УДК 004.4'2

ИСПОЛЬЗОВАНИЕ *UNITY* С *MEDIAPIPE* В РАЗРАБОТКЕ МОБИЛЬНЫХ *AR*/*VR* ПРИЛОЖЕНИЙ

Куклин Василий Алексеевич[1], Самойлов Юрий Евгеньевич[2]

[1]Студент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: wasyakuklin2@yandex.ru.

[2]Старший преподаватель;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: samoilov.yuri@yahoo.com.

Данная работа посвящена XR-технологиям, а именно области дополненной реальности (AR). В ней была разработана библиотека для Unity, обеспечивающая кроссплатформенный трекинг рук поверх стандартного фреймворка ARFoundation. Библиотека интегрируется в существующие AR-проекты, позволяя разработчикам добавить отслеживание 21 ключевой точки руки, с помощью ML-алгоритмов, запущенных с помощью фреймворка Mediapipe. Работа будет полезна желающим добавить распознавание жестов, манипуляции виртуальными объектами или иммерсивное взаимодействие с помощью рук в свои проекты на Unity ARFoundation для платформ Android и iOS.

Ключевые слова: *Unity*, *Mediapipe*, *AR*, смартфоны.

Для цитирования:

Куклин В. А., Самойлов Ю. Е. Использование *unity* с *mediapipe* в разработке мобильных *AR*/*VR* приложений // Системный анализ в науке и образовании. 2026. № 3. С. 22-33. EDN: EDIPRQ. URL: https://sanse.ru/index.php/sanse/article/view/733.

USE OF UNITY WITH MEDIAPIPE IN MOBILE AR/VR DEVELOPMENT

Kuklin Vasilii A.[1], Samoilov Yurii E.[2]

[1]Student:
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: wasyakuklin2@yandex.ru.

[2]Senior teacher:
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: samoilov.yuri@yahoo.com.

This paper is devoted to XR technologies, particularly AR technologies. A Unity library was developed to enable hand-tracking functionality in AR Foundation projects. The library can be integrated into existing AR projects, allowing developers to track 21 hand landmarks using ML algorithms powered by MediaPipe. This document will be useful for developers who want to add gesture recognition, virtual object manipulation, or immersive hand-based interaction to their Unity projects for Android and iOS.

Keywords: Unity, Mediapipe, AR, smartphones.

Статья находится в открытом доступе и распространяется в соответствии с лицензией Creative Commons «Attribution» («Атрибуция») 4.0 Всемирная (CC BY 4.0) https://creativecommons.org/licenses/by/4.0/deed.ru

For citation:
Kuklin V. A., Samoilov Use of unity with mediapipe in mobile AR/VR development. *System analysis in science and education*, 2026;(3):22-33 (in Russ). EDN: EDIPRQ. Available from: https://sanse.ru/index.php/sanse/article/view/733.

Введение

На сегодняшний день технологии дополненной реальности (*AR*) и виртуальной реальности (*VR*) активно развиваются, находя применение в образовании, медицине, промышленности и развлечениях [1, с. 1]. Рынок быстрыми темпами развивается: за 2025 год его стоимость увеличилась на 18,4%, и прогнозы показывают, что рынок и дальше будет расти быстрыми темпами [2, с 1].

Существует множество платформ, способных запускать приложения расширенной реальности. В основном можно выделить 3 группы устройств:

- *VR* гарнитура (*HTC*, *Oculus*);
- AR гарнитура (Hololens, Apple Vision Pro, Ray-Ban meta, Xreal);
- смартфоны (*iOS*, *Android*).

Отметим, что типы приложений для *AR* гарнитуры часто пересекаются с приложениями для смартфонов. И, если опускать возможности самых дорогих *AR* гарнитур (*Hololens*, *Apple Vision Pro*), которые уже скорее можно отнести к *MR* гарнитурам, основные типы приложений возможно реализовать как на *AR* гарнитуре, так и на смартфоне: перевод текста с изображения, синхронный перевод речи, размещение *3D*-объектов в пространстве, пространственная навигация, тур/гид по музеям.

Распространенной платформой для запуска *AR*-приложений являются смартфоны. Они доступны, есть у большинства пользователей и обладают достаточной вычислительной мощностью. Основными видами трекинга на платформах *Android* и *iOS* являются безмаркерные методы, использующие компьютерное зрение или инерциальные датчики (акселерометр, гироскоп и магнитометр). Эти подходы позволяют отслеживать плоскости поверхностей и движение камеры в пространстве без использования специальных маркеров, что упрощает опыт конечным пользователям.

Несмотря на развитость платформ, они уступают дорогим *MR* гарнитурам. Более дорогие платформы позволяют не просто видеть виртуальный мир, но и взаимодействовать с ним при помощи взгляда или рук. Эти способы взаимодействия реализованы на аппаратном уровне, и значительно меняют опыт использования *AR* приложений в лучшую сторону. На смартфонах пользователь ограничен касанием экрана, что отрицательно влияет на погружение и естественность взаимодействия.

Так как наличие качественного способа взаимодействия с виртуальной средой способно сильно повлиять на опыт пользователя, перенести лучшие практики на самую доступную среди конечных пользователей платформу (*Android*, *iOS*) будет целесообразным решением. Современные смартфоны обладают достаточной мощностью для запуска *ML* алгоритмов, способных компенсировать аппаратные ограничения платформы.

В работе приводятся результаты попытки принести на платформы *iOS, Android* опыт, который можно испытать на более дорогих платформах. Разработанный инструмент может использоваться будущими разработчиками *AR* приложений под *iOS* и *Android*, чтобы легко добавить в свои проекты естественный для пользователя способ взаимодействия с виртуальной средой.

1. Анализ предметной области

1.1 Техники трекинга окружения

Чтобы точно понять, почему опыт с дорогих платформ сложно перенести на платформы *Android* и *iOS*, необходимо знать существующие технологии трекинга. На рисунке 1 представлены основные техники трекинга.

