МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

ПРИМЕНЕНИЕ МЕТОДОВ РАСПОЗНАВАНИЯ РЕЧИ И ОБРАБОТКИ ИЗОБРАЖЕНИЙ В РАЗРАБОТКЕ ИГР

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Сорокина Леонида Сергеевича

Научный руководитель:
к.ф.-м.н., доцент _____ Огнева М. В.

подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ Огнева М. В.

подпись, дата

Саратов 2023

ВВЕДЕНИЕ

Актуальность темы. На успех игры влияют несколько факторов: использование новых механик или технологий, вариативность и иммерсивность (глубина погружения). Задача последнего пункта – создать иллюзию, что игровой мир мало отличается от реального. Улучшение иммерсивности является актуальной задачей на протяжении всего существования игровой индустрии. Каждая компания решает её индивидуальным способом: проработанными диалогами, современной графикой, реалистичным поведением персонажей. Благодаря развитию отрасли ИИ есть возможность улучшить поведение игровых персонажей, сделать их более «живыми». Именно поэтому внедрение методов машинного обучения в разработку игр может привести индустрию к новому этапу развития.

Цель бакалаврской работы – создать игрового компаньона, который будет обрабатывать голосовые команды игрока и использовать методы обработки изображений для проверки попадания противников под заданные условия.

Поставленная цель определила **следующие задачи**:

1. Сделать обзор аналогов
2. Сделать обзор методов для реализации ИИ в играх
3. Сделать обзор архитектуры игрового движка Unreal Engine 5
4. Сделать обзор библиотек
5. Реализовать механику стрельбы
6. Реализовать возможность распознавания речи с помощью библиотеки Vosk
7. Реализовать обработку изображений с помощью OpenCV
8. Объединить все наработки в единую систему

Теоретическая и/или практическая значимость бакалаврской работы. Благодаря внедрению методов машинного обучения в разработку игр поведение персонажей станет более похожим на человеческое, а

нахождение игрока в игровой среде – более естественным. Использование небольших моделей с высокой точностью сведут проблемы с производительностью к минимуму, улучшив при этом иммерсивность.

Структура и объём работы. Бакалаврская работа состоит из введения, 7 разделов и 5 подразделов, заключения, списка использованных источников и 24 приложений. Общий объем работы – 84 страниц, из них 36 страниц – основное содержание, включая 13 рисунков, цифровой носитель в качестве приложения, список использованных источников информации – 57 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор аналогов» посвящен рассмотрению игр, которые используют распознавание речи и обработку изображений в своём игровом процессе. К первым относятся «Binary Domain», «SoundSelf», «Radio General», «Starship Commander: Arcade», «The Broken Seal: Arena», «Tower!3D Pro», «Lovercraft World». Ко вторым – «Pacman», «Starcraft», «Chess».

Второй раздел «Обзор методов для реализации ИИ в играх» посвящен рассмотрению методов реализации ИИ в играх «Black & White» и «Creatures». В игре «Black & White» разработчики использовали несколько подходов для создания ИИ. Во-первых, символические пары «атрибут-значение» для представления убеждения существа о каком-либо отдельном объекте. Во-вторых, ситуационное исчисление (ИИ на основе правил), чтобы дать существам их базовый интеллект в отношении объектов. В-третьих, деревья решений, которые представляют убеждения агентов об общих типах объектов. Алгоритм, используемый для построения деревьев и минимизации энтропии, основан на ID3. Для создания ИИ в «Creatures» разработчики использовали четырёхслойную нейронную сеть, которая состояла из 952 нейронов и 5000 связей между ними. Эти нейроны организованы в 9 функционально различных групп, которые называются «долями». Поведение каждого существа зависит от 2 критериев: какое действие нужно выполнить и на какой объект применить действие. За это отвечают доли «Decision» и «Attention». Далее представлены описания оставшихся долей. «Stimulus Source» (активируется в зависимости от объектов, которые видит существо), «Noun» (активируется, когда игрок набирает название объекта), «Concept» (возможные ситуации, в которые может попасть существо), «Drive» (обозначения физического и эмоционального состояния существа), «General Sense» (используется для конкретных событий, таких как похлопывание, пощечина и столкновение со стеной), «Verb» (активируется так же, как «Noun», используется для влияния игрока на принятие решения существом).

