Труды МАИ. 2026. № 147.
Trudy MAI. 2026. No. 147. (In Russ.)

Научная статья / Original Article
УДК 004.4
URL: https://trudymai.ru/published.php?ID=188118
EDN: https://www.elibrary.ru/VPUVDS

РАЗРАБОТКА ИНСТРУМЕНТАРИЯ ДЛЯ СОЗДАНИЯ ГОЛОСОВЫХ ИНТЕРФЕЙСОВ С ЦЕЛЬЮ ИХ ПРИМЕНЕНИЯ В АЭРОКОСМИЧЕСКИХ VR- И AR-ТРЕНАЖЕРАХ

К.С. Васильева✉

Федеральное государственное автономное образовательное учреждение высшего образования "Казанский (Приволжский) федеральный университет"

Казань, Россия

✉ksenya_vasilieva@mail.ru

Цитирование: Васильева К.С. Разработка инструментария для создания голосовых интерфейсов с целью их применения в аэрокосмических VR- и AR-тренажерах // Труды МАИ: электрон. журнал. № 147. URL: https://trudymai.ru/published.php?ID=188118

Аннотация. В работе представлена архитектура и реализация программного инструментария для среды Unity, предназначенного для создания голосовых интерфейсов и интеграции их в приложения виртуальной реальности (Virtual Reality, VR) и дополненной реальности (Augmented Reality, AR), ориентированные на использование в аэрокосмической отрасли. Показано, что существующие модули голосового взаимодействия, применяемые в VR- и AR-приложениях, не универсальны, жестко привязаны к конкретным сценариям использования и не обеспечивают достаточной гибкости настройки. Сравнительный анализ существующих решений показал, что ключевыми недостатками являются такие проблемы как отсутствие пользовательских интерфейсов для связывания голосовых команд с логикой приложения, невозможность выбора между локальными и облачными сервисами автоматического распознавания речи (Automatic Speech Recognition, ASR), зависимость от сетевого подключения и

отсутствие устойчивости к вариативности формулировок команд. В рамках работы проведено экспериментальное измерение временной задержки между началом произнесения фразы и получением текстового результата для сервисов ASR iFlyTek, Vosk и Whisper на наборе фраз трех типов сложности. Установлено, что минимальную задержку в распознавании речи среди облачных решений имеет сервис iFlyTek (2934 мс), тогда как среди локальных реализаций наилучший результат показала библиотека Vosk (3589 мс). Whisper показал более высокие значения: 4732 мс для модели tiny, 6314 мс для модели base и 4913 мс для модели small. Эксперимент проводился на автономной VR-гарнитуре Pico 4 с 8 ГБ оперативной памяти под управлением операционной системы Android 10. В работе описаны архитектура и схема функционирования разработанного инструментария, поддерживающего локальное и облачное распознавание речи, предоставляющего средства конфигурации голосовых команд и параметров ASR, а также механизм автоматической генерации различных формулировок команд на основе модели GLM-4.7-Flash. Предложенное решение снижает трудозатраты при разработке голосовых интерфейсов и повышает устойчивость системы к вариативности формулировок голосовых команд пользователя за счет автоматической генерации семантически близких фраз. Был реализован прототип голосового интерфейса, предназначенного для взаимодействия с виртуальной приборной панелью на сцене Unity. Его работоспособность подтверждает применимость разработанного инструментария при создании VR- и AR-систем аэрокосмического назначения.

Ключевые слова: голосовой интерфейс; автоматическое распознавание речи; Unity; VR; AR; Vosk; iFlyTek; GLM-4.7-Flash.

DEVELOPMENT OF A TOOLKIT FOR VOICE INTERFACE CREATION IN AEROSPACE VR AND AR SIMULATORS

K.S. Vasilieva✉

Federal State Autonomous Educational Institution of Higher Education «Kazan (Volga Region) Federal University», Kazan, Russia

✉ksenya_vasilieva@mail.ru

Citation: Vasilieva K.S. Development of a Toolkit for Voice Interface Creation in Aerospace VR and AR Simulators // Trudy MAI. 2026. No. 147. (In Russ.).
URL: https://trudymai.ru/published.php?ID=188118

Abstract. This paper presents the architecture and implementation of a software toolkit for the Unity environment, designed for creating voice interfaces and integrating them into virtual reality (VR) and augmented reality (AR) applications targeted at the aerospace industry. It is shown that existing voice interaction modules used in VR and AR applications are not universal, are strictly tied to specific use cases, and do not provide sufficient customization flexibility. A comparative analysis of existing solutions revealed that key shortcomings include the lack of user interfaces for linking voice commands to application logic, the inability to choose between local and cloud-based Automatic Speech Recognition (ASR) services, network connectivity dependence, and a lack of robustness to variability in command wording. This paper also experimentally measured the time delay between the onset of a phrase and the receipt of the text result for the iFlyTek, Vosk, and Whisper ASR services using a set of phrases of three complexity levels. It was found that the iFlyTek service has the lowest speech recognition latency among cloud-based solutions (2934 ms), while the Vosk library demonstrated the best result among local implementations (3589 ms). Whisper demonstrated higher values: 4732 ms for the tiny model, 6314 ms for the base model, and 4913 ms for the small model. The experiment was conducted on a standalone Pico 4 VR headset with 8 GB of RAM running the Android 10 operating system. This paper describes the architecture and functional diagram of the developed toolkit, which supports local and cloud-based speech recognition, provides tools for configuring voice commands and ASR parameters, and a mechanism for automatically generating various command formulations based on the GLM-4.7-Flash model. The proposed solution reduces the labor costs of developing voice interfaces and improves the system's resilience to variability in the formulation of user voice commands by automatically generating semantically similar phrases. A prototype voice interface was implemented for interacting with a virtual instrument panel in a Unity scene. Its functionality confirms the applicability of the developed toolset for creating VR and AR systems for aerospace applications.

Keywords: voice interface; automatic speech recognition; Unity; VR; AR; Vosk; iFlyTek; GLM-4.7-Flash.

Введение

В настоящее время технологии виртуальной реальности (VR) и дополненной реальности (AR) находят широкое применение в аэрокосмической отрасли на различных этапах жизненного цикла аппаратов и оборудования – от проектирования до эксплуатации. В частности, исследование [1] показывает, что использование VR для процедурного обучения персонала обслуживанию современных воздушных судов демонстрирует сопоставимую или улучшенную эффективность по сравнению с традиционными тренажерами, что подтверждается опытным путем.