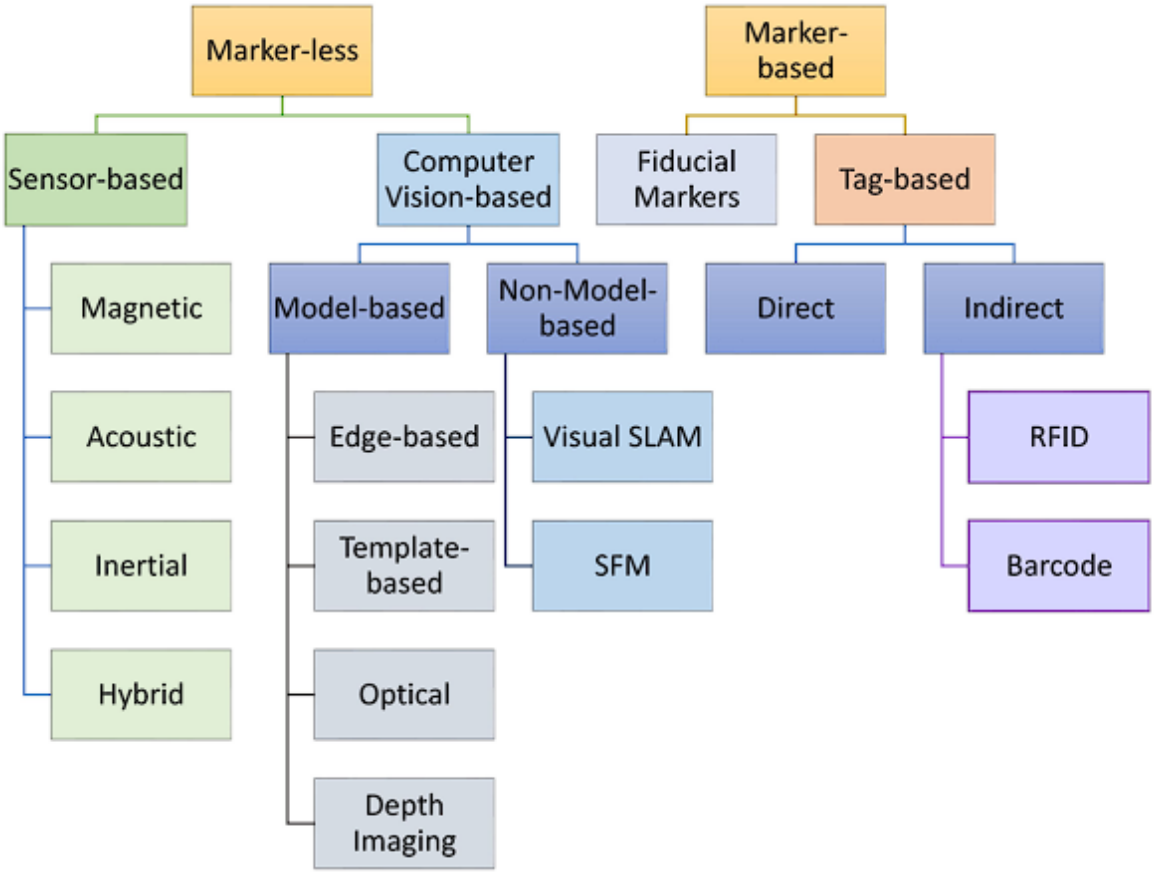


Рис. 1. Категоризация техник трекинга в дополненной реальности [1, с 7]

- *Marker-based* техники подходят для любых платформ, но требуют подготовки окружение (нанесения тегов на окружающие поверхности). Этот вариант подходит, для технологических стендов или музейных экспонатов.
- *Marker-less* техники, как правило, не требуют подготовки окружения для запуска приложений, но выставляют требования к платформе запуска.

При использовании *Marker-less* техник устройству необходимо либо иметь сенсоры, способные предоставлять данные об окружающем пространстве (*Sensor-based*), либо иметь достаточную вычислительную мощность, чтобы получать данные путем обработки изображений (*Computer Vision-based*).

Важно отметить, что при обработке видеопотока, требуются не только большие вычислительные ресурсы, но также увеличивается количество погрешностей, ошибок, связанных с ограничениями алгоритмов.

Чем дороже платформа запуска *XR* приложений, тем больше в ней сенсоров, позволяющих на аппаратном уровне получать данные об окружающем пространстве и положении устройства в нем. Смартфоны, в свою очередь не разрабатывались специально для запуска *XR*-приложений, но наличие в них камеры, а также датчиков: гироскопа и акселерометра делает возможным запуск *AR*-приложений.

Из-за ограничений целевых платформ, самым подходящим типом трекинга, который можно использовать для решения задачи является *Computer Vision-based* техника трекинга. Поэтому она будет использована при разработке решения.

1.2 Инструменты для разработки

Ознакомимся с основными инструментами для разработки *XR* приложений. Существуют открытые низкоуровневые стандарты и библиотеки, используемые для создания приложений расширенной реальности, например, *ARToolKit*, *OpenXR*, *Microsoft XNA*, *METAIO SDK*, *Open Scene*.

Подобные решения предполагают высокий порог вхождения и используются, только когда необходим полный контроль над кодом.

Как правило, большинство потребностей покрывают высокоуровневые *SDK*, привязанные к соответствующим платформам. Например, *ARKit* (*iOS*), *ARCore* (*Android*), *AR.js* (*WebXR*). В случае кроссплатформенной разработки, использование различных *SDK* приводит к надобности ведения нескольких кодовых баз.

Среди инструментов есть только одно решение, добавляющее слой абстракции перед всеми плагинами, позволяя даже при кроссплатформенной разработке вести только одну кодовую базу – фреймворк *ARFoundation* для *Unity*. В настоящее время даже большинство *standalone*-решений (например, *Vuferia*), доступны для использования в *ARFoundation* при подключении соответствующего плагина.

