало в общей сложности 12 агентов, не считая Изабеллу, которая услышала о вечеринке в кафе Хоббс к концу симуляции.

Генеративные агенты, как индивидуальные агенты, так и группы, стремятся воспроизводить правдоподобное поведение, основанное на окружающей среде и опыты. Правильная работа с отдельными агентами восстанавливает прошлый опыт и генерирует правдоподобные планы, реакции, и мысли, которые формируют их поведение. Сообщество из агентов демонстрируют распространение информации, формирование отношений, и координация агентов в разных частях общества.

В этом проекте были представлены генеративные агенты, интерактивные вычислительные агенты, моделирующие поведение человека. Их архитектура обеспечивает механизм хранения отчетов об опыте агента, углубляющие его понимание себя и окружающего через размышление, и получение компактного набора этой информации для информирования его действий.

Оценки предполагают, что данные генерирующие агенты могут играть роль во многих интерактивных приложениях, от инструментов проектирования до социальных вычислений систем для реалистичных сред. Однако, поведение агентов также нуждается в корректировке из-за некорректного распознавания объектов и текста. [11]

ЗАКЛЮЧЕНИЕ

Включение больших языковых моделей в разработку игр открывает мир возможностей для улучшения пользовательского опыта и создания динамического внутриигрового контента.

При этом стоит отметить следующие недостатки, такие как:

* ресурсоемкое потребление,
* медленный отклик и низкая производительность,
* сложная настройка,
* ограниченные знания из-за отсутствия подключения к интернету у некоторых моделей.

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 сферы применения LLM моделей в играх, такие как генерация реплик, текстов, картинок, звуков. Особенно подробно описано применение LLM для генерации игровых уровней.

Описаны некоторые виды игровых агентов. Исследован процесс создания игровых генеративных агентов на основе публикаций и проектов. Изучены дальнейшие перспективы использования таких агентов в играх. Рассмотрены преимущества и недостатки интеграции LLM в разработке игр.

СПИСОК ИСПОЛЬЗОВАННЫХ 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. What is Intelligent Game Agent // URL: <https://www.chidrestechtutorials.com/gaming/game-ai/intelligent-game>- agent.html. Дата обращения: 18.11.2023;
5. 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;
6. Моделирование поведения агентов для реализации игрового искусственного интеллекта / А. О. Анохин, Н. П. Садовникова, А. В. Катаев, Д. С. Парыгин // Прикаспийский журнал: управление и высокие технологии. – 2020. – № 2(50). – С. 85-99. – DOI 10.21672/2074-1707.2020.50.2.096-110. – EDN NXRNGX. Дата обращения 12.11.2023;
7. Seriya igr F.E.A.R. [F.E.A.R. series of games]. Fandom. // URL: <https://feargame.fandom.com/wiki/F.E.A.R_game_series> Дата обращения 12.11.2023;
8. ChatGPT: влияем на галлюцинации или как потопаешь, так и полопаешь // URL: [https://habr.com/ru/articles/727458/](https://habr.com/ru/articles/727458/%20) Дата обращения 11.11.2023;
9. NVIDIA поражает — NPC в играх теперь будут вести осмысленные диалоги и показывать естественную мимику // URL: <https://www.goha.ru/nvidia-porazhaet-npc-v-igrah-teper-budut-vesti-osmyslennye-dialogi-i-pokazyvat-estestvennuyu-mimiku-xLyezp> Дата обращения 12.11.2023;
10. Generative Agents: Interactive Simulacra of Human Behavior // URL: <https://arxiv.org/abs/2304.03442> Дата обращения 11.11.2023;
11. Communicative Agents for Software Development // URL: <https://arxiv.org/abs/2307.07924> Дата обращения 12.11.2023;
12. Большие языковые модели: полное руководство в 2023 году // URL: <https://research.aimultiple.com/large-language-models/> Дата обращения 10.11.2023;