**Плясунов Кирилл Андреевич**, магистрант,
ФГБОУ ВО «Поволжский государственный университет телекоммуникаций и информатики», г. Самара

**Гуленков Максимилиан Олегович**, магистрант,
ФГБОУ ВО «Поволжский государственный университет телекоммуникаций и информатики», г. Самара

# ИНТЕГРАЦИЯ НЕЙРОСЕТЕВОГО AI В UNREAL ENGINE

**Аннотация:** В данной работе рассматривается методика интеграции самописного нейросетевого искусственного интеллекта, разработанного на Python, в игровой движок Unreal Engine посредством взаимодействия через C++ интерфейс. Представлен proof-of-concept, направленный на предсказание урона в игровом процессе с использованием обучаемой модели.

**Ключевые слова:** Интеграция AI, нейросетевые модели, python, игровой процесс, Unreal Engine, предсказание урона.

В последнее десятилетие наблюдается активное развитие методов искусственного интеллекта (AI) и их применение в различных сферах, включая разработку игр. Однако большинство решений сосредоточено на использовании готовых библиотек, тогда как интеграция собственных моделей, разработанных на Python, в игровые движки остаётся актуальной задачей для исследования. Разработанная система состоит из двух основных компонентов:

- Нейросетевая модель на Python реализованная с использованием PyTorch, обученная на игровых данных для предсказания урона;
- Интеграционный мост на C++ – компонент Unreal Engine, позволяющий вызывать Python-скрипты с передачей входных параметров и последующим получением результата для использования в игровом процессе.

Схема интеграции Python и Unreal Engine представлена на рисунке 1.

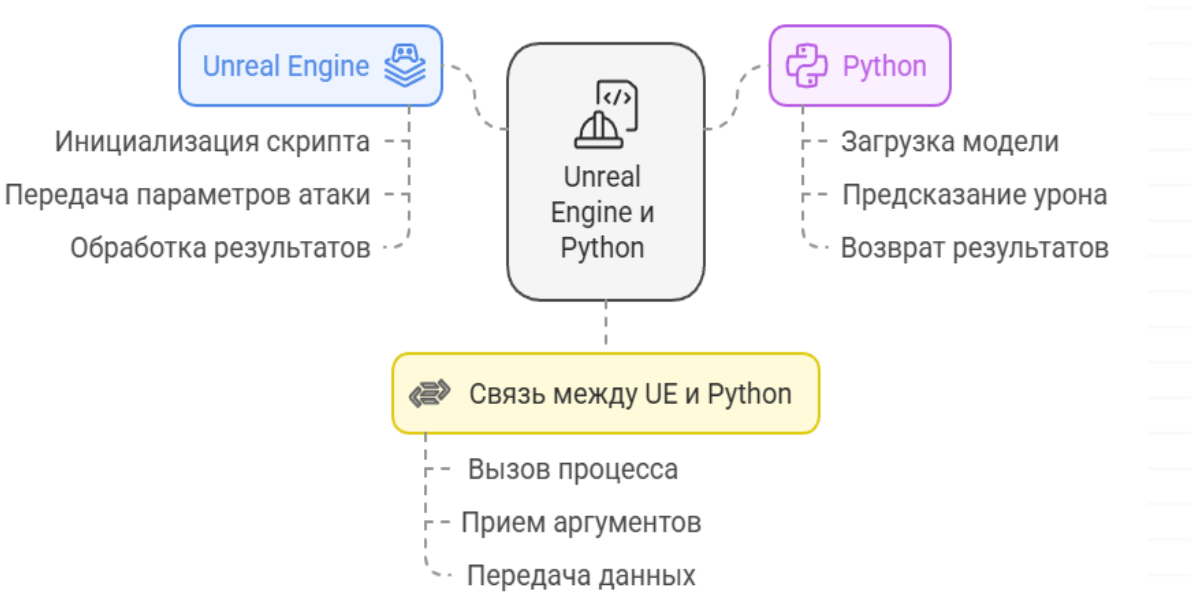


Рис. 1 Схема интеграции Python и Unreal Engine

Для обмена данными между Python и Unreal Engine используется механизм вызова внешнего процесса через C++ (FPlatformProcess::ExecProcess). При запуске скрипта с параметрами, Python-скрипт обрабатывает данные, возвращает предсказанный урон, который затем используется в логике игры.