1.3 *Computer Vision-based* трекинг на базе *Mediapipe*

Проблема трекинга рук с использованием монокулярной камеры уже привлекала внимание крупных технологических компаний. В частности, компания *Google* предложила решение в рамках своего кроссплатформенного фреймворка *Mediapipe* – открытой библиотеки для запуска моделей машинного обучения в реальном времени [3, с. 1]. Подобное решение идеально подходит под нашу задачу. *Mediapipe* написан на *C++*, но существует плагин, интегрирующий его в *Unity*, путем переписи исходных файлов фреймворка на язык, используемый в движке (*C#*).

На платформе *GitHub* лежит много проектов, пытающихся использовать *Mediapipe* вместе с *Unity* для создания различных проектов. Однако, большинство из них либо используют оригинал фреймворка, используя *C++* или *Python* (что привязывает проект к *PC* платформе), либо не адаптированы под *ARFoundation* – ключевой компонент для создания современных *AR*-приложений на *Unity*.

Это создаёт серьёзные трудности при попытке внедрения таких решений в готовые *AR*-проекты, требуя значительных доработок и модификаций.

2. Формирование технологического стека

Как было сказано выше, из-за ограничений платформ *Android* и *iOS*, в работе необходимо использовать *Computer Vision-based* техники трекинга.

Для запуска вычислений, связанных с выбранным способом трекинга, будет использоваться *Mediapipe.* Его использование позволит сохранить кроссплатформенные преимущества движка, а также сильно упростит реализацию решения.

Преимущества *Unity* быстро сделали его стандартом для индустрии [1, с. 12]. Будущее решение обязательно должно интегрироваться в движок, чтобы существующие разработчики *XR*-приложений могли его использовать.

Интеграция решения с *Unity* предполагает использование *ARFoundation*, ведь именно на основе этого фреймворка разрабатывается большинство *AR*-приложений. Используя расширения *ARCore* и *ARToolKit*, у разработчиков будет возможность использовать нативные для платформ *SDK*.

Наличие созданного сообществом плагина для интеграции *Mediapipe* в *Unity* делает разумным выбор связки *Unity* и *Mediapipe* как основной стек будущего решения.

Важно отметить одно ограничение, связанное с выбранным способом трекинга – в случаях, когда в кадр не попадают пальцы руки, модели приходится предсказывать их положение. В такие моменты предсказанное положение может не совпадать с реальным.

Эта проблема вытекает из ограничений платформы. Ее невозможно решить, не добавляя дополнительных датчиков к рукам. Использующим решение необходимо учитывать это при разработке своих продуктов.

3. Архитектура решения

Разберем модульную структуру первой стабильной версии получившегося инструмента. Разработанные классы были разделены на 6 модулей:

- CoreComponents;
- DefaultComponents;
- ExtraComponents;
- DrawingExampleComponents;
- PuzzleExampleComponents.

3.1 *CoreComponents*

Необходимый и достаточный модуль для работы с библиотекой. Он содержит все необходимое для запуска *ML* решения. Как видно на рисунке 2, модуль состоит из главного класса, запускающего *Mediapipe*, а также интерфейсов, которые пользователю необходимо реализовать для взаимодействия с библиотекой.

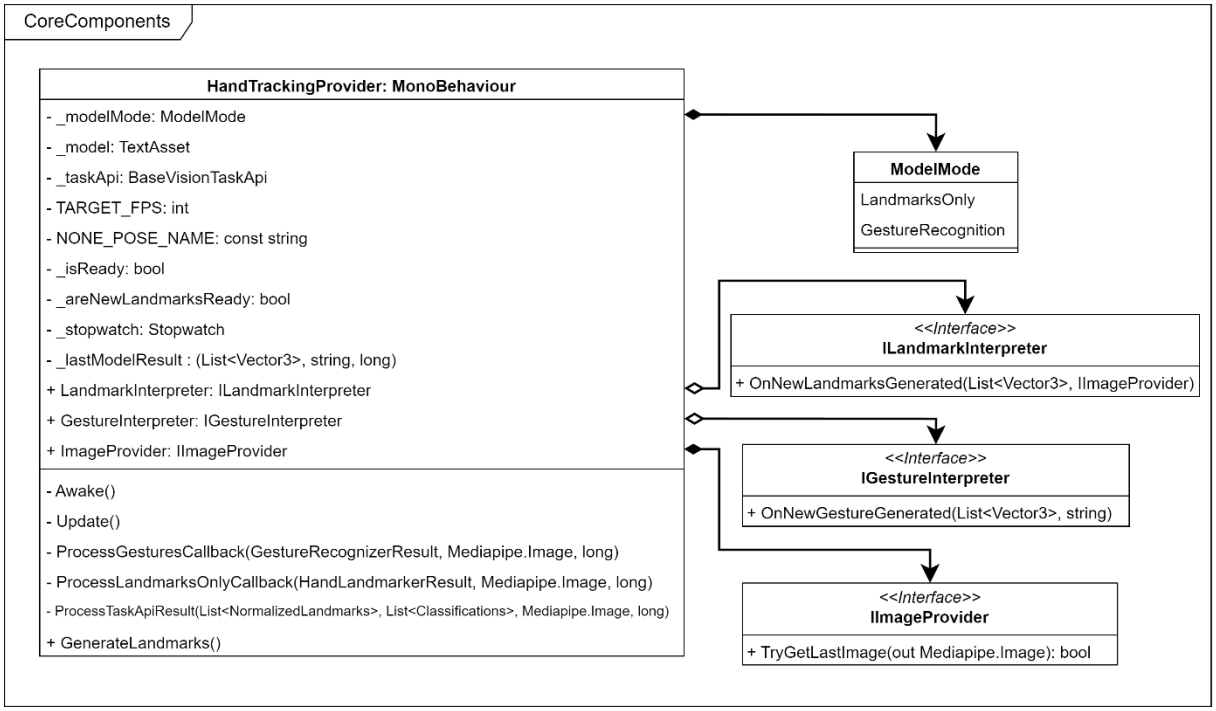


Рис. 2. Диаграмма классов модуля CoreComponents

3.2 *DefaultComponents*

Модуль состоит из готовых имплементаций интерфейсов из модуля *CoreComponents*. Общую схему модуля и связей внутри модуля видно на рисунке 3.