Наибольший эффект данные технологии демонстрируют при решении задач моделирования операций по созданию, установке и поддержания работоспособности масштабных систем, предназначенных для применения в условиях космического пространства [2].

Отдельное направление исследований посвящено использованию VR для проведения макетно-конструкторских и экспериментальных испытаний, включая отработку сценариев взаимодействия оператора с перенесенными в виртуальную среду макетами космических аппаратов [3]. Такой подход позволяет анализировать конструктивные и эксплуатационные особенности изделий без необходимости создания их физических прототипов.

Исследование [3] показывает, в рамках развития решений проблемы использования VR в аэрокосмической отрасли необходимо совершенствование средств человеко-машинного взаимодействия, обеспечивающих более естественное и интуитивно понятное управление в условиях высокой информационной нагрузки. Одним из перспективных направлений развития является использование голосового управления в качестве человеко-машинного интерфейса. Данный подход позволит снизить когнитивную нагрузку на оператора и обеспечит более высокий уровень иммерсивности, имитируя, например, взаимодействие с бортовыми системами или радиообмен. Внедрение

голосового взаимодействия с виртуальной средой в VR- и AR-приложениях приобретает особую актуальность в сценариях, где использование традиционных средств управления, таких как контроллеры, становятся недоступными, например, в условиях занятости рук или ограниченности движений пользователя при выполнении сложных операций. В подобных ситуациях управление с помощью голоса становится универсальным инструментом для решения подобных задач.

Автоматическое распознавание речи (ASR) активно используются в VR- и AR-приложениях, которые решают задачи в разных областях, включая аэрокосмическую отрасль. Например, в работе [4] представлена метавселенная, созданная с целью обучения техническому обслуживанию самолетов Boeing-737. Главным инструментом управления и навигации в данной среде является голос: модуль понимания речи Neuro-Symbolic Speech Executor (NSSE) включает в себя искусственный интеллект для обработки преобразованной в текст речи и вызова последовательности команд, а также формирование ответа пользователю на основе контекста и информации о конкретном самолете. В другом исследовании [5] голосовые команды позволяют взаимодействовать с автомобилем в AR и выполнять такие действия, как изменение цвета автомобиля и открытие дверей.

Стоит отметить, что подобные проекты нацелены на решение конкретных прикладных задач и строят архитектуру взаимодействия с ASR, которая не является универсальной и обобщенной. Это затрудняет адаптацию данных модулей для других сценариев различных предметных областей.

Цель данной работы – разработка инструментария в среде Unity [6] для создания и интеграции голосовых интерфейсов в аэрокосмических VR и AR приложениях. В рамках работы проведен анализ существующих инструментов для реализации голосового управления в VR- и AR-приложениях, разработанных в среде Unity, а также выполнен сравнительный анализ локальных и облачных сервисов ASR на основе экспериментального измерения временных характеристик. Спроектирована расширяемая архитектура инструментария. Разработан визуальный редактор голосовых команд и реализован механизм

сопоставления результатов сервиса ASR с методами приложения, основанный на алгоритме нечеткого поиска. Разработан модуль генерации синонимичных фраз.

Анализ существующих решений

На данный момент для разработки голосовых интерфейсов существует ряд плагинов, ориентированных на создание VR и AR приложений в среде разработки Unity. Каждый из них позиционирует себя прежде всего как инструмент быстрого внедрения систем ASR в проект, однако уровень их универсальности и расширяемости различен. В данном разделе приведен анализ данных решений с точки зрения их функциональности и ограничений в использовании.

Game Voice Control Plugin [7], разработанный Stendhal Syndrome Studio, представляет собой плагин для создания голосовых интерфейсов в редакторе Unity, реализующий распознавание голоса на основе локальной библиотеки PocketSphinx [8]. Поддерживается распознавание речи на 9 языках, включая русский язык. Конфигурация модуля распознавания речи представлена заданием словаря распознаваемых лексем, что позволяет использовать в качестве голосовых команд нестандартную лексику и слова, не существующие в выбранном языке. Однако данное решение не обладает семантической гибкостью: система распознавания речи ориентирована преимущественно на точное соответствие заданным фразам. Также функциональные возможности плагина включают активацию системы ASR по заданному пользователем слову и настройку порога чувствительности к обнаружению голоса микрофоном.

Meta Voice SDK [9] – инструментарий для интеграции голосового управления в VR-приложения, созданные в среде Unity. Он разработан компанией Meta Platforms (организация, деятельность которой признана экстремистской и запрещена на территории РФ) и интегрирован и доступен на ее платформе Wit.ai. У разработчика есть возможность конфигурировать пользовательские голосовые команды, обучая модель на множестве примеров похожих фраз, что позволяет системе корректно интерпретировать семантически похожие фразы за счет применения методов обработки естественного языка (NLP). Однако процессы создания команд и обработка аудио происходит в облачном сервисе Wit.ai, что

накладывает ограничения в виде доступности сервисов Meta в регионе использования и доступа к сети Интернет.

В данном разделе стоит отметить решения, являющиеся программными оболочками для Unity существующих систем ASR, но не позволяющие задавать пользовательские команды и создавать, и интегрировать полноценные модули голосового управления, а также перекладывающие задачу обработки полученного результата на разработчика. Например, плагин **Unity Speech to Text Plugin for Android & iOS** [10] делегирует процесс преобразование речи в текст фреймворкам конкретной операционной системы (ОС) – Speech для iOS (версия 10 и выше) и Speech Recognition & Synthesis, разработанный компанией Google, для Android (версия 23 и выше). Вследствие этого тестирование работоспособности модуля, разработанного с использованием данного плагина, возможно только на мобильных устройствах, что может затруднять процесс разработки.

Еще один плагин, **Speech Control Plugin for VR** [11] от разработчиков Weelco Inc., использует облачный сервис Google Cloud Speech-to-Text [12], что вынуждает будущее приложение быть зависимым от наличия стабильного подключения к сети. Данное решение использует клиент-серверную архитектуру, где клиент, реализованный в Unity, отвечает за захват и передачу на сервер аудиоданных, делегируя преобразование аудио в текст серверу, представленному внешним облачным сервисом. Также для использования плагина пользователю необходимо сгенерировать API-ключ для доступа к сервису ASR, который затем явно указывается в окне Inspector редактора Unity. При декомпиляции сборки приложения данный ключ может быть легко извлечен, что создает угрозу безопасности сервиса и преждевременного достижения лимита запросов к нему.