Данные были получены из тестовой игры, где урон игрока рассчитывается по формулам учитывающим несколько параметров. В течение игровых сессий наша система логировала ключевые параметры боя: силу атаки, тип оружия, уровень игрока и фактический урон, полученный после столкновения. Эти данные автоматически сохранялись в базе данных, после чего производилась предварительная обработка (удаление выбросов, нормализация и агрегация) [1]. Нейросетевая модель была реализована с использованием библиотеки PyTorch. Архитектура сети включала входной слой (3 параметра), несколько скрытых слоёв с функцией активации ReLU и выходной слой, предсказывающий урон. Для оптимизации модели использовался оптимизатор Adam, а в качестве функции ошибки – среднеквадратичная ошибка (MSE). Обученная модель сохраняется в файл, который затем используется в интеграционном процессе. Обучение проводилось на GPU-устройстве, что позволило значительно сократить время процесса. Использование таких средств, как TensorBoard, обеспечило визуализацию динамики обучения и позволило оптимизировать параметры модели [2].

Мы создали специальный Actor в Unreal Engine, который в режиме игры получает параметры атаки (например, силу удара, тип оружия и уровень игрока). Эти данные передаются в виде аргументов при запуске Python-скрипта. Для корректного обмена данными важно, чтобы значения параметров были переданы с правильной локализацией (например, с точкой как десятичным разделителем). Мы использовали функции вроде FString::SanitizeFloat для преобразования чисел в строку, что позволило избежать ошибок при интерпретации данных на стороне Python. После выполнения Python-скрипта, его стандартный вывод считывается в Unreal Engine. Полученные данные анализируются, а затем интегрируются в игровую логику (например, для обновления анимаций, расчёта урона или изменения параметров взаимодействия с окружением). Дополнительное логирование помогает отслеживать корректность передачи и обработки данных (проверка кода возврата, вывод ошибок и т.д.).

Для оценки работоспособности интегрированной системы проводились серии экспериментов, направленных на проверку точности предсказания, скорости выполнения и устойчивости системы в условиях реального игрового процесса. Эксперименты проводились в тестовой сборке Unreal Engine, с активированным режимом детального логирования. Дополнительно использовались средства мониторинга производительности (Unreal Insights), что позволило оценить нагрузку на систему и выявить узкие места в интеграции. Результаты тестирования показали, что система способна выдавать предсказания в реальном времени, что подтверждает работоспособность предложенного подхода. Задержка между отправкой данных и получением предсказания составила менее 50 мс, что является приемлемым для игровых приложений. Средняя ошибка предсказания находилась в пределах допустимых значений, что позволило уверенно использовать модель для динамического изменения параметров боя. Предсказание урона в игре моделью показано на рисунке 2.

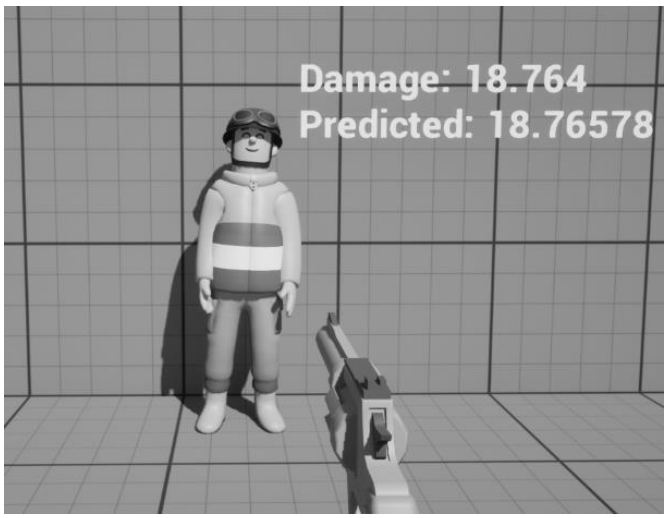

Рис. 2 Предсказание урона моделью

Интеграция нейросетевого модуля в Unreal Engine посредством Python предоставляет значительные преимущества в плане гибкости и быстроты разработки. Несмотря на proof-of-concept характер работы, результаты указывают на потенциал применения подобных методов для создания адаптивных игровых систем, таких как динамическое изменение сложности, адаптивное поведение NPC и генерация контента в реальном времени.

Представленный подход позволяет эффективно объединять возможности Python для машинного обучения и высокую производительность движка для реализации игровых механик. В дальнейшем рекомендуется проводить исследования по оптимизации обмена данными и адаптации модели на основе реальных игровых данных.

*Список литературы:*

1. Sapio F., Ratini R. Developing and testing a new reinforcement learning toolkit with unreal engine // International Conference on Human-Computer Interaction. – Cham: Springer International Publishing, 2022. – С. 317-334.

2. Вейдман С. Глубокое обучение: легкая разработка проектов на Python. – Санкт-Петербург: Питер, 2021. – 400 с