Используя эти реализации, разработчик способен быстро добавить в свои проекты следующие функции:

- отслеживание разметки руки на основе изображения с камеры смартфона (*CameraImageProvider*, *LandmarkInterpretator*);
- базовая интепретация разметки в виртуальном пространстве (*JitterFilter*, *CameraDepthModifier*);
- распознавание жестов (*PoseDetectBus*, *SimplePose*).

Отметим, что для распознавания доступно 8 жестов, предусмотренных в модели от *Mediapipe* [4, с. 1]. У разработчиков есть возможность расширить набор жестов за счет дообучения модели [5, с. 1], или реализации интерфейса *IDetectablePose*.

Пользователь может использовать этот модуль в случае, если он хочет скорее начать разработку своих проектов, сведя работу с библиотекой к минимуму. Подробности реализации классов и примеры их использования раскрыты в соответствующей главе руководства по эксплуатации библиотеки.

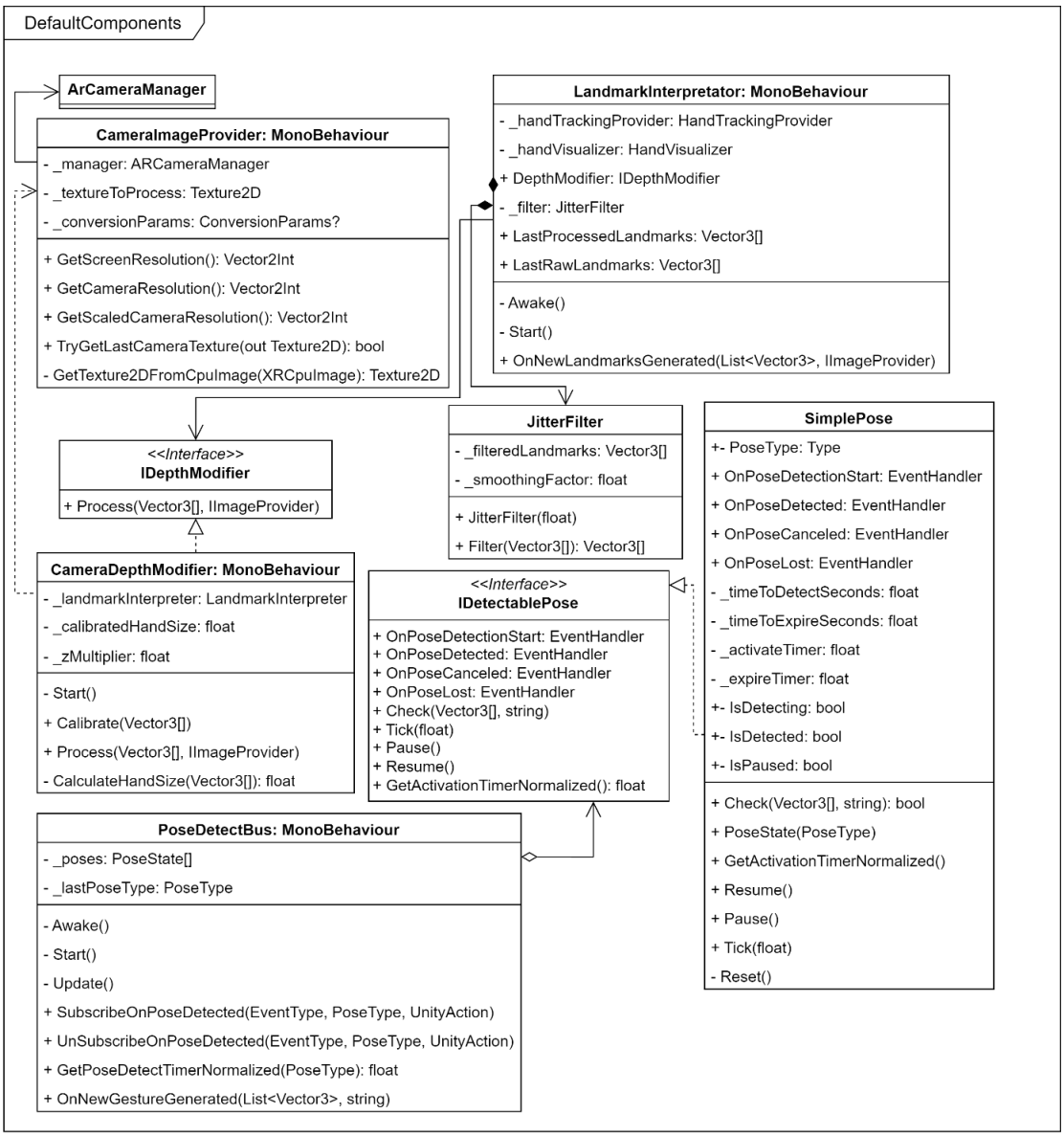


Рис. 3. Диаграмма классов модуля DefaultComponents

3.3 *ExtraComponents*

Модуль дополнительных компонентов, которые могут быть интересны разработчикам при разработке своих приложений. Сюда входит класс для отображения в реальном времени результатов разметки вместе с направлением нормали ладони (*HandVisualizer*). Также в модуль входит класс *VRMode*, с помощью которого можно быстро адаптировать свое приложение под *VR*-очки типа *Google Cardboard.*

Для работы модулей класса необходим модуль *DefaultComponents*. Классы модуля видно на рисунке 4.