Сравнительный анализ, результаты которого представлены в таблице 1, существующих решений для интеграции голосового управления в VR- и AR-приложения позволил выделить ряд недостатков и функциональных и системных ограничений:

1. Отсутствие пользовательских интерфейсов и механизмов декларативного сопоставления голосовых команд и логики приложения;

2. Ограниченная возможность отладки и настройки голосовых команд в редакторе Unity, что приводит к увеличению времени разработки и необходимости проведения тестирования только на целевых платформах;

3. Отсутствие гибкости при формировании требований к доступности сети;

4. Невозможность обработки синонимичных и семантически схожих фраз;

5. Недоступность на территории России.

Таблица 1

Результаты сравнительного анализа существующих решений для интеграции голосового управления в VR- и AR-приложения

Критерий	Game Voice Control Plugin	Meta Voice SDK	Unity Speech to Text Plugin for Android & iOS	Speech Control Plugin for VR
Возможность конфигурации голосовых команд	Ограниченная	Да	Нет	Нет
Проведение цикла разработки и тестирования в редакторе Unity	Да	Нет	Нет	Да
Устойчивость к вариативности формулировок голосовых команд	Нет	Да	Нет	Нет
Зависимость от подключения к сети	Нет	Да	Нет	Да
Доступность на территории Российской Федерации	Да	Нет	Нет	Нет

Постановка задачи

Наличие недостатков существующих решений обосновывает потребность в разработке универсального инструментария для редактора Unity, позволяющего создавать голосовые интерфейсы и интегрировать его в VR и AR среды, направленных на решение различных практических задач. На основе анализа, приведенного в предыдущем разделе, были сформулированы следующие функциональные и нефункциональные требования:

1. Доступность конфигурации ASR и голосового интерфейса в редакторе Unity. Инструментарий должен предоставлять средства для создания и редактирования голосовых команд внутри Unity Editor;
2. Возможность переключения между облачным и локальным режимами распознавания;
3. Наличие интерфейса и функционала для генерации семантически близких формулировок голосовых команд;
4. Поддержка русского языка и доступность на территории Российской Федерации;
5. Легкая интеграция в модули разрабатываемых с помощью данного инструментария приложений;
6. Наличие логирования таких параметров, как время, точность распознавания и т.д.

Особое внимание следует уделить минимизации времени задержки распознавания, определяющейся как временной интервал между моментом произнесения голосовой команды и фиксацией текстового результата распознавания, поскольку большая задержка влияет на отзывчивость системы в динамичных сценариях VR- и AR-приложений.

Выбор инструментальных средств

В качестве целевой платформы для разработки и использования плагина был выбран Unity. Выбор обоснован следующей совокупностью свойств данной среды разработки:

1. Кроссплатформенность и поддержка VR/AR устройств;
2. Возможность создания окон и элементов графического пользовательского интерфейса (GUI) в редакторе;
3. Встроенные библиотеки и готовые компоненты для управления контроллерами и взаимодействия с виртуальной и дополненной реальностью, такие как OpenXR, AR Foundation и др.;
4. Высокая производительность на мобильных устройствах, включая VR/AR устройства, а также наличие инструментов для оптимизации [13];

5. Поддержка инструментов отладки и тестирования функций и работоспособности VR/AR приложений в редакторе Unity.

Плагин реализован на языке программирования C#.

Для генерации синонимичных голосовых команд выбрана модель GLM-4.7-Flash [14], разработанная Z.ai. Данная модель показывает лучшие результаты на тестах эталонных показателей, представленные в таблице 2.

Таблица 2

Результаты проведения тестов эталонных показателей моделей

Эталонный показатель	GLM-4.7-Flash	Qwen3-30B-A3B-Thinking-2507	GPT-OSS-20B
GPQA	75.2%	73.4%	71.5%
τ^2-Bench	79.5%	49.0%	47.7%
BrowseComp	42.8%	2.29%	28.3%

Обращение к данной модели осуществляется посредством Hugging Face [15] – платформы с открытым исходным кодом, на которой размещены обученные модели для генерации текста, изображений, аудио, видео, а также инструменты для обработки естественного языка и компьютерного зрения. Hugging Face позволяет пользователям использовать готовые модели через API.

Тестирование сервисов ASR и апробация прототипа приложения с голосовым управлением, реализованным с использованием разработанного инструментария, проводились на автономной VR-гарнитуре Pico 4 с ОС Android 10, 8 ГБ оперативной памяти и накопителем объемом 128 ГБ. Для разработки тестовой виртуальной среды использовались компоненты XR Interaction Toolkit [16] и PICO Unity Integration SDK [17].

Сравнительный анализ сервисов ASR

В рамках данного эмпирического исследования основными критериями выбора сервисов ASR являлись поддержка русского языка и доступность для использования на территории Российской Федерации. Также предпочтение было отдано бесплатным сервисам или сервисам с достаточно большим количеством бесплатного времени аудиофрагментов для преобразования в текст. В результате

для экспериментального измерения времени задержки были выбраны следующие сервисы ASR: **iFLyTek Real-time ASR** [18], **Whisper** [19] и **Vosk** [20].

Whisper и Vosk предлагают для использования несколько моделей с разным количеством параметров и, соответственно, с разным объемом занимаемой памяти. Учитывая, что целевые устройства для запуска VR- и AR-приложений – это устройства на ОС Android и iOS, были выбраны модели, занимающие не более 1 ГБ: для Whisper – tiny, base и small, для Vosk – vosk-model-small-ru.

Для фиксации времени начала произнесения речи и времени получения результата распознавания был реализован вспомогательный модуль TimeMeasurement. Момент начала детекции голоса системой определяется как превышение суммарной энергией аудиосигнала порогового значения.

Для определения времени задержки распознавания был реализован модуль IFlyTekSpeechRealTimeRecognizerService, который отвечает за установление WebSocket соединения между клиентом Uniy и сервером iFlyTek, формирование пакетов данных и отправкой их на сервер. Схема работы данного модуля и его взаимодействие с другими вспомогательными модулями представлена на рисунках 1 и 2.