Третий раздел «Обзор библиотек» посвящен обзору библиотек для распознавания речи и обработки изображений. К первым относятся «Vosk» (предназначена для потоковой обработки аудио, работает без подключения к серверу, содержит небольшие модели для 20 языков), «CMU Sphinx» (библиотека для распознавания и набор для обучения моделей, работает без подключения к серверу, имеет гибкую архитектуру), «Google Cloud Speech API» (предоставляет подсказки для распознавания речи, может работать как с подключением к серверу, так и без него), «Microsoft Speech to Text API» (поддержка около 100 языков, гибкое развёртывание), «Houndify API» (одноэтапная обработка речи с уменьшенным временем выполнения, поддержка около 25 языков). Ко вторым – «OpenCV» (подходит для устройств с небольшим объёмом оперативной памяти, является кроссплатформенной, имеет оптимизации для процессоров Intel), «VIGRA» (сделан упор на настраиваемые алгоритмы и структуры данных, подходит для многомерных изображений), «VXL» (представлена в виде набора отдельных библиотек, работающих как единая система), «Magick++» (поддерживает неявный подсчёт ссылок и семантику значений, интегрированы STL-контейнеры), «Boost GIL» (преставление изображения абстрагировано от алгоритмов, высокая производительность, гибкость и расширяемость алгоритмов, совместимость с STL), «SIMD» (особенность библиотеки заключается в оптимизации алгоритмов под следующие расширения процессора: SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, AVX2 и AVX-512 для x86/x64, VMX(Altivec) и VSX(Power7) для PowerPC, NEON для ARM), «Libvips» (небольшое потребление памяти за счёт загрузки изображения частями, горизонтальный стиль многопоточности), «ONNX Runtime» (позволяет ускорить вывод результата машинного обучения на всех объектах развертывания с помощью единого набора API).

Четвёртый раздел «Обзор подходов к распознаванию речи» посвящен рассмотрению методов для распознавания речи: скрытые Марковские модели (СММ) и нейронные сети. Были описаны различные

типы нейронных сетей, используемых для решения данной задачи: RNN, RNN-Transducer, CNN, FCN, Speech-Transformer, Transformer-Transducer, Conformer.

Пятый раздел «Обзор нейросетей для обработки изображений» посвящен описанию YOLOv8 и MobileNetSSDv2. Для первой была рассмотрена архитектура, даны пояснения к компонентам, приведено сравнение с предыдущими версиями YOLO. Для второй также была рассмотрена архитектура, названы количество слоёв с параметрами и примерный размер итоговой модели после обучения.

Шестой раздел «Обзор архитектуры движка Unreal Engine 5» посвящен описанию иерархии игровых объектов и процессу их инициализации. Также было дано определение игрового движка и приведено сравнение Unreal Engine 5 с другими игровыми движками вместе с пояснением выбора первого для создания проекта.

Седьмой раздел «Практическая часть» посвящен описанию реализации умного компаньона. В первом подразделе подробно рассмотрена архитектура созданного приложения. Она состоит из игровой части, которая содержит в себе классы для работы с персонажами, передвижением, стрельбой и генерацией снарядов, модулей распознавания речи и обработки изображений, обработчика команд. Каждый из модулей после выполнения своей работы возвращает определённые данные обработчику команд – распознанный текст и угол поворота. Обработчик команд хранится в виде умного указателя внутри игровой части и запускается в отдельном потоке. Во втором подразделе рассмотрена игровая часть, т.е. были описаны классы, отвечающие за реализацию крепления оружия (TP_PickUpComponent, TP_WeaponComponent), генерации снарядов (TP_Projectile), передвижения персонажа (SmartCompanionCharacter), и использованные Blueprints – AnimBP, который ответственен за настройку переходов между анимациями. Отдельное внимание было уделено структуре AnimBP: AnimGraph с AnimStateMachine и EventGraph. Первый хранит в себе возможные анимации

персонажа и конечный автомат для переключения между ними. Автомат работает с 6 состояниями: Idle (покой – стандартная поза), StealthIdle (покой в скрытом режиме), WeaponIdle (покой с оружием), RunWithoutWeapon (движение без оружия), StealthRun (движение в скрытом режиме), WeaponRun (движение с оружием). Второй предназначен для настройки условий перехода между анимациями. Также были изложены процессы подключения сторонних библиотек к проекту и настройки системы сборки. Также была описана разница между модулями с точки зрения системы сборки и архитектуры проекта. Третий подраздел посвящен реализации модуля распознавания речи. Для работы данного модуля использовались следующие библиотеки: Vosk – распознавание речи, PortAudio – считывание аудиоданных с микрофона, Simpleson – обработка json. Модель, использованная в приложении, называется «vosk-model-small-en-us-0.15». Она была выбрана за счёт небольшого размера, высокой точности и возможности динамически настроить словарь. Частота ошибок (Word error rate) данной модели составляет 9.85. Также в данном подразделе описана реализация функций модуля – инициализация необходимых классов, получение данных и их распознавание, освобождение данных модуля. Четвёртый подраздел посвящен реализации модуля обработки изображений. Для работы данного модуля использовалась библиотека OpenCV и модель YOLOv8s. Для обучения модели на каждый цвет одежды был создан отдельный датасет из 300 изображений, которые представлены в виде скриншотов запущенного приложения. Данные были разбиты на тренировочные, валидационные и тестовые в соотношении «70/20/10». Обучение длилось 7 эпох. По результатам матрицы путаниц, представленной в работе, на 90% изображений был верно определён противник. Кроме основных методов интерфейса модуля (Initialize, Run и Shutdown) были добавлены дополнительные – preProcess (предобработка), postProcess (постобработка), getRotateAngle (вычисление угла поворота), setPrimaryModel (установка главной модели). Предобработка состоит из преобразования