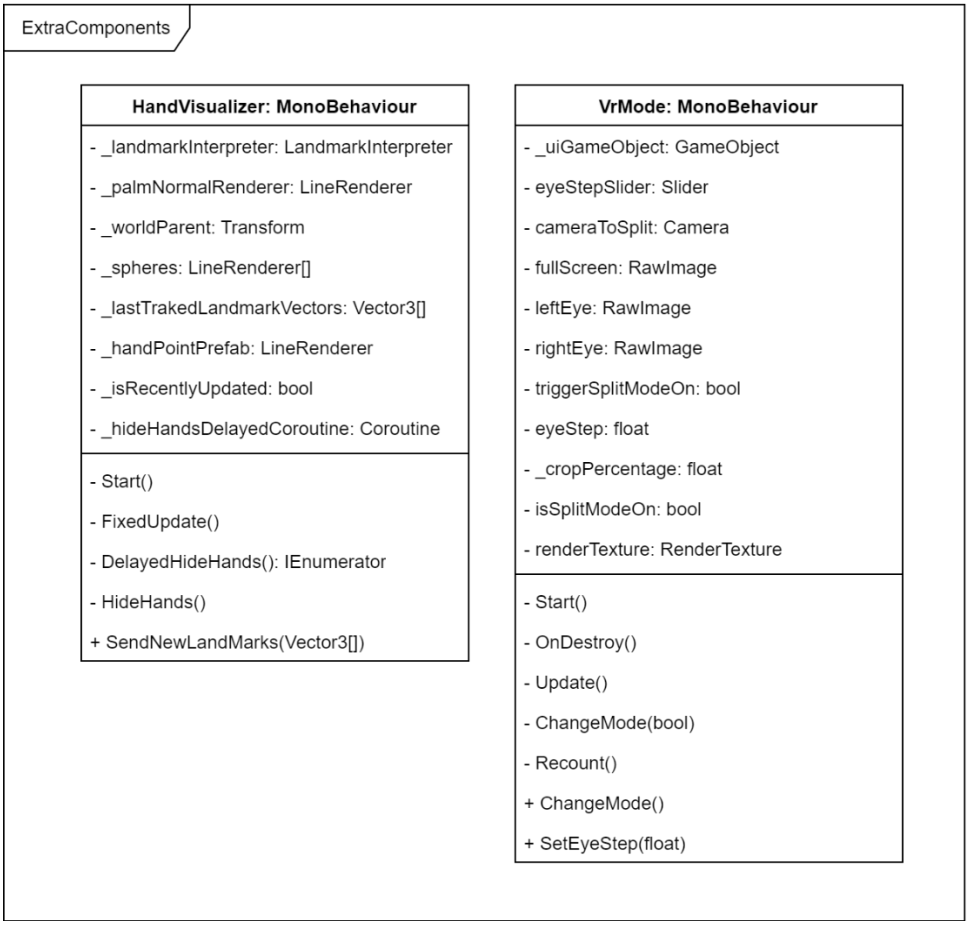


Рис. 4. Диаграмма классов модуля ExtraComponents

3.4 *DrawingExampleComponents и PuzzleExampleComponents*

Модули содержат компоненты, используемые в демо-приложениях, построенных с помощью данной библиотеки. На рисунке 5 видны классы приложения «Пазлы», а на рисунке 6 – приложения «*AR* рисование».