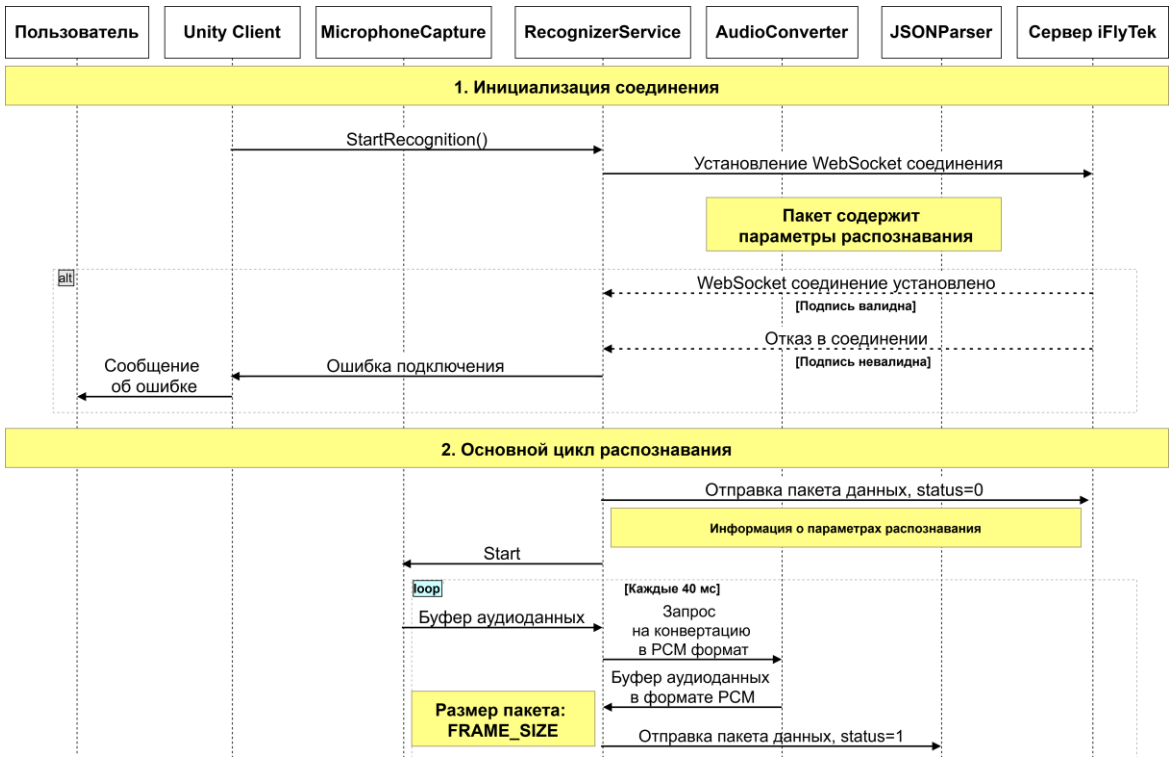

Рисунок 1 – Диаграмма последовательности первых двух этапов распознавания речи с помощью облачного сервиса iFlyTek; MicrophoneCapture – модуль для захвата аудиопотока с микрофона устройства, AuidoConverter – модуль для кодирования аудиофрагментов методом PCM, JSONParser – модуль для извлечения текстового результата распознавания из ответа сервера в формате JSON

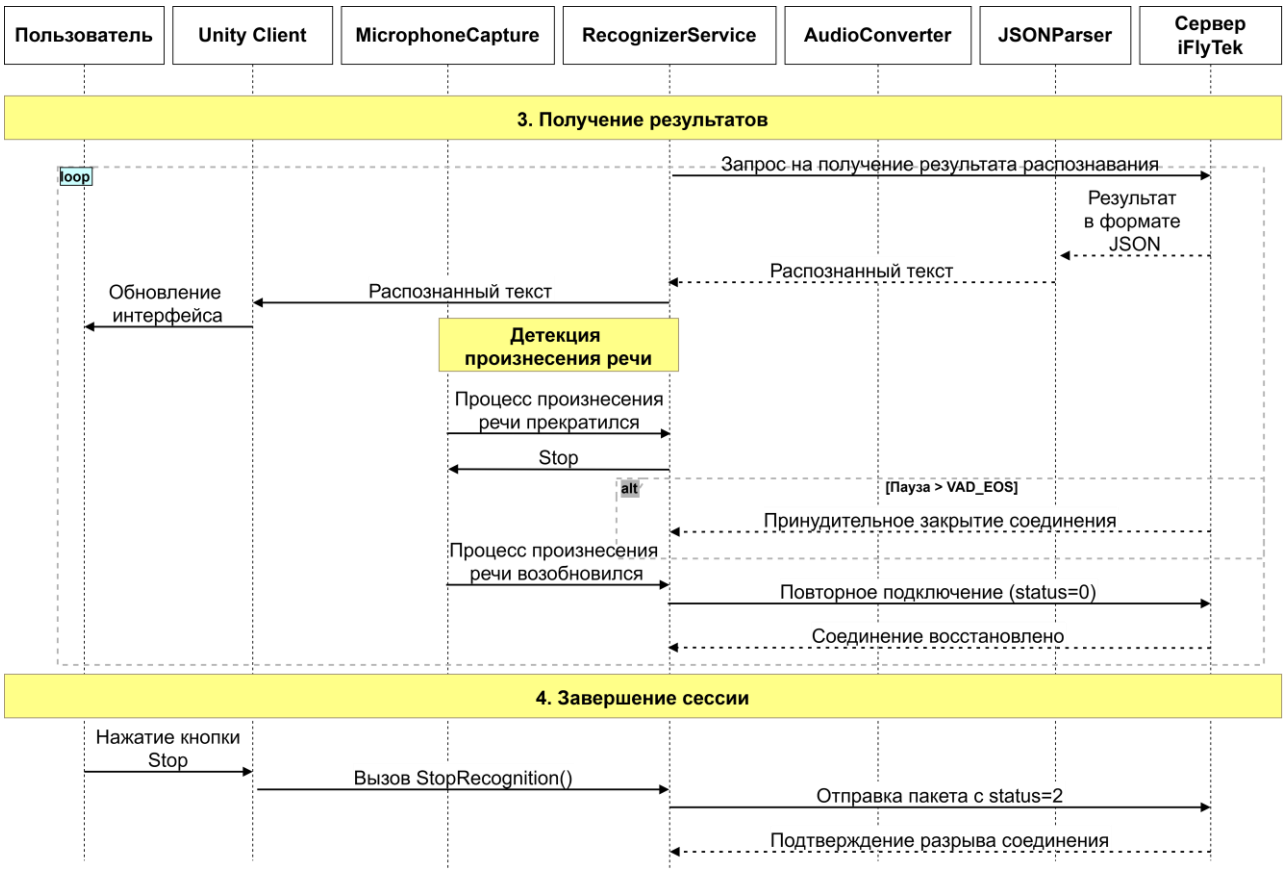

Рисунок 2 – Диаграмма последовательности первых двух этапов распознавания речи с помощью облачного сервиса iFlyTek

Для тестирования технологий распознавания речи Whisper и Vosk были использованы уже реализованные решения для Unity, работающие локально – Whisper.unity [19] от разработчика Macoron и Vosk-unity-asr [20] от компании Alpha Cephei.