изображения в blob, далее blob передаётся нейросети, а в результате работы модели возвращается матрица, содержащая координаты центра рамки с обнаруженным объектом, ширину и высоту рамки, вероятность верного обнаружения. Постобработка выполняет обход результатов и применяет немаксимальное подавление с целью удаления перекрывающихся рамок. Угол поворота вычисляется с помощью формулы арктангенса. Сначала происходит переход в режим от 1 лица, далее с учётом координат центра рамки и точки с координатами «320, 0», лежащей на середине изображения, вычисляются длины сторон прямоугольного треугольника, затем они используются для нахождения угла по формуле арктангенса. Настройка главной модели происходит с помощью получения от команд KillRed и KillBlue строки с названием модели – «red» или «blue». Затем установленная модель используется в предобработке. Пятый подраздел посвящен реализации обработки команд. Основным классом является CommandHandler, который наследован от интерфейса потока внутри Unreal Engine 5. Данный класс содержит хранилище с командами. При удерживании кнопки «G» запускается работа модуля распознавания речи. Обработанный текст сравнивается с ключами из хранилища. В случае совпадения выполняется связанная с ключом команда. Все команды в проекте наследуются от единого интерфейса.

ЗАКЛЮЧЕНИЕ

В данной работе были сделаны обзоры игр, в которых применяются методы обработки изображений и голоса, и библиотек с моделями для компьютерного зрения и распознавания речи. Также была рассмотрена архитектура игрового движка Unreal Engine 5. Помимо этого, была подобрана модель для распознавания речи, которая работает совместно с библиотеками Vosk и PortAudio. Также были созданы датасеты и обучены модели yolov8 для обработки изображений. Данные модели затем были настроены для работы с библиотекой OpenCV. Ещё было настроено внедрение библиотек в проект с точки зрения системы сборки игрового движка. В итоге был создан игровой персонаж, который распознаёт речь игрока и выполняет команды по обнаружению противников в красной и синей одеждах в зависимости от результата распознавания. Обработка команд была вынесена в отдельный поток, что ускорило работу.

Основные источники информации:

1. Zhang, J. et al. Research on the Application of Artificial Intelligence in Games // 2022 9th International Conference on Digital Home (ICDH). – IEEE, 2022. – P. 207-212.

URL: https://ieeexplore.ieee.org/document/9978360 (дата обращения: 30.05.2023).

2. Mi, Q. Gao, T. Improved Belgian AI Algorithm for Dynamic Management in Action Role-Playing Games // Applied Sciences. – 2022. – Vol.12, No.22. – P. 11860

URL: https://www.researchgate.net/publication/365637807_Improved_Belgian_AI_Algorithm_for_Dynamic_Management_in_Action_Role-Playing_Games (дата обращения: 30.05.2023).

3. Gupta, A. Sirpal, S. AI in Gaming and Entertainment: Applying Artificial Intelligence Algorithms in a Game // Applying AI-Based IoT Systems to Simulation-Based Information Retrieval. – IGI Global, 2023. – P. 63-76.

URL:

https://www.researchgate.net/publication/370454524_AI_in_Gaming_and_Entertainment_Applying_Artificial_Intelligence_Algorithms_in_a_Game (дата обращения: 30.05.2023).

4. Федосин С.А., Еремин А. Ю. Классификация систем распознавания речи. // Саранск. : МГУ им. Н.П. Огарева - 2009. - С. 3.

5. Тампель И.Б, Карпов А.А. Автоматическое распознавание речи. // СПб.: Университет ИТМО - 2016. - С. 113.

6. Побегайло, А. П. Системное программирование в Windows. // СПб.: БХВ-Петербург. - 2006. - С. 578-592

7. Nystrom, R. Game programming patterns. // Genever Benning, 2014.