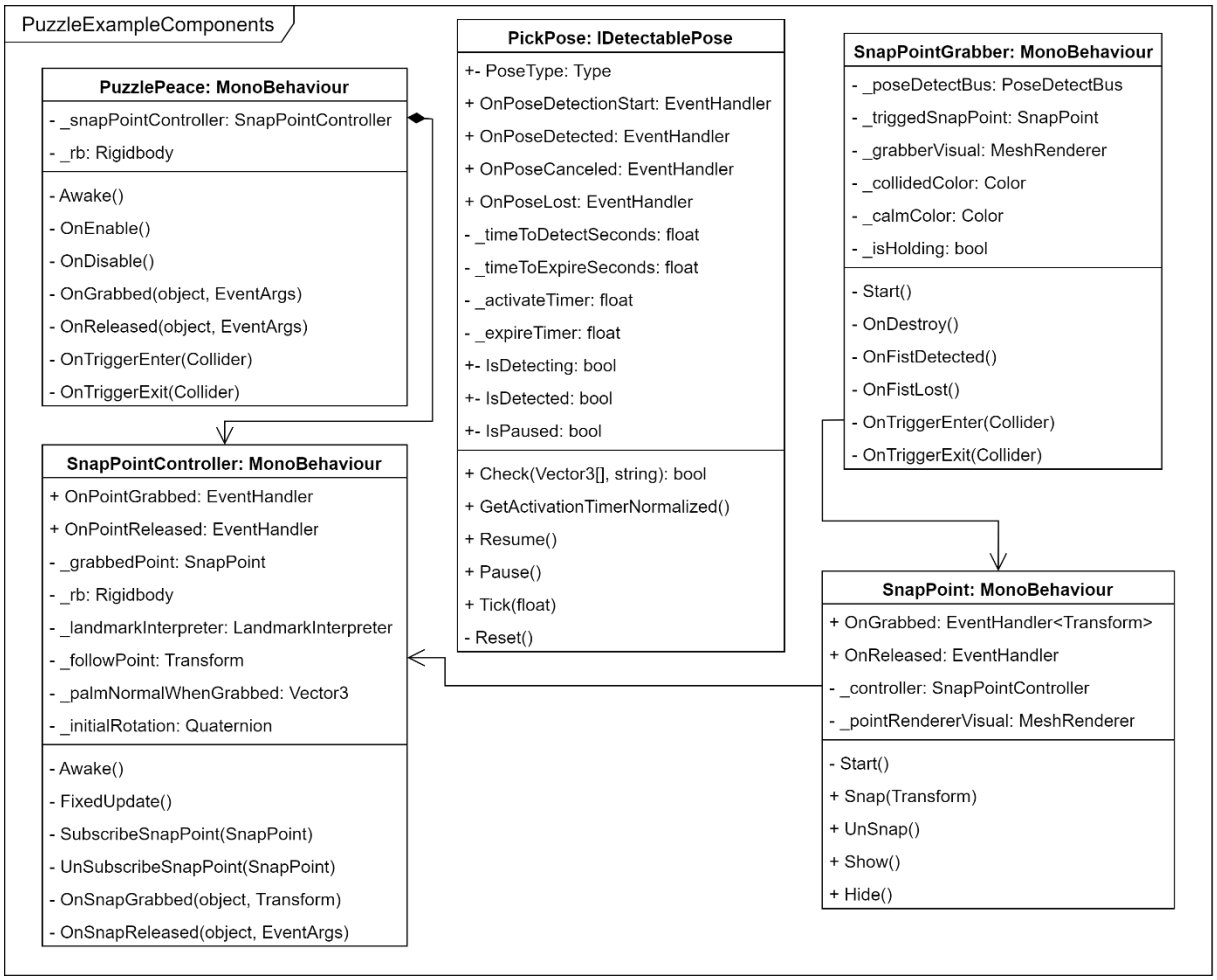


Рис. 5. Диаграмма классов модуля PuzzleExampleComponents

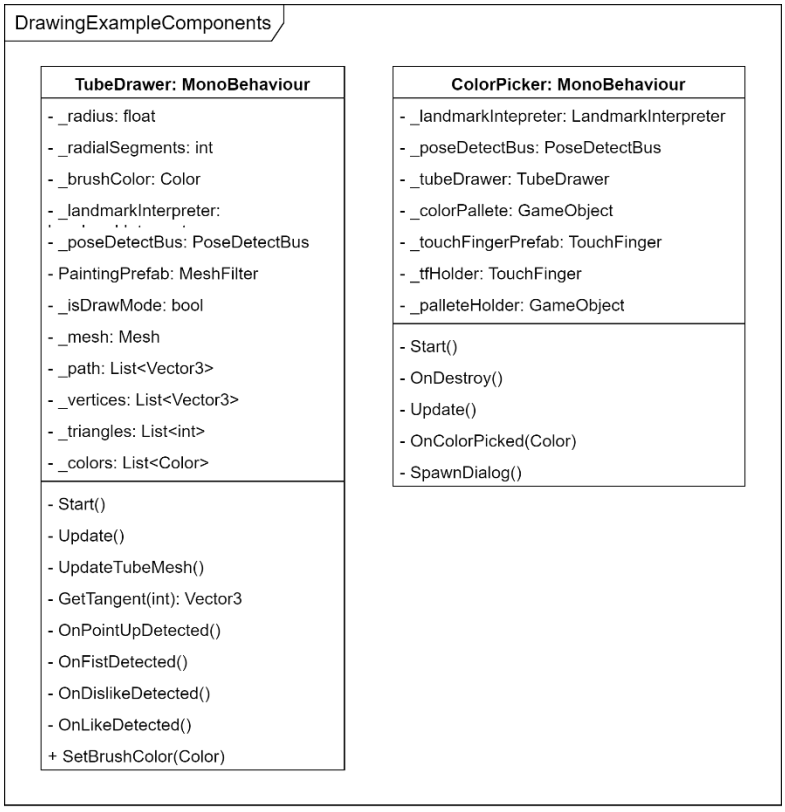


Рис. 6. Диаграмма классов модуля DrawingExampleComponents

4. Демонстрационные примеры

Чтобы у разработчиков была возможность быстро погрузиться в итоговую библиотеку, было создано 2 примера её использования «*AR* Рисование» и «*AR* Пазлы».

4.1 Приложение «*AR* Рисование»

Первым примером использования библиотеки стало приложение «*AR* Рисование». В нем можно рисовать объемные линии в воздухе (рис. 7). На выбор есть 3 цвета. Все управление реализовано жестами. Рисунок корректно запоминает свое положение в пространстве, и на него можно смотреть с разных ракурсов (рис. 8 и 9). Данный пример позволяет отлаживать алгоритм вычисления глубины. На рисунках 8 и 9 можно видеть, что иногда алгоритм может быть неточным.



Рис. 7. Процесс рисования в приложении «AR Рисование»