Тестирование проводилось на трех группах фраз. Для первой группы были выбраны односложные фразы с целью проверки работоспособности системы. Во второй группе использовались фразы, включающие в себя звуковые повторы, с целью проверки точности распознавания. Третья группа состояла из длинных (более 5 слов) фраз, содержащих числительные и имена собственные.

В таблице 3 представлены экспериментально полученные значения метрик, рассчитанные как среднее арифметическое измерений по множеству повторений в рамках определенной группы фраз. Также было вычислено стандартное отклонение для определения подозрительных значений.

Таблица 3

Результаты экспериментального измерения времени задержки распознавания сервисами ASR

ASR	Модель	Время задержки (мс) в группе №1 Пример – «Старт»	Время задержки (мс) в группе №2 Пример – «Надеть одежду, одеть надежду»	Время задержки (мс) в группе №3 Пример – «Поверни на девяносто градусов»	Среднее время задержки (мс)
iFlyTek Real-time ASR	–	2000	3429	3373	2934
Whisper	tiny	3521	5964	4711	4732
	base	5287	7628	6027	6314
	small	3800	5890	5049	4913
Vosk	vosk-model-small-ru	2500	3490	4777	3589

Облачный сервис iFlyTek продемонстрировал наименьшую среднюю задержку. Среди локальных решений Vosk показывает наилучший результат.

Создание редактора голосовых команд

Для возможности гибкого управления голосовыми командами и соответствующими им методами было создано пользовательское окно Voice Command Editor в редакторе Unity (рисунок 3).

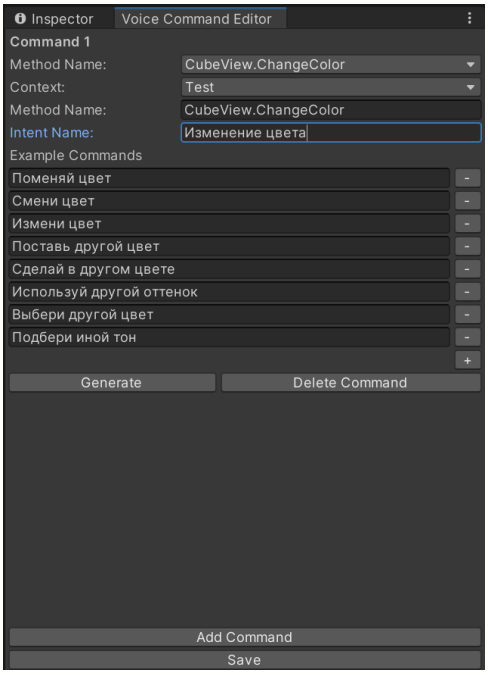

Рисунок 3 – Окно Voice Command Editor для редактирования команд в Unity

В данном окне разработчику доступны следующие функции:

1. Add Command – создание новой команды;
2. Delete Command – удаление команды;
3. Generate Command – генерация новых голосовых команд;
4. Save – сохранение изменений и перезапись файла с командами.

Голосовая команда в контексте инструментария – структура данных, включающая ее название, список ее вариаций и метод, который будет вызван, если система распознавания речи зафиксирует голосовую команду, похожую на одну из представленных. Список голосовых команд хранится в виде JSON-файла в StreamingAssets и создается автоматически при первом сохранении. Разработчик имеет возможность заменять файл с голосовыми командами на другой в конфигураци инструментария.

Для того чтобы разработчик мог безошибочно конфигурировать методы, которые будут вызваны после распознавания голосовой команды, был создан атрибут VoiceCommandAttribute. Таким образом, разработчик может отметить каждый метод, который он хочет использовать в своем голосовом интерфейсе данным атрибутом.

Преимущество разрабатываемого инструментария также заключается в том, что в компонентах, которые будут управляться голосом, не нужно прописывать зависимости от дополнительных классов голосового управления – достаточно только отметить нужные методы атрибутами и указать для них голосовые команды в пользовательском окне.

Чтобы повысить устойчивость системы к вариативности пользовательских формулировок голосовой команды, реализован сервис PhraseGenerationService, отвечающий за автоматическое расширение набора фраз. Сервис формирует запрос к модели GLM-4.7-Flash, включающий текущие варианты фраз и заранее заданную инструкцию, и асинхронно отправляет его, используя вспомогательный микросервис с функцией инкапсуляции конфиденциальных API-ключей и защиты от их хранения в коде клиента, на сервер Hugging Face. После получения ответа осуществляется парсинг фраз и их валидация (удаление дубликатов и лишних символов).

Архитектура и ключевые компоненты инструментария

Архитектура разработанного инструментария представлена на рисунке 4.

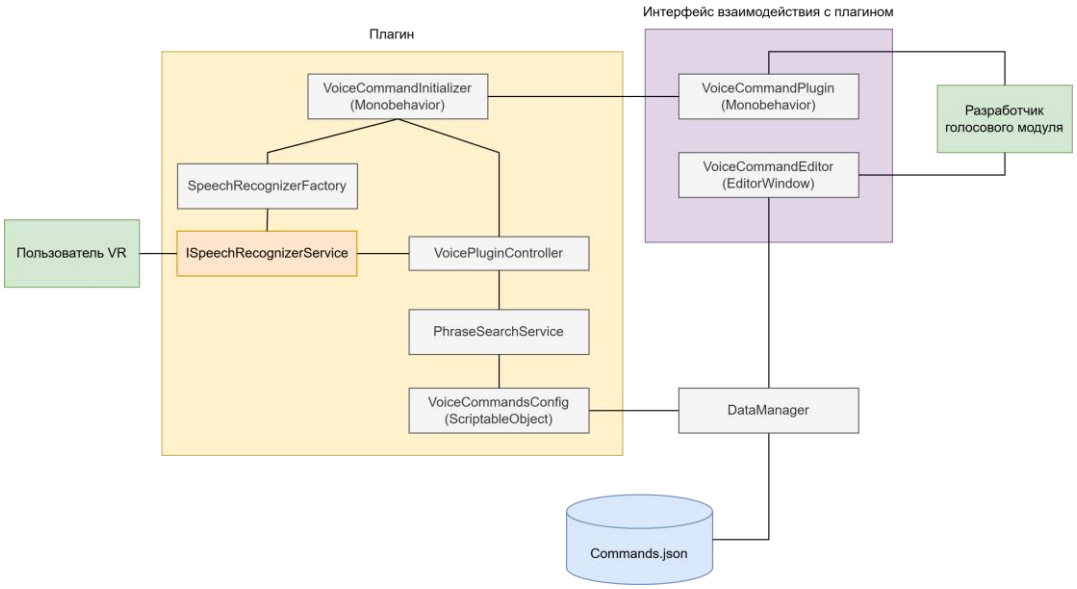

Рисунок 4 – Архитектура инструментария для создания голосового управления

Гибкость архитектуры обусловлена возможностью простой замены сервиса ASR путем добавления новой реализации интерфейса ISpeechRecognizerService.

Точкой входа является компонент VoiceCommandInitializer. При старте данный класс инициализирует необходимые для работы плагина сервисы:

1. Реализация интерфейса ISpeechRecognizerService, представленная классами для облачного и локального распознавания речи.
2. PhraseSearchService – сервис поиска метода по результату распознавания.

Конфигурация работы инструментария определяется разработчиком в компонентах VoiceCommandPlugin и VoiceCommandInitializer (рисунок 5).

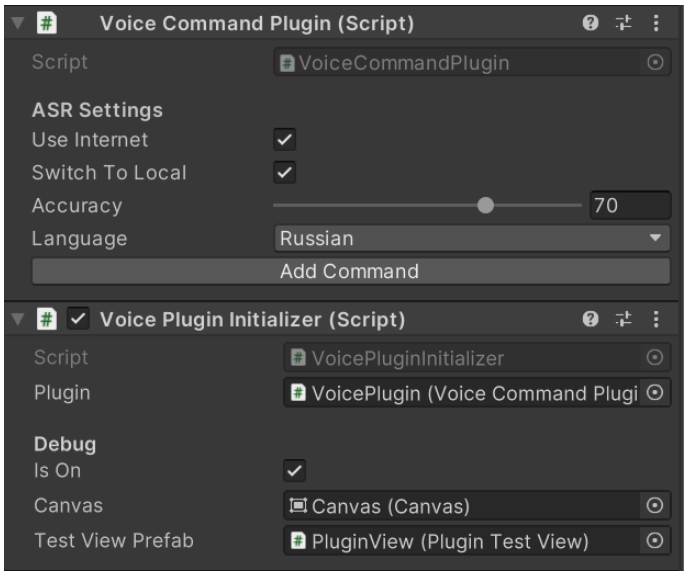

Рисунок 5 – Интерфейс для конфигурации параметров инструментария; Use Internet – выбор между локальным и облачным сервисами ASR, Switch To Local – переключение на локальное распознавание речи, если облачный сервис становится недоступным, Accuracy – точность распознавания, Language – выбор языка распознавания речи, Add Command – открытие окна VoiceCommandEditorWindow, [Debug] – Включение режима отладки

PhraseSearchService отвечает за сопоставление распознанного текста с набором голосовых команд в JSON-файле. Для реализации поиска команды использован FuzzySharp [21] – это библиотека для языка C# (.NET), предназначенная для нечеткого сравнения строк на основе вычисления расстояния Левенштейна. Она является портом Python-библиотеки FuzzyWuzzy для Python и позволяет находить степень схожести между строками, даже если они содержат опечатки, разный порядок слов или незначительные различия.

FuzzySharp предоставляет несколько стратегий для поиска. Была выбрана стратегия частичного поиска, которая позволяет находить фразы по входным данным, имеющим ошибки или опечатки. Для возможности произнесения команды в разном порядке был использован алгоритм токенизированного сравнения PartialTokenSetScorer.

Для ускорения доступа к командам результаты поиска кэшируются.

VoicePluginController управляет взаимодействием между компонентами инструментария и выступает в качестве системы вызова методов функциональной логики (рисунок 6).

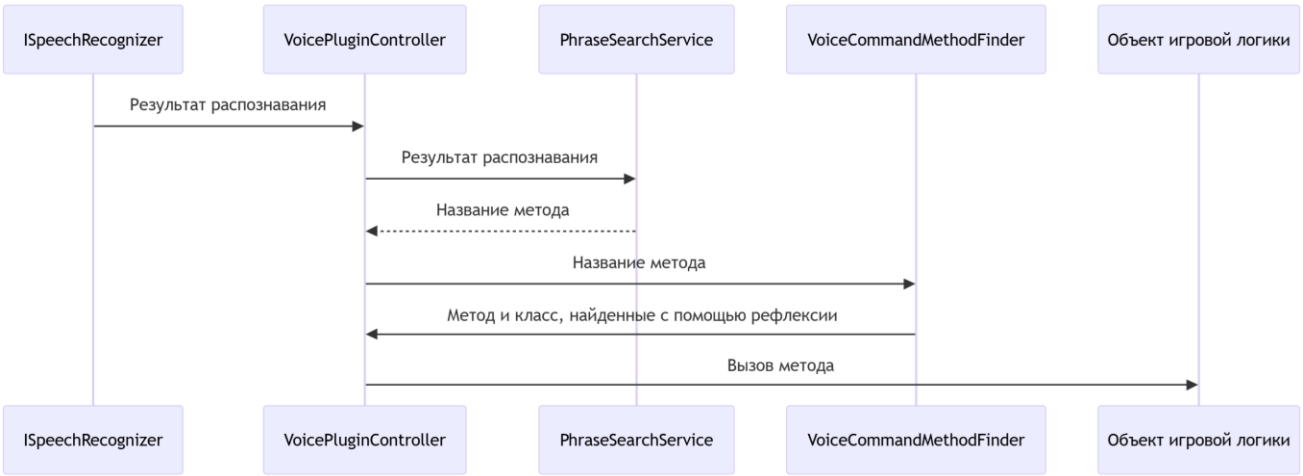

Рисунок 6 – Диаграмма последовательности работы VoicePluginController

Реализация голосового интерфейса с использованием разработанного инструментария

Для проверки работоспособности предложенного решения был реализован прототип голосового интерфейса, предназначенного для взаимодействия с виртуальной приборной панелью на сцене Unity.

Компонент PanelController содержит методы для управления приборами, помеченные атрибутом VoiceCommandMethod. Для каждого метода в редакторе VoiceComnmandEditor были сгенерированы соответствующие голосовые команды и их вариации, примеры которых представлены в таблице 4.