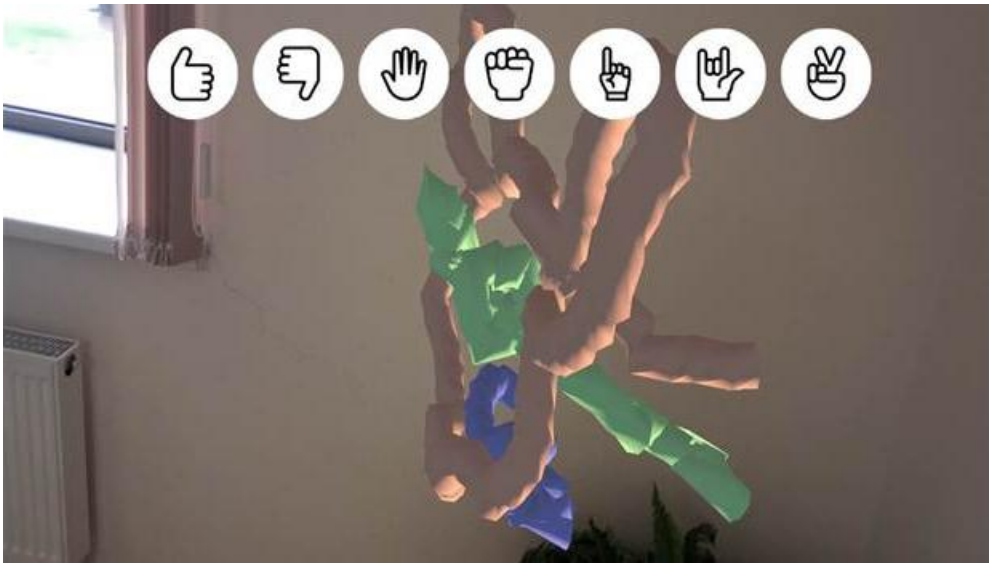


Рис. 8. Рисунок в дополненной реальности с правого ракурса

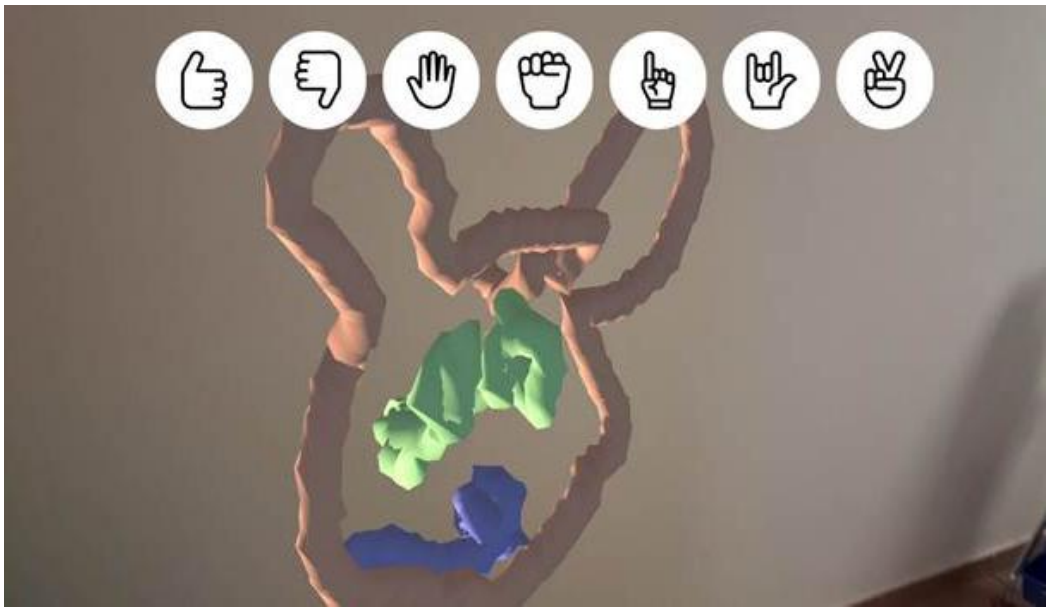


Рис. 9. Рисунок в дополненной реальности с левого ракурса

4.2 Приложение «Пазлы»

Второй пример показывает не просто возможность разместить объект в пространстве, но также и взаимодействие с виртуальной средой.

В приложении перед пользователем появляется 2 доски – с фрагментами изображения (пазлами) и самим изображением (рис. 10). Игроку необходимо сопоставить фрагменты с исходным изображением и перенести каждый фрагмент на свое место (рис. 11). Пазлы являются физическими объектами – на них действуют силы гравитации, и они имеют свои физические границы. Помимо этого, пользователь может двигать сами доски (рис. 10), чтобы расположить их так, как ему удобно.

Стоит отметить, что все взаимодействие происходит с помощью жеста, который был добавлен без дообучения встроенной в *Mediapipe* модели. Поза *Pick* считается активной, когда расстояние между кончиками большого и указательного пальца меньше определенного порога. Возможность добавить новый жест для распознавания является важной для решения функцией.