Таблица 4

Примеры соответствия голосовых команд и методов компонента PanelController

Голосовая команда	Вариации голосовых команд	Вызываемый метод
«Включить подсветку приборов»	«Зажги свет», «Включи освещение» и т.д.	SetPanelLight(true)
«Показать давление»	«Какое давление», «Давление за бортом» и т.д.	ShowPressureData()
«Установи курс на 270»	«Курс 270», «Поворот на 270»	SetAngle(270)

В результате система корректно распознавала голосовые команды пользователя и сопоставляла их с методами логики приложения. Выполнение действия происходило как при полном, так и при частичном совпадении произнесенной фразы с заданной голосовой командой. Команды, отсутствующие в конфигурационном файле, игнорировались системой.

Во время выполнения тестового сценария на VR-гарнитуре Pico 4 было произведено отключение доступа к сети, после чего через 10 секунд система переключилась с облачного сервиса ASR iFlyTek на локальное распознавание с помощью Vosk.

Проведенная апробация описанного голосового модуля показала, что разработанный инструментарий обеспечивает корректную интеграцию голосового управления в VR-приложения и демонстрирует устойчивость работы механизма поиска и сопоставления голосовых команд с системными методами и событиями. Данный инструментарий может быть использован для создания более сложных голосовых интерфейсов в интерактивных приложениях виртуальной и дополненной реальности.

Заключение

В ходе выполнения работы были получены следующие результаты:

1. Проведенный анализ существующих решений позволил выявить такие проблемы, как отсутствие пользовательского интерфейса для проведения полного цикла разработки модуля голосового управления, жесткая привязка к наличию или отсутствию доступа к сети, отсутствие устойчивости системы к различным формулировкам голосовой команды, а также риск утечки API-ключей к облачным сервисам.

2. Недостатки текущих решений, обозначенные в пункте 1, позволили обосновать актуальность разработки универсального инструментария для создания и интеграции голосовых интерфейсов, а также сформировать требования к нему.

3. Проведенное тестирование сервисов ASR показало, что облачный сервис iFlyTek Real-time ASR демонстрирует наименьшую среднюю задержку распознавания (2934 мс), что делает его оптимальным выбором для сценариев со стабильным подключением к сети. Среди локальных решений лучший результат показала библиотека Vosk (3589 мс).

4. Разработанный инструментарий реализует комплексный подход к созданию голосовых интерфейсов. Центральный элемент – визуальный редактор голосовых команд VoiceCommandEditor, обеспечивающий декларативное связывание фраз с логикой приложения с помощью атрибутов. Архитектура инструментария за счет использования абстракций предоставляет разработчику гибкий выбор между облачным и локальным сервисами ASR, а также дает возможность переключения между сервисами во время работы приложения. Разработанный сервис автоматической генерации вариантов голосовых команд обеспечивает устойчивость системы к разнообразию формулировок пользователя. Таким образом, предложенное решение позволяет существенно сократить трудозатраты на разработку и тестирование голосовых интерфейсов в VR и AR сценариях, направленных на решение различных задач.

5. Реализован прототип голосового интерфейса для управления бортовой панелью в тестовой сцене Unity. Прототип был запущен на VR-

гарнитуре Pico 4 и продемонстрировал корректность работы механизма распознавания голосовых команд и вызова соответствующих им методов, а также устойчивость системы к нестабильности доступа к сети. Это подтверждает возможность использования голосового инструментария при разработке голосовых модулей VR- и AR- приложений.

Конфликт интересов

Автор заявляет об отсутствии конфликта интересов.

Conflict of interest

The author declares no conflict of interest.

Список источников

1. M. E. McCullins, S. Hampton, S. G. Fussell, K. Kiernan, and J. Thropp, “The effectiveness of using virtual reality training environments for procedural training in fourth-generation airliners,” The Aeronautical Journal, vol. 129, no. 1342, pp. 3327–3346, 2025. doi:10.1017/aer.2025.10086
2. Кабанов А.А., Амосов М.В. VR/AR в изучении, создании и эксплуатации аэрокосмической техники: из макромира в микромир, от наблюдения к действиям // Труды МАИ. 2023. № 128. DOI: 10.34759/trd-2023-128-21
3. Поляков А.А., Защиринский С.А. Использование виртуального пространства для проведения макетно-конструкторских испытаний по электронному макету космического аппарата // Труды МАИ. 2019. № 107. URL: https://trudymai.ru/published.php?ID=107877
4. A. Siyaev and G.-S. Jo, “Neuro-Symbolic Speech Understanding in Aircraft Maintenance Metaverse,” IEEE Access, vol. 9, pp. 154484–154499, 2021, doi: https://doi.org/10.1109/access.2021.3128616.
5. V. Krishnamurthy, B Jafrin Rosary, G Oliver Joel, S. Balasubramanian, and S. Kumari, “Voice command-integrated AR-based E-commerce Application for Automobiles,” May 2023, doi: https://doi.org/10.1109/iconscept57958.2023.10170152.

6. Платформа Unity для разработки в реальном времени | Движок для 3D, 2D, VR и AR // Unity URL: https://unity.com/ru (дата обращения: 30.12.2025).

7. Game Voice Control [Offline speech recognition] | Audio | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/audio/game-voice-control-offline-speech-recognition-178047 (дата обращения: 30.12.2025).

8. cmusphinx/pocketsphinx: A small speech recognizer // GitHub URL: https://github.com/cmusphinx/pocketsphinx (дата обращения: 30.12.2025).

9. Meta - Voice SDK - Immersive Voice Commands | Integration | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/integration/meta-voice-sdk-immersive-voice-commands-264555 (дата обращения: 30.12.2025).

10. yasirkula/UnitySpeechToText: A native Unity plugin to convert speech to text on Android & iOS // GitHub URL: https://github.com/yasirkula/UnitySpeechToText (дата обращения: 30.12.2025).

11. Speech Control Plugin for VR | Audio | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/audio/speech-control-plugin-for-vr-76855 (дата обращения: 30.12.2025).

12. Speech-to-Text: AI voice typing & transcription // Google Cloud URL: https://cloud.google.com/speech-to-text (дата обращения: 30.12.2025).

13. N. Ashtari and P. K. Chilana, “How New Developers Approach Augmented Reality Development Using Simplified Creation Tools: An Observational Study,” Multimodal Technologies and Interaction, vol. 8, no. 4, p. 35, Apr. 2024, doi: https://doi.org/10.3390/mti8040035.

14. zai-org/GLM-4.7-Flash // Hugging Face URL: https://huggingface.co/zai-org/GLM-4.7-Flash (дата обращения: 30.12.2025).

15. Hugging Face – The AI community building the future. URL: https://huggingface.co/ (дата обращения: 30.12.2025).

16. XR Interaction Toolkit // Unity Documentation URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit%403.0/manual/index.html (дата обращения: 30.12.2025).

17. PICO Unity Integration SDK // PICO Developer URL: https://developer.picoxr.com/document/unity/ (дата обращения: 30.12.2025).

18. Short Form ASR WebAPI Document (Automatic Speech Recognition) // iFLYTEK Open Platform Documents URL: https://global.xfyun.cn/doc/asr/voicedictation/API.html (дата обращения: 30.12.2025).

19. Macoron/whisper.unity: Running speech to text model (whisper.cpp) in Unity3d on your local machine. // GitHub URL: https://github.com/Macoron/whisper.unity (дата обращения: 30.12.2025).

20. alphacep/vosk-unity-asr: Automatic Speech Recognition in Unity using Vosk library // GitHub URL: https://github.com/alphacep/vosk-unity-asr (дата обращения: 30.12.2025).

21. JakeBayer/FuzzySharp: C# .NET fuzzy string matching implementation of Seat Geek's well known python FuzzyWuzzy algorithm. // GitHub URL: https://github.com/JakeBayer/FuzzySharp (дата обращения: 30.12.2025).

References

1. M. E. McCullins, S. Hampton, S. G. Fussell, K. Kiernan, and J. Thropp, “The effectiveness of using virtual reality training environments for procedural training in fourth-generation airliners,” The Aeronautical Journal, vol. 129, no. 1342, pp. 3327–3346, 2025. doi:10.1017/aer.2025.10086

2. Kabanov A.A., Amosov M.V. VR/AR v izuchenii, sozdanii i ekspluatatsii aerokosmicheskoi tekhniki: iz makromira v mikromir, ot nablyudeniya k deistviyam // Trudy MAI. 2023. № 128. DOI: 10.34759/trd-2023-128-21

3. Polyakov A.A., Zashchirinskii S.A. Ispol'zovanie virtual'nogo prostranstva dlya provedeniya maketno-konstruktorskikh ispytanii po elektronnomu maketu kosmicheskogo apparata // Trudy MAI. 2019. № 107. URL: https://trudymai.ru/published.php?ID=107877

4. A. Siyaev and G.-S. Jo, “Neuro-Symbolic Speech Understanding in Aircraft Maintenance Metaverse,” IEEE Access, vol. 9, pp. 154484–154499, 2021, doi: https://doi.org/10.1109/access.2021.3128616.

5. V. Krishnamurthy, B Jafrin Rosary, G Oliver Joel, S. Balasubramanian, and S. Kumari, “Voice command-integrated AR-based E-commerce Application for Automobiles,” May 2023, doi: https://doi.org/10.1109/iconscept57958.2023.10170152.

6. Platforma Unity dlya razrabotki v real'nom vremeni | Dvizhok dlya 3D, 2D, VR i AR // Unity URL: https://unity.com/ru (accessed: 30.12.2025).

7. Game Voice Control [Offline speech recognition] | Audio | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/audio/game-voice-control-offline-speech-recognition-178047 (accessed: 30.12.2025).

8. cmusphinx/pocketsphinx: A small speech recognizer // GitHub URL: https://github.com/cmusphinx/pocketsphinx (accessed: 30.12.2025).

9. Meta - Voice SDK - Immersive Voice Commands | Integration | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/integration/meta-voice-sdk-immersive-voice-commands-264555 (accessed: 30.12.2025).

10. yasirkula/UnitySpeechToText: A native Unity plugin to convert speech to text on Android & iOS // GitHub URL: https://github.com/yasirkula/UnitySpeechToText (accessed: 30.12.2025).

11. Speech Control Plugin for VR | Audio | Unity Asset Store // Unity Asset Store URL: https://assetstore.unity.com/packages/tools/audio/speech-control-plugin-for-vr-76855 (accessed: 30.12.2025).

12. Speech-to-Text: AI voice typing & transcription // Google Cloud URL: https://cloud.google.com/speech-to-text (accessed: 30.12.2025).

13. N. Ashtari and P. K. Chilana, “How New Developers Approach Augmented Reality Development Using Simplified Creation Tools: An Observational Study,” Multimodal Technologies and Interaction, vol. 8, no. 4, p. 35, Apr. 2024, doi: https://doi.org/10.3390/mti8040035.

14. zai-org/GLM-4.7-Flash // Hugging Face URL: https://huggingface.co/zai-org/GLM-4.7-Flash (accessed: 30.12.2025).

15. Hugging Face – The AI community building the future. URL: https://huggingface.co/ (accessed: 30.12.2025).

16. XR Interaction Toolkit // Unity Documentation URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit%403.0/manual/index.html (accessed: 30.12.2025).

17. PICO Unity Integration SDK // PICO Developer URL: https://developer.picoxr.com/document/unity/ (accessed: 30.12.2025).

18. Short Form ASR WebAPI Document (Automatic Speech Recognition) // iFLYTEK Open Platform Documents URL: https://global.xfyun.cn/doc/asr/voicedictation/API.html (accessed: 30.12.2025).

19. Macoron/whisper.unity: Running speech to text model (whisper.cpp) in Unity3d on your local machine. // GitHub URL: https://github.com/Macoron/whisper.unity (accessed: 30.12.2025).

20. alphacep/vosk-unity-asr: Automatic Speech Recognition in Unity using Vosk library // GitHub URL: https://github.com/alphacep/vosk-unity-asr (accessed: 30.12.2025).

21. JakeBayer/FuzzySharp: C# .NET fuzzy string matching implementation of Seat Geek's well known python FuzzyWuzzy algorithm. // GitHub URL: https://github.com/JakeBayer/FuzzySharp (accessed: 30.12.2025).

Информация об авторах

Васильева Ксения Сергеевна, магистрант, Федеральное государственное автономное образовательное учреждение высшего образования «Казанский (Приволжский) федеральный университет», г. Казань, Россия; e-mail: ksenya_vasilieva@mail.ru

Information about the authors

Ksenia S. Vasilieva, master's student, Federal State Autonomous Educational Institution of Higher Education «Kazan (Volga Region) Federal University», Kazan, Russia; e-mail: ksenya_vasilieva@mail.ru

Получено 13 февраля 2026 ● Принято к публикации 12 марта 2026 ● Опубликовано 30 апреля 2026
Received 13 February 2026 ● Accepted 12 March 2026 ● Published 30 April 2026