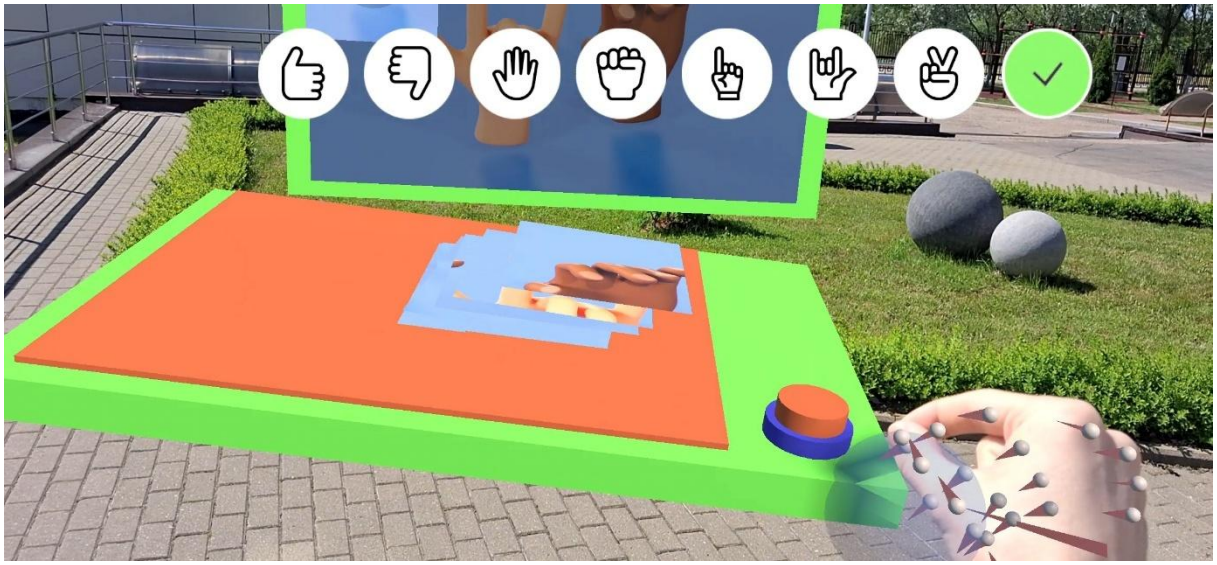


Рис. 10. Взаимодействие игрока с доской

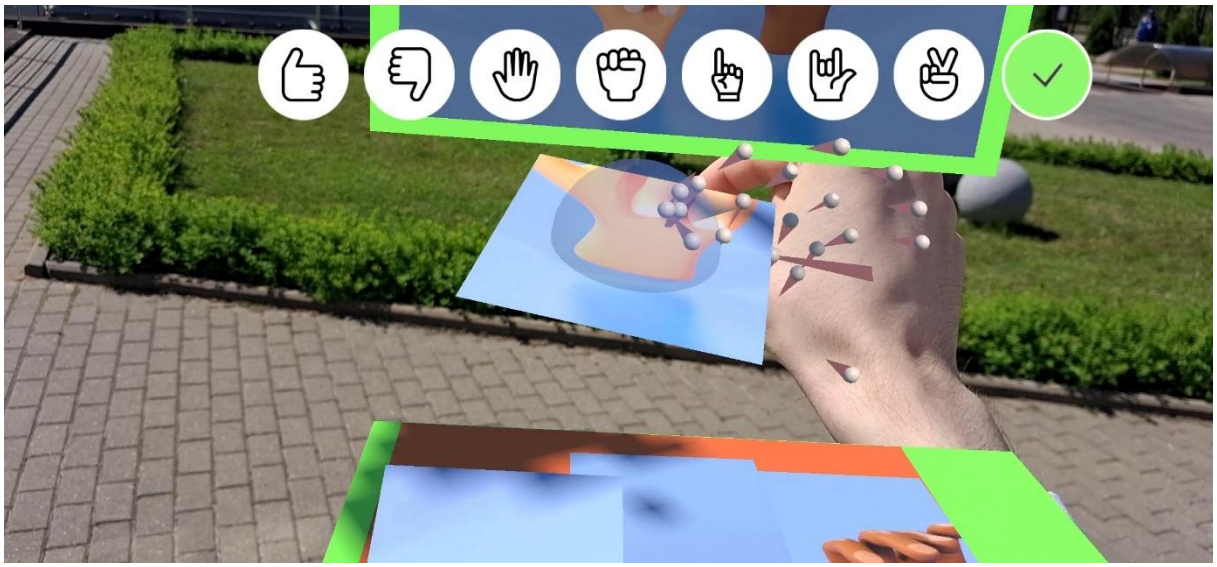


Рис. 11. Взаимодействие игрока с пазлами

5. Репозиторий

В результате работы сформировался *GitHub* репозиторий, доступный по следующей ссылке: *https://github.com/Baykoku-cs/HandsForMobileAR*. Правила работы с ним расписаны в *README* файле репозитория. Рассмотрим получившийся репозиторий.

5.1 Лицензия

Решение распространяется по *MIT* лицензии, так же, как и *MediapipeUnityPlugin*, который является основой библиотеки. Любой вправе свободно изменять и использовать исходный код решения, но продукт предоставляется без каких-либо гарантий.

5.2 Версии библиотеки

У проекта есть первая стабильная версия – 0.1.0. Она содержит все модули, описанные в этой работе. Перед публикацией плагин был проверен: после добавления его и соответствующих зависимостей (процесс запуска библиотеки описан в документации) в пустой проект, демо-сцены корректно запускаются и работают.

5.3 Документация

В папке *docs* можно найти всю информацию, касающуюся проекта, тем не менее, перед погружением в документацию важно прочитать *README* репозитория.

В с документацией можно найти:

- Руководство пользователя в форматах *.pdf* и *.docx*. Следуя руководству, пользователь реализует каждый из модулей и будет отлично ориентироваться в библиотеке.
- Документация к *api* решения. Этот документ пользователь может использовать, чтобы быстро освежить память по модулям библиотеки.

Заключение

В ходе работы было создано решение, расширяющее возможности *ARFoundation* и позволяющее добавить в свои проекты дополненной реальности трекинг рук.

Архитектура решения гибкая – разработчик сможет с легкостью расширить и модифицировать продукт так, чтобы он подходил для решения его конкретных задач. Отметим, что базовая метрика оценки *.NET* кода – индекс удобства поддержки [6] равен 84 для всей библиотеки (95 для модуля *CoreComponents*).

Большая часть проекта сопровождается документацией. Благодаря демо-примерам у разработчиков будет возможность быстро освоить библиотеку и начать разработку собственных приложений. Модульная архитектура решения позволит удобно импортировать в свой проект только нужный разработчику контент.

Проект продолжит активно развиваться. Существует множество различных функций, которые расширят библиотеку желаемыми возможностями. Например, одними из запланированных изменений можно назвать:

- компиляция плагина *Mediapipe* с модулями для запуска библиотеки в *GPU* режиме;
- добавить возможность пользователю настраивать количество отслеживаемых моделью рук;
- адаптация под вертикальную ориентацию смартфона;
- разработать компонент с настройкой окклюзии.

Список источников

1. In-Depth Review of Augmented Reality: Tracking Technologies, Development Tools, AR Displays, Collaborative AR, and Security Concerns / T. A. Syed, M. S. Siddiqui, H. B. Abdullah [et al.] // Sensors. – 2022. – Vol.23, Iss.1. – https://doi.org/10.3390/s23010146.
2. Augmented Reality and Virtual Reality Market. – Marketsandmarkets Research Private Ltd., 2025. – URL: https://www.marketsandmarkets.com/Market-Reports/augmented-reality-virtual-reality-market-1185.html (дата обращения 18.10.2025).
3. Mediapipe hands: On-device real-time hand tracking / F. Zhang, V. Bazarevsky, A. Vakunov [et al.] // arXiv : [электронный архив]. - arXiv:2006.10214.– https://arxiv.org/abs/2006.10214.
4. MediaPipe Hand Gesture Classification model card. – URL: https://storage.googleapis.com/mediapipe-assets/gesture_recognizer/model_card_hand_gesture_classification_with_faireness_2022.pdf (дата обращения 17.04.2026). – Based on Model Cards for Model Repoing, In Proceedings of FAT* Conference (FAT*2019). ACM, New York, NY, USA
5. Hand gesture recognition model customization guide // Google Developers. – URL: https://ai.google.dev/edge/mediapipe/solutions/customization/gesture_recognizer (дата обращения 28.04.2026)
6. Значения метрик кода // Microsoft Learn: важная информация всегда под рукой. – Microsoft, 2026. – URL: https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values (дата обращения 28.05.2026).